Rethinking Multilevel Selection in Genetic Programming

Shelly X. Wu
Computer Science Department
Memorial University of Newfoundland
St John's, Canada, A1B 3X5
xiaonan@mun.ca

Wolfgang Banzhaf
Computer Science Department
Memorial University of Newfoundland
St John's, Canada, A1B 3X5
banzhaf@mun.ca

ABSTRACT

This paper aims to improve the capability of genetic programming to tackle the evolution of cooperation: evolving multiple partial solutions that collaboratively solve structurally and functionally complex problems. A multilevel genetic programming approach is presented based on a new computational multilevel selection framework [19]. This approach considers biological group selection theory to encourage cooperation, and a new cooperation operator to build solutions hierarchically. It extends evolution from individuals to multiple group levels, leading to good performance on both individuals and groups. The applicability of this approach is evaluated on 7 multi-class classification problems with different features, such as non-linearity, skewed data distribution and large feature space. The results, when compared to other cooperative evolutionary algorithms in the literature, demonstrate that this approach improves solution accuracy and consistency, and simplifies solution complexity. In addition, the problem is decomposed as a result of evolution without human interference.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods; I.5.2 [**Design Methodology**]: Classifier design and evaluation

General Terms

Algorithms, Design, Experimentation

Keywords

Cooperative evolutionary algorithms, Multilevel selection, Hierarchical model, Emergent decomposition, Coevolution, Multi-class Classification

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

Genetic programming has proven to be an efficient and powerful problem-solving strategy. However, like all evolutionary algorithms, it is not a panacea; it has difficulty solving high dimensional, multimodal problems, which normally mean huge search spaces, or complicated fitness definitions, or expensive fitness evaluations. An effective way to solve increasingly complex problems is to explicitly evolve solutions in the form of interacting coadapted subcomponents [12]. Evolutionary algorithms that searching for such cooperative subcomponents are called cooperative evolutionary algorithms (CEAs), in which individuals represent partial solutions.

With respect to cooperative GP, considerable research has been devoted to investigate team evolution, major research work including GP Teaming [3], Orthogonal Evolution of Teams (OET) [14], and Symbiotic Bid-Based GP (SBB) [10]. A team, referring to a solution, is an aggregation of some individuals (subcomponents) that are evolved simultaneously in one population. It is important to note that GP in team evolution conducts not merely a multimodal search; the cooperative interactions between subcomponents are also taken in account.

This paper takes a different approach to tackle the evolution of cooperation which is based on a computational multilevel selection framework [19]. This framework on one hand captures the gist of multilevel selection theory (MLS) [16, 17, 20] in biology to encourage cooperation; on the other hand, it extends MLS to hierarchically create solutions for complex problems from simple subcomponents. The goal of this study is to discuss how to map the abstract framework to a GP implementation called Multilevel Genetic Programming (MLGP). We examine the applicability of MLGP in multi-class classification (MCC) problems. The problem space of MCC contains specialized sub-domains that may likely be overlooked by a single monolithic classifier; a feasible approach is to use multiple but simpler classifiers which co-adapt to balance the detection rate and false alarm rate of final solutions. Because of this, problem decomposition is impossible to assess prior to runs. Therefore, MCC problems are particularly suitable for testing MLGP's ability for cooperation and problem decomposition. MLGP is compared with traditional Linear GP, OET, SBB, and XCSR (a real-valued XCS classifier system) on 7 UCI benchmark datasets. The results indicate MLGP is more accurate and consistent with respect to classification performance. We also show that MLGP returns the simplest solution on all datasets,

and adapts the complexity of solutions to the dataset's level of difficulty.

2. RELATED WORK

Based on how final solutions are presented, CEAs can further be categorized into population-based methods and team-based methods. In population-based methods, the entire population becomes the solution of targeted problems, such as in Learning Classifier Systems (LCS) [7] and the work described in [5, 8]. One of their drawback is not measuring the performance of subcomponents as a whole. Without such a measurement, evolved individuals are not sufficient to consistently provide cooperative behavior, and the completeness of final solutions cannot be guaranteed [14, 19].

Team-based methods overcome this disadvantage by introducing the idea of teams. Well-known team-based methods include GP Teaming [3], Orthogonal Evolution of Teams (OET) [14], Cooperative Coevolutionary Evolutionary Algorithms (CCEA) [12][1], Individual Evolution (IE) [4, 11] and Symbiotic Bid-Based GP (SBB) [10].

Teams are built in two different ways. In CCEA, SBB, and IE, a certain number of individuals are selected from the population to form a team. Therefore, the problem of credit assignment has to be solved to share team fitness among team members according to their contribution to a solution, preferring team members with non-overlapping behaviors [15]. A limitation of CCEA and IE, when compared to SBB, is that *a priori* knowledge is needed to decide the number of individuals required in a team; such information may sometimes be not available. The other way to form a team, like in GP teaming and OET, is to explicitly specify team representation and team composition. Teams are then treated as objects of evolution, and their survival probability fully affects their members'. The strong coupling between teams and their members eliminates the credit assignment problem, but it also misjudges individual contribution based only on collaborative performance; as a result, good team members are at the risk of loosing reproductive opportunities because they might be teamed with free-riders or less fit individuals. In addition, the problem decomposition has to be done manually.

