A Multiobjective Genetic Programming based Ensemble for Simultaneous Feature Selection and Classification

Kaustuv Nag and Nikhil R. Pal, *Fellow, IEEE*

***Abstract*—We present an integrated algorithm for simultaneous feature selection and designing of diverse classifiers using a steady state multiobjective genetic programming, which minimizes three objectives: false positives, false negatives, and the number of leaf nodes in the tree. Our method divides a c-class problem into c binary classification problems. It evolves c sets of genetic programs to create c ensembles. During mutation operation, our method exploits the fitness as well as unfitness of features, which dynamically change with generations with a view to using a set of highly relevant features with low redundancy. The classifiers of i^{th} class determine the *net belongingness* of an unknown data point to the i^{th} class using a weighted voting scheme, which makes use of the false positive and false negative mistakes made on the training data. We test our method on eight microarray and eleven text data sets with diverse number of classes (from 2 to 44), large number of features (from 2000 to 49151), and high feature-to-sample ratio (from 1.03 to 273.1). We compare our method with a bi-objective genetic programming scheme that does not use any feature selection and rule size reduction strategy. It depicts the effectiveness of the proposed feature selection and rule size reduction schemes. Furthermore, we compare our method with four classification methods in conjunction with six features selection algorithms and full feature set. Our scheme performs the best for 380 out of 474 combinations of data sets, algorithm and feature selection method.**

***Index Terms*—Classification, ensemble, feature selection, genetic programming.**

I. Introduction

CLASSIFICATION is one of the most important and frequently encountered problems in data mining and machine learning. A wide range of real world problems of different domains can be restated as classification problems. This includes diagnosis from microarray data, text categorization, medical diagnosis, software quality assurance, and many more. The objective of classification is to take an input vector $\boldsymbol{x} = (x_1, ..., x_d)^T$ and to assign it to one of the K classes $\boldsymbol{C_k}$, where $k = 1, ..., K$. A model, called classifier, is used to solve this problem. The classifier encodes a set of criteria. Depending upon some features of the data point, these criteria assign the data point to a particular class [1]. Sometimes, ensembles of weak classifiers are used to obtain better classification accuracy. High dimensionality, high feature-to-sample ratio, and redundant and/or noisy features are some common sources of difficulties associated with classification.

Feature selection (FS) is a process to select a small but useful subset of features from the set of available features which is adequate for solving the problem in an efficient manner. Some available features may be redundant, not useful, and may cause confusion during the learning phase. These unwanted features needlessly increase the size and complexity of the feature space. It increases the computation cost for learning, and may sometimes be responsible for finding suboptimal solutions of the problem. This makes FS techniques important for the analysis of high dimensional data sets, especially when feature-to-sample ratio is extremely high.

The microarray technology has made it possible to diagnose different types of cancers directly using the microarray data sets. One of the main difficulties we face to do this is the high feature-to-sample ratio of microarray data sets, which makes FS an important step. Finding keywords as well as contexts from text data is essential to detect (without human intervention) the context of web pages, emails, or questions/answers etc. A large set of distinct words in large texts (high number of features) and many categories of texts (high number of classes) are the two complicated challenges that we face in this task.

Genetic Programming (GP) [2]–[5] is a biological-evolution inspired methodology where each solution is a program or an equation that evolves with one or more objective functions to perform specific tasks. Many authors [6]–[12] have used GP to design classifiers or to generate rules for binary classification problems. Some researchers have also attempted to solve multi-class problems [13]–[17]. GP has also been used for simultaneous feature selection and classification [18]. In [19], the authors have used GP to create ensembles of classifiers (genetic programs). They have used these ensembles to classify microarray data sets. Though GP is a powerful tool, it has a drawback: without special care each genetic program (equation) becomes huge. As an effect, they don't learn the patterns in the training data. Rather, memorises them. It also makes genetic programs to be difficult to comprehend. Besides, though ensembles can perform better than individual classifiers [20], to obtain better performance, each ensemble should be diverse and each member of the ensemble should be accurate [20]–[22]. Without special care, due to lack of explicit diversity preservation mechanism, the solutions of

Manuscript received August 7, 2014; revised November 25, 2014 and January XX, 2015; accepted January XX, 2015. This paper was recommended by Associate Editor X. XXXXXXXX.