To create successful teams, collaboratively members must work to improve the performance of the team as a whole, and individually they must be relatively successful at their own distinct subtasks [14]. All team-based methods listed above define a team fitness function to meet the first criterion. In every generation at least one team is formed explicitly or implicitly, and has its fitness evaluated. Team fitness shapes the interactions between team members, and guides evolution to approach the goal of cooperation. Unfortunately, only some of the methods take advantage of such a measurement to optimize team performance through team competition; GP Teaming considers team fitness at group reproduction, while OET and SBB at survival selection. With respect to the second criterion, obviously individual fitness is required. However, only GP Teaming, OET, and IE define individual fitness. Without such an accurate measure, according to [14, 19], even though teams perform relatively well, their members tend to perform relatively poor, which prevents further improvement of team performance.

[1]CCEA can be regarded as a special case of team-based methods. Although it does not define teams explicitly, it forms teams temporarily at individual fitness evaluation.

3. MULTILEVEL GP

In nature, the success of cooperation can be witnessed at all levels of biological organizations. A growing number of biologists have now come to believe that the theory of group selection is the explanation [1]. Individuals who cooperate with others group members, such when hunting as a team or watching predators for others, have a greater chance to survive severe competition.

In group selection theories [16, 17, 20], individuals are divided into groups, where they only interact with members of the same group. Selection operates within groups and between groups. Within-group selection equals natural selection as commonly understood; it selects individuals in a group proportional to fitness. Individuals, therefore, compete against each other in the pursuit of their own interests. Between-group selection, in contrast, examines the total productivity of groups, and prefers the group with the best performance or the group whose individuals cooperate best. Hence it forces individuals to coadapt so that a cohesive group can be formed. It also resolves and reduces conflicts within groups, because conflicts would compromise group performance. In short, competition between groups encourages the emergence of cooperation within groups. One key point in group selection theory is to relate individual reproduction probability to group performance; for example, in reproduction or survival selection, a group is always selected first, from which an individual is selected as parents or for replacement. Individuals and groups are relative in group selection models: groups can be regarded as individuals on a higher level, so that a new level of dynamics can act upon them. In this way, a hierarchical or nested structure can be constructed. This new perspective is now called multilevel selection (MLS) theory [16].

Inspired by MLS, Wu and Banzhaf [19] proposed a new computational multilevel selection framework. This framework extends evolution from individuals to multiple group levels, through which cooperative solutions can be hierarchically built out of simple ones. Multilevel Genetic Programming (MLGP) is a new genetic programming variant based on this framework. To better understand the working mechanism and features of MLGP, we consider MCC problems as an example. Before a detailed description of MLGP, we define, first, the representation of individuals and groups, and their fitnesses functions for MCC problems.

MLGP evolves two types of entities: individuals and groups. Individuals in the context of MCC are binary classifiers whose chromosome contains a program evolved by Linear GP and a class label. During training phase when given an input example, if an individual's program returns a value greater than 0, then this individual is said to be accurate at classifying the input data provided their class label matches; otherwise, this individuals misclassifies the input data. The individual fitness, therefore, is defined as the following:

$$f_i = TPR_i \times (1 - FPR_i)^2 \quad (1)$$

where TPR_i and FPR_i are the true positive rate (TPR) and false positive rate (FPR) of individual i, respectively. The FPR is given more weight here to encourage low misclassification errors.

Groups are compositions of existing individuals and groups. Their fitness is as follows:

$$g_j = \frac{\sum_{i=0}^{n} f_i}{n} \times \frac{C_{covered}}{C_{total}} \times \frac{N_{correct}}{N_{total}} \times \sqrt{\frac{2 + GS_j}{2 \times GS_j}} \quad (2)$$

where the first term is the average individual fitness of group j, the second term is the class coverage, which is the number of classes covered by group j over the total number of classes in the training dataset, the third term is the data coverage, defined as the number of correctly classified data samples by group j over the total number of training data samples, and the last term is a normalized term to control the size of group j (GS_j). The second and third terms together rate group coverage. Obviously, this fitness function is a measure of how group members collaboratively increase overall data coverage on all classes and individually maximize their own classification accuracy with as few members as possible.

MLGP, as shown in Algorithm 1, is completed in 5 steps:

Algorithm 1: Multilevel Genetic Programming

```
1  P ← Initialize_Population(N);
2  Evaluate_Individual_Fitness (P);
3  while population does not converge or maximum
   generation is not reached do
4      for i ← 0 to m do
5          gp ← Reproduce_a_Group(P);
6          Evaluate_Group_Fitness(gp);
7          Add_a_Group_to_Population(gp, P);
8      end
9      for i ← 0 to n do
10         idv ← Reproduce_an_Individual(P);
11         Evaluate_Individual_Fitness (idv);
12         Add_an_Individual_to_Population(idv, P);
13     end
14     Niching_on_Individuals ();
15     Niching_on_Groups ();
16     P' ← Survival_Selection (P, N, M);
17     P ← P';
18 end
```

Initialization Line 1 and 2 randomly initialize the population with N individuals, each with a unique ID. Individuals become the lowest level in the hierarchical structure. Class labels from training datasets are randomly assigned to individuals as their class labels. Individuals are independent and competitive with each other, without being aware of collaborative goals.

Evolution on group levels Line 4 to 8 explain the evolution on group levels. In every generation, up to m new groups are created by three evolutionary operators, i.e. cooperation, crossover and mutation, with a user-defined probability. Cooperation mixes individuals and groups on all levels together, from which two entities are selected to form a new group. If individuals with the same ID appear in a group, only one individual is kept. Crossover and mutation, in contrast, select parents from groups only. Crossover exchanges individuals in two groups; individuals with the same class labels are exchanged with priority; otherwise, arbitrarily selected individuals are exchanged. Mutation adds or removes an individual from a group. The individual being added is selected from the individual pool (i.e. individuals on the lowest level). The probability of selection, unless specified, is proportional to fitness. Once a new group is created, its fitness is evaluated. The framework requires to check the validation of a group before adding it to the population. This eliminates free riders from groups and prevents group sizes from increasing unnecessarily. For this specific application, the validation is maintained by the last term in group fitness function (see Equation 2). Free riders result in a bigger group without improving data coverage, which diminishes team fitness. Therefore, such groups will not be favored by between-group selection.

Evolution on the individual level Line 9 to 13 are about producing no more than n offspring with new IDs on the individual level. To select a parent individual for reproduction, a group has to be selected first based on fitness, from which an individual is selected with uniform probability. Uniform selection, as opposed to the roulette wheel selection suggested by the framework, is employed in within-group selection, because individuals all contribute to achieve cooperation despite their fitness. Crossover exchanges randomly selected program segments between two parents, while mutation copies, deletes, adds, swaps, and changes instructions in an individual's program with predefined independent probabilities. The fitness of new individuals is evaluated. Individuals are evolved separately from groups, simply because individuals are the most basic building blocks, and only when individuals have fully exploited their local environment, will it be possible to find optimal groups.

Niching Line 14 and 15 conduct niching on individuals and groups. Preserving diversity is mandatory on individual and group levels with the purpose of maintaining all subcomponents in final solutions. A revised fitness sharing is used here: due to the specialty of MCC problems, the similarity of two individuals is measured on their phenotypes; that is, we consider the number of data examples shared between two individuals, as similar individuals will have similar data coverage. To do so, we establish a niche for each individual, when the data overlap between this individual and others exceeds a threshold. If this individual does not have the best fitness in the niche, its fitness has to be reduced by the following equation

$$f_i = f_i * \left(1 - \frac{Avg^i_{overlap}}{TP^i}\right) \quad (3)$$

where $Avg^i_{overlap}$ is the average data overlap between individual i and others in its niche, and TP^i is the true positive of individual i. The advantage of considering this fitness sharing is that it preserves diversity but not at the expense of the best individuals in each niche. Individuals with different class labels have no need to share because they have no overlap. Niching on groups is conducted in a similar manner except that groups only share fitness with others having the same set of class labels. This approach attempts to save the best groups with various granularity in the population.

Survival selection The population size is doubled after the evolution of individuals and groups. The survival selection on Line 16 and 17 reduces the population to its original size by saving the best N individuals and M groups for the next generation.

The above steps will be repeated until a predefined termination criterion is reached, e.g. the maximum generation, a desired fitness or accuracy.

In summary, MLGP uses a bottom-up approach to build

complex solutions from simple subcomponents via the cooperation operator. Group size and the role played by each member, instead of being declared *a priori*, are automatically deduced through evolution. MLGP also satisfies the two criteria of creating successful teams: both group fitness and individual fitness are defined. In addition, individuals and groups are both treated as the objects of evolution; that is to say, both are optimized. This distinguishes MLGP from the other five team-based approaches. The second difference is that MLGP cannot select an individual for reproduction without first selecting a group on fitness. The ability to cooperate, not just fitness, is always considered at individual selection. This important feature also eliminates the credit assignment problem, as members in a cooperative group, despite their individual fitness, will have a higher chance to reproduce.

4. EXPERIMENTAL SETUP

The complexity of classification tasks depends on various data features, such as the separability of classes, dimensionality of the feature space, sparseness of available samples, and the number of classes. To find out how MLGP performs when encountering complexity, we selected seven datasets from UCI data repository [6]: ANN Thyroid Disease (Thyroid), Cleveland Heart Disease (Heart), Statlog Shuttle (Shuttle), Bupa Liver Disorder (Bupa), Pima Indians Diabetes (Pima), Original Breast Cancer Wisconsin (Cancer), and KDD Census Income (Census). The detailed information about the datasets are summarized in Table 1. The first three datasets have at least three classes, while the rest only have two. Shuttle, Thyroid and Census also have imbalanced class distribution, where the data distribution for minor classes is as low as less than 0.01%. This property is ideal to demonstrate how the idea of group selection helps to maintains individuals for both minor and major classes. Bupa and Pima are two datasets known for poor class separability; the rate of error has been observed in the region of 30% across a wide range of machine learning algorithms [10]. Other reasons to select those datasets are that they have established performance levels [9, 10, 15, 18], and they all represent real-world data, rather than artificial data. Therefore, the results obtained on those datasets make us sufficiently confident to judge if the new computational multilevel selection framework is applicable to solve real-world problems.