K. Nag is with the Department of Instrumentation and Electronics Engineering, Jadavpur University, Kolkata-700098, India, e-mail: kaustuv.nag@gmail.com.

N. R. Pal is with the Electronics and Communication Sciences Unit (ECSU), Indian Statistical Institute, Calcutta, 700108, India.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier

single objective GP may loose diversity.

The objective of this work is to find an embedded methodology of simultaneous feature selection and classification employing GP as a tool. Some of the novel contributions of our schemes are as follows. We have introduced a new multi-objective genetic programming (MOGP), called ASMiGP. It is enriched by several new operators. The mating selection judicially uses Roulette wheel selection, instead of two tire multiobjective selection scheme (where domination is preferred over diversity). The crossover is new and uses male-female differentiation so that the off-spring is more likely to be close to the female parent in the genotypic space. The mutation is restrictive and performs less exploration in the hypothesis space. Thus, it reduces disruption. For feature nodes, instead of fitness, mutation uses unfitness to select the mutation point. Altering the fitness and unfitness of the features in different stages of learning, we change the objective of searching in the corresponding stages. We use ASMiGP to learn c diverse sets of classifiers (equations) minimizing three objectives: false positive, false negative, and number of leaf nodes of a tree to restrict the rule size. Throughout the learning process, implicit FS is performed using MOGP, whereas several filter based approaches are used in different stages of the procedure. In this way, we obtain concise rules that involve only simple arithmetic operations. A weighted negative voting is then used among the rules of each ensemble. These weights are determined on the basis of the performance of the binary classifiers on the training data set. A new measure of weighted negative voting, called *net belongingness*, is also introduced.

The proposed method has been tested on eight multi-class microarray data sets having large number of features, varying from 2000 to 49151, and high feature-to-sample ratio, varying from 32.26 to 273.06. It has also been tested on eleven high dimensional (varying from 3182 to 26832) text data sets, where the number of classes (categories) vary from 6 to 44. Experimental results reveal that our method can generate ensembles of classifiers with concise rules that can do a good job of classification with a small subset of features.

II. Background and Related Works

This section provides a background to GP, GP-based online feature selection and classification, ensembles, voting schemes, and concise rule finding. This section also includes some related works on these topics.

A. A Concise Introduction to GP

GP [2]–[5] is an approach to find programs that can solve a given problem. GP uses Darwinian principle of "survival of the fittest". It evolves the programs using biologically inspired genetic operations like reproduction, crossover, and mutation in the search space to find such programs which would be able to solve the given problem [15], [23], [24]. The overall search process of traditional single objective GP is quite similar to traditional population based genetic algorithm. The main difference among them is in the representation of solutions. The usual steps of traditional single objective GP can be found in [2], [15], [24].

There exist several ways to encode a program. Probably, the most popular one is to use a tree structure [25]–[27] which we use. In tree-based GP, a program is represented by a tree where the internal nodes are from a set of functions $\mathcal{F}$, and the leaf nodes are from a set of terminals $\mathcal{T}$. The subtrees of a function node are the arguments of that function. The sets $\mathcal{F}$ and $\mathcal{T}$ must satisfy the closure and sufficiency properties [2], [15]. To satisfy the closure property, $\mathcal{F}$ must be well defined and closed for any combination of probable arguments that it may encounter [15]. Again, to satisfy the sufficiency property, $\mathcal{F}$ and $\mathcal{T}$ must be able to represent any possible valid solution of the problem.

B. Feature Selection (FS): Why Embedded?

FS methods are generally divided into two main groups: *filter* method and *wrapper* method [28]. A filter method does not use any feedback from the classifier or any mining algorithm. It relies on the general characteristics of data. On the contrary, to measure the goodness of features, a wrapper method uses a predetermined classifier or mining algorithm, which will finally use the selected features. Consequently, a wrapper method exploits interactions among the subset of features on which it is tested. But, to find an optimal set of features, a wrapper method needs to measure performances on all possible subsets of features. This becomes infeasible for high dimensional data sets. To overcome this problem, wrapper methods typically use a heuristic-based forward or backward selection mechanism [28], which does not evaluate all possible subsets. So, in this work, we consider *embedded* methods, where FS and designing of the classification system are done *together*. Embedded methods do not need to evaluate all possible subsets. Moreover, they can account for interaction between features and the classifier that is used to solve the problem [29]. Usually, embedded methods attain comparable accuracy to wrapper methods as well as comparable efficiency to filter methods. Though for every classification tool it may not be easy to define such an integrated mechanism, several attempts of FS using embedded methods have already been made. These attempts include using single objective GP [18], neural networks [30], [31], and support vector machines [32].