Additional data pre-processing is performed. We map categorical values into numeric values by their appearing order. Missing values are replaced by the value of the nearest data samples (measured by the euclidean distance) for the relevant attribute.

50 runs were executed on each dataset. 10-fold cross-validation was used to assess datasets denoted by '*' in Table 1; that is five runs per partition. The function set used in the experiments included 7 arithmetic operators: $+$, $-$, $\times$, $\div$, cos, ln, exp, and 1 conditional operator if, which returns the additive inverse of the first operand if it is smaller than the second operand; for example *if* $(R[x] < R[y])$ *then* $(R[x] = -R[x])$. Introns in individual programs were removed by the method described in [2]. 8 registers were used and initialized by the mean value of a randomly selected input attribute. Java parallel programming dispatched individual fitness evaluation and niching to 15 threads running on 15 CPUs.

Parameters shared by all experiments are shown in Table 2, where $ProgramSize_{max}$ refers to the maximum program length, $P_{coopgp}, P_{xovergp}, P_{mutgp}$ for group cooperation, crossover, and mutation probability, $P_{xoveridv}$ for individual crossover, $P_{del}, P_{add}, P_{mut}, P_{swap}, P_{copy}$ for deleting, adding, changing, swapping, and copying instructions at individual mutation, and R_{gp}, R_{idv} for group and individual niching radius. Parameters specific to each dataset were as follows: the maximum generation in runs for Thyroid, and Cancer is 2000, and 10000 for the rest. The population contains 30 individuals and 20 groups for Cancer, 60 individuals and 20 groups for Thyroid, Bupa, Pima and Census, and 140 individuals and 30 groups for Heart and Shuttle.

Table 2: Classification problems parameterization.

Parameters	Value	Parameters	Value
$ProgramSize_{max}$	48	P_{coopgp}	0.5
$P_{xovergp}$	0.8	P_{mutgp}	0.3
$P_{xoveridv}$	0.8	P_{del}, P_{add}	0.6
P_{mut}, P_{swap}	0.6	P_{copy}	1
R_{gp}	0.5	R_{idv}	0.4

Two indicators were used to measure the performance of MLGP on MCC tasks: overall detection rate (ODR) and average detection rate (ADR). ODR is defined as the number of data samples the final solution correctly classified over the total number of test data; ADR is defined as the average detection rate on all classes. ADR is independent of class distribution; hence it is a good supplement to ODR, especially for skewed datasets. Take the Thyroid dataset as an example; only 2% of the test data are from class 1. Even if a final solution missed all data samples in class 1, its ODR can still reach as high as approximately 98%.

5. EVALUATION

This section presents the experiments for highlighting the effects of group selection on MLGP and comparing the performance of MLGP with traditional LGP, one population-based CEA (XCSR), and two team-based CEAs (OET and SBB). The results of the control algorithms, some presented in the format of box plots/violin plots, were gathered from [9, 10, 15, 18]. Violin plots are a combination of a box plot and a kernel density plot which shows the probability density of the data at different values. Box plots allow us to compare two result sets without knowing their underlying statistical distributions. In addition, if the notches of two boxes do not overlap, the median of the two datasets differ at the 0.95 confidence interval. The detection accuracy mentioned in [9, 15, 18], and the multi-class detection rate or the score in [9, 10] are equivalent to ODRs and ADRs in our experiments, respectively.

5.1 Understanding Group Selection

One of the key concepts developed by MLS is to associate the survival of individuals to the performance of their group. It encourages the emergence of cooperative groups, because only through cooperation will individuals seize the chance to reproduce offspring. To find out if this key insight plays the same role in computational settings, we compare MLGP with a control algorithm, called CtrlMLGP, which functions in the same way as MLGP, except parent individuals are

Table 1: Summary of the datasets used in the evaluation. '*' indicates a dataset has no separate test set; therefore we divided the dataset into partitions of 90% for training and 10% for test. The value in parentheses following the name of the dataset indicates the number of features.

Type	Dataset		Data Distribution							
			Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Total
Multi-class	Thyroid (21)	Training	93	191	3488	-	-	-	-	3772
		Test	73	177	3178	-	-	-	-	3428
	Heart (13)*	Training	148	50	33	32	12	-	-	275
		Test	16	5	3	3	1	-	-	28
	Shuttle (9)	Training	34108	37	132	6748	2458	6	11	43500
		Test	11478	13	39	2155	809	4	2	14500
Two-class	Bupa (6)*	Training	181	131	-	-	-	-	-	312
		Test	19	14	-	-	-	-	-	33
	Pima (8)*	Training	451	242	-	-	-	-	-	693
		Test	49	26	-	-	-	-	-	75
	Cancer (10)*	Training	413	217	-	-	-	-	-	630
		Test	45	24	-	-	-	-	-	69
	Census (41)	Training	187141	12382	-	-	-	-	-	199523
		Test	93576	6186	-	-	-	-	-	99762

selected directly from individual pool. That is to say the CtrlMLGP considers no group selection at reproduction.

We run the two algorithms 50 times on the Thyroid dataset, which has 3 classes with imbalanced data distribution. The mean classification accuracies and average class coverage (CLS) are shown in Table 3. The ADR values apparently indicate a performance difference between the two algorithms. The low ADR obtained by CtrlMLGP implies it is not able to cover all classes; on average it covers 2.02 classes out of 3. In fact, CtrlMLGP can hardly include a classifier in groups to cover data examples from class 1, the smallest class. Our further investigations show that individuals evolved for class 1 normally start with a low TPR and a high FPR, i.e. a relatively low individual fitness, because of very scarce training data. Therefore, when competing against individuals who classify data for major classes in the same population, they are less favored given that selection is proportional to fitness. In other words, the growth of fitness happens very slowly on such individuals. As a result, chances of having them in the best group are slim.

Table 3: The average classification accuracies and class coverage of MLGP and CtrlMLGP on Thyroid datasets over 50 runs. Standard deviations are listed inside of parentheses.

	ODR	ADRs	CLS
MLGP	0.978(0.008)	0.954(0.031)	3(0.0)
CtrlMLGP	0.931(0.025)	0.595(0.070)	2.02(0.141)

However, if we allow group selection at reproduction, the outcome is different. Individuals are randomly selected in a group; individuals, despite their fitness, face equal reproduction opportunities. Because such individuals provide new data coverage, their appearance in a group surely improves group fitness, which in turn increases their probability of being selected again. Consequently, the fitness of weak individuals with unique contributions is improved much quicker in a group than in a population. This experiment also demonstrates the importance of individual optimization. Only when individuals are properly explored, will it be possible to build a good solution from them.

5.2 Classification Accuracy

We first evaluate the performance of MLGP on the four two-class datasets. The average classification accuracies and class coverage by the best groups collected from 50 runs are summarized in Table 4. The violin plots of the average ODRs and ADRs are further shown in Figure 1.

Table 4: The average classification accuracies and class coverage of MLGP on four two-class datasets over 50 runs. Standard deviations are listed inside of parentheses.

Dataset	ODR	ADR	CLS
Bupa	0.675(0.063)	0.651(0.072)	2(0.0)
Pima	0.716(0.040)	0.670(0.049)	2(0.0)
Cancer	0.968(0.013)	0.970(0.015)	2(0.0)
Census	0.854(0.019)	0.805(0.016)	2(0.0)

It is evident from Table 4 that the best groups evolved by MLGP successfully covered both classes, even the minority class (class 2) in Census dataset. XCSR, on the contrary, indiscriminately labeled almost all instances in class 2 to class 1, resulting a quite low ADR (around 0.504) [9].

We first compare the ODR box plots obtained by MLGP and traditional LGP on Cancer dataset (See Figure 1 and Figure 10 in [18] for details). Because the minimal ODRs in MLGP are larger than the upper quartile (UQ) value in traditional LGP, it follows that the notches of the two boxes are impossible to overlap. Therefore, we can conclude that MLGP outperforms traditional LGP on this dataset at the 0.95 confidence interval. In the case of the other datasets, we compare ADR box plots produced by MLGP and SBB (box plots of SBB can be found as Figure 4 in [10]). On the Census dataset, MLGP outperforms SBB at the 0.95 confidence interval, given the fact that the two boxes do not overlap. On the Bupa dataset, MLGP and SBB have the same maximum and UQ ADRs, but MLGP has higher minimum, lower quartile (LQ), and median values. On the Puma dataset, MLGP has higher values on all statistics except the maximum value. That is to say, on both datasets SBB's graph is generally lower than MLGP's graph; in addition, the ADRs of SBB have larger variability than MLGP because of a

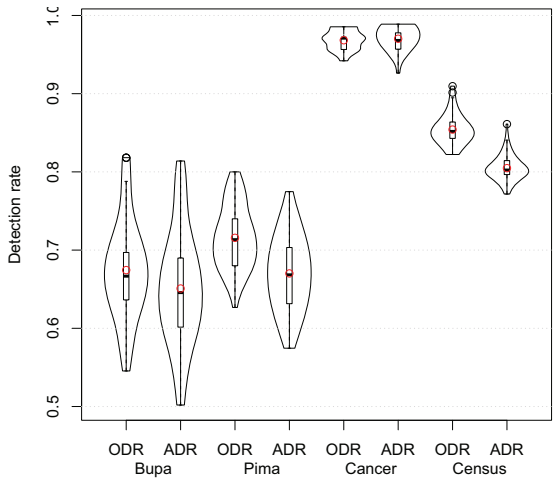

Figure 1: Violin plots of ODRs and ADRs for the four two-class datasets over 50 runs. Each box indicates the lower quartile, median, and upper quartile. The horizontal lines at the end of whiskers represent the maximum/minimum values. Points outside of the boxes represent outliers to whiskers of 1.5 times interquartile range, and points inside of the boxes show the mean values of ODRs or ADRs.

longer interquartile range. Overall, it appears that MLGP performs better and more consistently than SBB. However, because the boxes overlap and the notches are not shown, the significance of the differences cannot be checked.