C. GP in Classification and Feature Selection (FS)

Many researchers have used GP as a tool for classification and FS. Some literature on this topic can be found in [1], [33]. It has been used in both filter [34]–[37] and wrapper [38]–[41] approaches. GP has also been used for extracting decision trees [42]–[47]. Among these in [45], [46] MOGP has been used. GP has also been adopted for learning rule based systems [7]–[9], [48]–[51]. Both binary classification [6], [7], [10]–[12], [48] and multi-class [8], [9], [13]–[17], [49]–[51] classification problems have been addressed by GP based (rule based) systems. Even, researchers have applied GP based (rule based) system for binary classification of imbalanced data [52], [53]. Discriminant functions (DFs) based GP for online FS and classification has been adopted in [18] to solve multi-class problems. It is noteworthy that GP has also been used in feature extraction for edge detection [54].

Since, in this work, we have used MOGP for learning DFs, we discuss some relevant GP based systems. In DFs based GP, each program is a mathematical equation, where the variables are features of the data. Usually every program is associated with a class. For every input data point, the program converts it to a single output value. If this output value is more than a predefined threshold (usually zero), the point belongs to the class to which the program is associated with. Thus, a single equation is enough for binary classification problems. For c-class problems, there are two common ways. The first and more frequently used approach is to consider a c-class problem as c binary classification problems [1]. Thus, c number of DFs are used to discriminate c classes. The second and less popular approach is to use a single DF with $(c - 1)$ threshold values to create c intervals. Each of these intervals is then assigned to a particular class. For both categories, a common practice is to evolve a population where each individual encodes either one (for binary classification problems having single threshold, or for multi-class problems having multiple thresholds) or multiple DFs each of which uses a single threshold. Such encoding schemes have been used in [15], [18]. In another practice, each solution encodes multiple DFs having a single threshold. The final output value in this case may be obtained using a (weighted) voting scheme among the output values of each function of the same individual. An example of this scheme can be found in [19].

In the recent years, researchers have made some notable attempts to solve classification problems using MOGP. In [55], the authors have proposed a MOGP to obtain a group of nondominated classifiers, with which the maximum receiver operating characteristic convex hull (ROCCH) can be obtained. To achieve this, they have adopted four different multiobjective frameworks into GP. For further improvement of each individual GP's performance, they have defined two local search strategies that have been especially designed for classification problems. The experimental results in [55] have demonstrated the efficacy of the proposed memetic scheme in their MOGP framework. Another convex hull-based MOGP, called CH-MOGP, can be found in [56]. In [56], the authors have shown the differences between the conventional multi-objective optimization problem and ROCCH maximization problem. They [56] have introduced a convex hull based sorting without redundancy and a new selection procedure which are suitable for ROCCH maximization problem. Again, in [22], the authors have proposed a MOGP based approach that is especially designed for unbalanced data sets. This approach [22] evolves diverse and accurate ensembles of GP classifiers with good performance on both the majority and the minority classes. The individual members of the evolved ensembles, that are composed of nondominated solutions, vote on class membership.

D. GP, Bloating, and Concise Rules

During evolution of GP, variable length genomes gradually start to increase its length without significant improvement in fitness. This incident, called *bloating* [57], is a well known phenomenon in GP. Bloating causes genetic programs (i) to keep reducible genome structures, and (ii) to memorize training data points, rather than recognizing patterns hidden in them. To find rules which are to some extent human interpretable and can be analyzed, each of the genetic programs must be concise. A plausible way to achieve this target is to control bloating. A popular way to handle bloating is to take into account the program size [2], [58]–[61]. Some other methods include spatially-structured populations [62], [63], island based models to introduce spatial structure to the population [62], [63], intron deletion [64], and dynamic population sizing [65], [66]. The work in [67] attempts to understand the GP evolved solutions; while authors in [68] attempt to find comprehensible rules in subgroup discovery.