We then increase the difficulty of the problem by feeding MLGP three more datasets with multiple classes. The results are detailed in Table 5 and plotted in Figure 2.

Table 5: The average classification accuracies and class coverage for the three multi-class datasets over 50 runs. Standard deviations are listed inside of parentheses. Results shown for SBB and XCSR are cited from [9], and OET from [15]. The best values from the three approaches are shown in bold.

Dataset		ODR	ADR	CLS
Thyroid	MLGP	**0.978**(0.009)	**0.950**(0.041)	3(0.0)
	SBB	0.960	0.935	N/A
	XCSR	0.976	0.924	N/A
Shuttle	MLGP	**0.999**(0.001)	**0.983**(0.020)	7(0.0)
	SBB	0.967	0.953	N/A
	XCSR	0.982	0.416	N/A
Heart	MLGP	**0.744**(0.043)	0.688(0.072)	5(0.0)
	OET	0.568(0.030)	N/A	N/A

All three datasets have skewed data distributions, especially Shuttle in which class 6 only has 6, out of total 43500, data examples. However, such skewness apparently affects XCSR most; the low ADR value implies that XSCR is not able to detect data samples of rare classes. In contrast, MLGP and SBB have comparable values on ODRs and ADRs. In fact, the class coverage of MLGP shown in Table 5 clearly indicates that best groups evolved by MLGP identified data from all classes.

As expected, the performance of MLGP on the Heart dataset is better than traditional LGP at the 0.95 confidence interval if we compare their box plots in Figure 2 and Figure 9 from [18]. MLGP also performs better than OET (see Table 5) on this dataset.

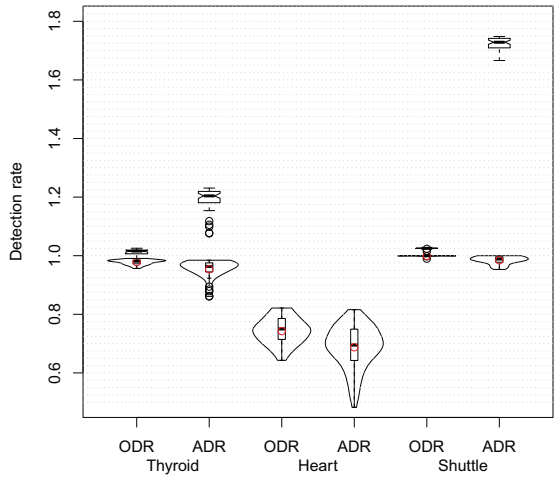

Figure 2: Violin plots of ODRs and ADRs of the best groups for the three multi-class datasets over 50 runs. The box plots depict the distribution of normalized ODRs and ADRs on Thyroid and Shuttle datasets.

As shown in Table 5, MLGP outperforms SBB and XCSR on both datasets with respect to either ODRs or ADRs. To find out if the differences are statistically significant, we then compare their box plots. The box plots of SBB and XCSR (Figure 1 and 2 in [9]) were drawn using the normalized ODR and ADR values. For fair comparison, the same normalization procedure (see [9] for details) was applied to ODR and ADR values in MLGP. Box plots based on the transformed values were depicted in Figure 2. MLGP outperforms SBB and XCSR on Shuttle at the 0.95 confidence interval on both ODR and ADR values. However, with respect to Thyroid dataset, all three ADR box plots have very close maximum values, but again MLGP has the highest median, and the shortest interquartile range; for example the LQ value of MLGP is aligned with the median of SBB and XCSR. The similar patterns are also observed on the ODR box plots. Because the MLGP results are highly clustered, it is difficult to tell whether their box plots overlap or not. However, we can safely conclude that MLGP at least performs as same as SBB and XCSR.

In conclusion, MLGP, in terms of classification accuracies, outperforms SBB on Census and Shuttle at the 0.95 confidence interval, and performs slightly better than or at least as good as SBB on Bupa, Pima, and Thyroid datasets. It excels SBB in consistency on all datasets. The reason that SBB has a diverse distribution over accuracies may partly be due to the uniform probability selection scheme used at within-group and between-group selection. Uniform probability selection does not distinguish individuals and groups based on their performance (fitness). Therefore, optimization opportunities are spread over all individuals and groups. MLGP performs better than XCSR on skewed datasets, such as Thyroid, Shuttle, and Census, because XCSR, as we stated before, lacks a measurement of group performance. MLGP also exceeds single binary classifiers evolved by traditional LGP on performance, because binary classifiers only

focus on one class at a time, and ignore correlations with other classes.

5.3 Solution Complexity

MLGP builds solutions hierarchically out of simple subcomponents without specifying in advance their structure. We are interested to know how complex solutions are, especially when compared to solutions returned by SBB. The solution complexity in this study is measured by the number of members in a team.

Figure 3 plots the average number of individuals in the best groups from 50 runs on the four two-class datasets. The

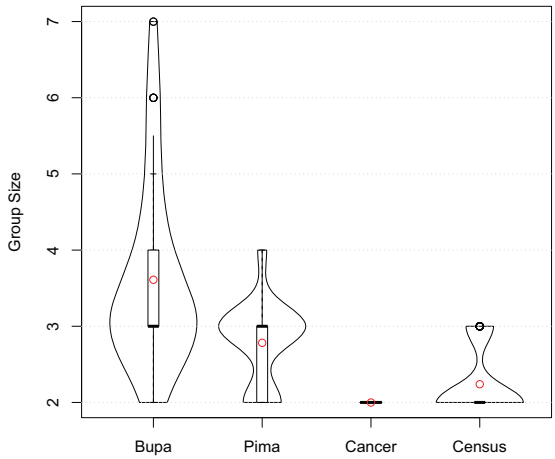

Figure 3: Solution complexity of best groups on the four two-class datasets over 50 runs.

solution complexity of SBB obtained on Bupa, Pima and Census datasets can be found in Figure 8 of [10]. MLGP has the same solution complexity as SBB on Bupa. However, for the other two datasets, SBB tends to find more complicated solutions with larger numbers of individuals than MLGP (at 0.95 confidence interval); for example, the median values obtained by SBB on Pima and Census are 4, while they are 3 and 2 by MLGP, respectively.

On Thyroid and Shuttle datasets, solution complexity of MLGP is significantly smaller than SBB or XCSR (see Table 6). It found the most compact groups on all runs, where only one individual is used to classify every single class.

Table 6: The average solution complexity of MLGP, SBB and XCSR for the three multi-class datasets over 50 runs. Standard deviations are listed inside of parentheses. Results shown for SBB and XCSR are cited from [9]. The best values among the three approaches are shown in bold.

	MLGP	SBB	XCSR
Thyroid	**3**(0.0)	9.5(0.9)	881.2(14.3)
Shuttle	**7**(0.0)	10.0(0)	644.8(39.4)
Heart	6.667(1.361)	N/A	N/A

Therefore, we can conclude that MLGP beats SBB and XCSR in terms of solution complexity. The obvious reason is that MLGP explicitly expresses how to control group size with a group fitness function. Particular interesting is that MLGP automatically keeps the solution complexity in proportion to the separability of the datasets. For highly separable datasets, MLGP returns the smallest group with each member being responsible for one class; However, for poorly separable datasets such as Heart, Bupa and Pima, MLGP tends to evolve large groups in which one class is covered by more than one individual. The results clearly demonstrate the good problem decomposition ability of MLGP; an appropriate number of subcomponents and their roles emerge through evolution without human interference. The driving evolutionary force behind this is the between-level selection, which controls the hierarchical structure by screening out invalid levels and groups.

6. DISCUSSION

So far we have demonstrated how to implement the new computational multilevel selection framework using LGP to solve multi-class classification problems. The experimental findings confirm that MLGP is able to improve solution accuracy and to simplify solution complexity as compared to other approaches in the literature. However, the following issues should be given special consideration before MLGP is applied to new problems:

1. Evolutionary transitions. Even though not mentioned in this paper, MLGP has the potential to be extended to an evolutionary transition model, in which groups, depending on their levels, become a new complex organism functioning differently from their components. The cooperation operator would be responsible for such a transition. Instead of simply combining individuals together, this operator could define more specific transition rules to change the genotype or phenotype of a new organism, thus expressing various functions. This model, we believe, will be useful to solve problems whose subcomponents have more complicated interactions, such as agents in multi-agent systems.

2. Niching. With no exception, the framework requires to use niching or similar techniques to maintain different partial solutions in a population, from which a full solution can be built. Designing an appropriate niching scheme, nevertheless, can be very tricky as it is strongly correlated with the problem. Canonical fitness sharing, resource sharing or crowding is always a good starting point. However, one important thing to remember about fitness sharing is that it reduces fitness of all individuals within niching radius, including the best one in the niche. We are then facing the risk of loosing potentially good individuals; if they are closely surrounded by others, their fitness may degrade much faster than worse individuals with no neighbors.

3. Group fitness definition. After multiple trials on different group fitness definitions, we advise to consider at least two factors: average individual performance and overall group performance. Missing either of them will cause evolution to drift to suboptimal solutions.

4. Cooperation measurement. Evidence in biology and social science suggests that excluding or penalizing free-riders can maintain cooperation. In the same way any implementation of the framework should measure to what degree individuals cooperate in a group. Removing free-riders means compact groups and savings on computing resources. In [19], an individual's contribution is judged by the number of new strings it provided to its group. In this study, the contribution was indirectly assessed in the group fitness function jointly by the group size and overall data coverage; free riders increase group size without improving coverage. The

Shapley value [13] in game theory is also an interesting approach to determine contributions in cooperation.

5. Parameterization. The framework extends evolution to group levels; therefore, one needs to specify values for new parameters namely the cooperation, crossover and mutation rates for reproducing groups, niching radius for groups and individuals, and the number of groups in a population. Like any other EA, there are no universally optimal parameter settings that suit every problem. Based on our experiments, we suggest high cooperation and crossover rates, but a relatively low mutation rate, as cooperation constructs new groups and crossover discovers possible individual combinations. The number of individuals and groups in a population will vary depending on specific problems. Complex problems normally need a large individual pool in order to preserve all potential subcomponents. The group pool is normally smaller than the individual pool, and its size increases as the individual pool grows, but with a smaller rate.