E. Ensemble Classifier

In classification, the basic task is to search through a hypothesis space to find a suitable hypothesis that will be able to classify the data points in a better way. An ensemble combines multiple hypotheses to form a better one [19], [21], [22], [53], [69], [70]. Empirically, an ensemble performs better when each of the classifiers is highly accurate (strong learner) and the members of the ensemble are significantly diverse. The explanation behind the better performance of ensemble classifiers than a single classifier has been described in [20]. Normally, to decide the class label of a data point, the member classifiers of an ensemble use (weighted) voting. Ensembles are often used in bio-informatics [71].

III. Proposed Work

A. Representation of Classifiers or Solutions

In this work, we evolve c-populations of genetic programs. Each individual of these populations is a binary classifier. Each binary classifier is represented by a single tree. When a data point is passed through an individual of the i^{th} population, if the output value is positive, the individual says that the passed data point belongs to the i^{th} class; otherwise it says that the point does not belong to that class. The internal (non-leaf) nodes of these trees are functions $\mathcal{F}$. The terminal nodes must be either a feature or a constant from the set $\mathcal{C}$. We have imposed some constraint on the minimum size (architecture) of the trees. In each tree there must be at least one function node and two terminal nodes (with at least one feature node). Though, a single feature (terminal) node might be enough to determine the class label, this would rarely happen in practice. The above restrictions make the trees more useful without any loss of generalization capability. Again, after generation of any tree throughout the learning phase (using mutation, crossover, or random initialization), the largest subtree consisting of only constants as its leaf nodes is replaced by a constant leaf node having the equivalent constant value.

B. MOGP for Learning

Larger genetic programs may memorize the training patterns which, in turn, may increase the training accuracy and reduce the understandability of the rules. Therefore, we aim to find smaller but accurate classifiers. Again, when c is high enough, even a balanced c-class data set may get converted to c number of highly imbalanced bi-classification data sets. Instead of minimization of classification error, simultaneous minimization of false positive (FP) and false negative (FN) would be more appropriate in this regard. Suppose a classifier makes some mistakes $m = m_p + m_n$ on a given training data set, where m_p and m_n are respectively the number of FPs and FNs. Consider two classifiers, each making the same number of mistakes m on a given data set. Let for the first classifier $m_p = m_n$ and for the second classifier $m_p >> m_n$ or $m_p << m_n$. If the cost of a FP is the same as that of a FN, we would prefer the first classifier. Consequently, minimization of both FP and FN would be a better strategy than minimization of the total number of misclassification, particularly for imbalanced data sets. Thus, we have three objectives, i.e., minimizations of (a) FP, (b) FN, and (c) rule size. Moreover, when different classes are overlapped, minimization of FP is usually in conflict with the minimization of FN and vice versa. Multiobjective optimization is more suitable when we need to optimize more than one conflicting objective. Therefore, during the learning of each binary classifier, we minimize three objectives using an MOGP: (a) FP, (b) FN, and (c) number of leaf nodes in the tree. The third objective is used to reduce the size of the tree which enhances the understandability and reduces the pattern memorization capability. The algorithm, proposed in this work, is called ASMiGP: Archive based Steady State Micro Genetic Programming, which is presented in Algorithm 1.

Algorithm 1: ASMiGP

```
1  Initialize population using ramped-half-and-half method.
2  Initialize the archive solutions using initial solutions.
3  while $Evaluations^{Current} \leq Evaluations^{Maximum}$ do
4      repeat
5          operator = Select crossover or mutation.
6          if operator == crossover then
7              Select male and female parents (mating
               selection).
8              Perform crossover.
9          end
10         else
11             Select an individual (mating selection).
12             Mutate the individual.
13         end
14     until the infix equation of off-spring is distinct from
       infix equation of any individual of the archive
15     Evaluate the new off-spring.
16     $Evaluations^{Current} = Evaluations^{Current} + 1$
17     Update the archive using new offspring
       (multi-objective environmental selection).
18 end
19 $\mathcal{F}ronts$ = Perform fast-non-dominated-sort.
20 return first front of $\mathcal{F}ronts$.
```

In an evolutionary search, it is desired to have as many generations as possible and steady state nature of an algorithm maximizes the number of generations when the number of function evaluations is fixed [72]. Due to maximization of generations, a steady state evolutionary search is more exploitative to enhance the searching in a region which is more likely to have or closer to the Pareto front and avoiding exploration in regions that are less likely to improve the solutions. It causes faster convergence. In other words, independent of the fitness evaluation process, steady state selection is more performant than discrete generational selection [73]. Therefore, we have used a steady state algorithm instead of a generational one.