7. CONCLUSION

In this paper, we introduced a multilevel genetic programming (MLGP) system, based on the new computational multilevel selection framework proposed by Wu and Banzhaf [19], to tackle the evolution of cooperation. MLGP, with the help of group selection and a cooperation operator, extends evolution from individuals to multiple group levels. As a result, this process facilitates hierarchically constructed cooperative solutions out of simple ones. The applicability of MLGP is verified on multi-class classification problems, in which 7 benchmark datasets with different data features, such as non-linearity, skewed data distribution, and large feature space were tested. The results show that MLGP, when compared to traditional GP, OET, XCSR and SBB, is able to improve solution accuracy and to simplify solution complexity. In the future, we plan to study the evolutionary transition by MLGP, and its potential applications.

8. ACKNOWLEDGMENTS

W.B. would like to acknowledge support from NSERC Discovery Grants, under RGPIN 283304-07. S.W. would like to thank Dr. Malcolm Heywood for helpful comments and suggestions.

9. REFERENCES

[1] M. E. Borrello. The rise, fall and resurrection of group selection. *Endeavour*, 29(1):43–47, March 2005.

[2] M. Brameier and W. Banzhaf. A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Transactions on Evolutionary Computation*, 5(1):17–26, February 2001.

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

[4] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer. Individual GP: an alternative viewpoint for the resolution of complex problems. In W. Banzhaf, J. Daida, and et al., editors, *Proceedings of the* 1^{st} *Genetic and Evolutionary Computation Conference (GECCO '99), Orlando, Florida, USA*, volume 2, pages 974–981, Orlando, Florida, USA, 1999. Morgan Kaufmann.

[5] S. Forrest, R. E. Smith, B. Javornik, and A. S. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211, 1993.

[6] A. Frank and A. Asuncion. UCI machine learning repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2011.

[7] J. Holland and J. Reitman. Cognitive systems based on adaptive algorithms. In D. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems*. Academic Press, New York, NJ, USA, 1978.

[8] J. Horn and D. E. Goldberg. Natural niching for evolving cooperative classifiers. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Proceedings of the* 1^{st} *Annual Genetic Programming Conference*, pages 553–564. The MIT Press, Cambridge, MA, USA, 1996.

[9] P. Lichodzijewski and M. Heywood. Managing team-based problem solving with symbiotic bid-based genetic programming. In M. Keijzer, editor, *Proceedings of the* 10^{th} *Genetic and Evolutionary Computation Conference (GECCO '08), Atlanta, GA, USA*, pages 363–370, 2008.

[10] P. Lichodzijewski and M. Heywood. Symbiosis, complexification and simplicity under GP. In M. Pelikan and J. Branke, editors, *Proceedings of the* 12^{th} *Genetic and Evolutionary Computation Conference (GECCO '10), Portland, OR, USA*, pages 853–860. ACM, New York, 2010.

[11] G. Olague, E. Dunn, and E. Lutton. Individual evolution as an adaptive strategy for photogrammetric network design. In C. C. et al., editor, *Adaptive and Multilevel Metaheuristics*, pages 157–176. Springer-Verlag Berlin Heidelberg, 2008.

[12] M. A. Potter and K. A. de Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[13] L. S. Shapley. A value for n-person games. In *Contributions to the Theory of Games, volume II*, volume 28 of *Annals of Mathematical Studies*, pages 307–317. Princeton University Press, 1953.

[14] T. Soule and P. Komireddy. Orthogonal evolution of teams: A class of algorithms for evolving teams with inversely correlated errors. In R. L. Riolo, T. Soule, and B. Worzel, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 8, pages 79–95. Springer US, 2006.

[15] R. Thomason and T. Soule. Novel ways of improving cooperation and performance in ensemble classifiers. In H. Lipson, editor, *Proceedings of the* 9^{th} *Genetic and Evolutionary Computation Conference (GECCO '07), London, England, UK*, pages 1708–1715. ACM, 2007.

[16] A. Traulsen and M. A. Nowak. Evolution of cooperation by multilevel selection. In *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, volume 103 (29), pages 10952–10955. National Academy of Sciences, July 18 2006.

[17] D. S. Wilson. A theory of group selection. In *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, volume 72 (1), pages 143–146. National Academy of Sciences, 1975.

[18] G. Wilson and M. Heywood. Introducing probabilistic adaptive mapping developmental genetic programming with redundant mappings. *Genetic Programming and Evolvable Machines*, 8(2):187–220, 2007.

[19] S. X. Wu and W. Banzhaf. A hierarchical cooperative evolutionary algorithm. In M. Pelikan and J. Branke, editors, *Proceedings of the* 12^{th} *Genetic and Evolutionary Computation Conference (GECCO '10), Portland, OR, USA*, pages 233–240. ACM, New York, 2010.

[20] S. X. Wu and W. Banzhaf. Investigations of Wilson's and Traulsen's group selection models in evolutionary computation. In *Proceedings of the* 10^{th} *European Conference on Artificial Life, Budapest, Hungary, 2009*. LNCS, Springer, in press, 2011.