C. Feature Selection (FS)

In this work, we have used the *embedded* model of FS. Explicit FS is performed during population initialization and mutation. Implicit FS is performed during crossover. *Filtering* is also performed at three different stages of the learning process; in particular, at the beginning and after 50% and after 75% of evaluations as described next in this subsection.

To facilitate the FS, following [74], we define an index that assesses the discriminating ability of a feature. Consider a two-class problem. Note that for a multi-class problem, the one-versus-all case can also be viewed as a two-class problem. Let there be n_p number of training points and the class label for the j^{th} data point be $+1$ if it belongs to class 1 and -1 if it belongs to class 2. Let the value of the f^{th} feature for the i^{th} data point be $f_i; i = 1, \cdots, n_p$. If the f^{th} feature is a good discriminatory feature then for all data points from class 1, it should have high values and for all points from class 2, it should have low values or vice-versa. Hence for a good discriminatory feature, we can define an ideal feature vector with values 1, if the data point is from class 1 and 0 otherwise; or the feature value is 0, if it is from class 1, otherwise it is 0. Let $\mathbf{S}_f$ be the vector containing the ideal feature values for feature f and s_{fi} be the ideal feature value of the f^{th} feature for the i^{th} sample. Note that, there could other important features that are not linearly related to the class structure. We are not considering them in this preliminary filtering step. As in [74] we compute the Pearson's correlation (or any other measure of similarity) between f and s as a measure of feature relevance:

$$C_f = \sum_{j=1}^{n_p}(s_{ji} - \overline{s})(f_j - \overline{f}) / \sqrt{\sum_{j=1}^{n_p}(s_{ji} - \overline{s})^2 \sum_{j=1}^{n_p}(f_j - \overline{f})^2}. \tag{1}$$

A higher value of $|C_f|$ indicates a stronger discriminative power of feature f. For a multi-class problem, using the one-versus-all strategy, for the i^{th} binary classification problem (class i versus the rest), the correlation for the f^{th} feature is denoted by C_f^i.

Now, we describe the feature selection procedure. Let the set of all features be $\mathcal{F}_{all}$. We intend to incorporate only those features from $\mathcal{F}_{all}$ which are more likely to help the classifiers to decide the class label. We assign different fitness and unfitness measures to the features during the learning phase. To remove a feature node from any tree (during mutation), we select it using Roulette wheel selection on the unfitness values of the features which are present in that tree. Similarly, when a new feature is inserted in the tree, it is selected using Roulette wheel selection on the fitness values. During the first

50% evaluations, the fitness of the features is defined as in equation (2), and the unfitness is defined as in equation (3).

$$F^{0\%,i}_{fitness}(f,i) = \begin{cases} \left(\dfrac{C^i_f}{C^i_{max}}\right)^2, & \text{if } \dfrac{\left|C^i_f\right|}{C^i_{max}} > 0.3 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$$F^{0\%,i}_{unfitness}(f,i) = 1.0 - F^{0\%,i}_{fitness}, \quad (3)$$

where $C^i_{max} = \max\limits_{f\in\mathcal{F}_{all}} \left|C^i_f\right|$. The equation (2) sets the fitness of very poor (poor with respect to its discriminating power) to zero to eliminate their impact during the initial evolution. Let, $\mathcal{F}_{\text{eval}=0\%,i} \subseteq \mathcal{F}_{all}$ be the features with nonzero fitness values. Basically, at this stage we are using a filter on the feature set $\mathcal{F}_{all}$ to obtain a smaller feature set.

After completion of 50% evaluations for each population, we find the features used in that population. Let the feature set be $\mathcal{F}_{\text{eval}=50\%,i}$. Then we make the fitness of all features in $\mathcal{F}_{all} - \mathcal{F}_{\text{eval}=50\%,i}$ to zero. This is done with the assumption that after 50% evaluations useful features have been used by the collection of decision trees. Now the fitness and unfitness values of all features in $\mathcal{F}_{\text{eval}=50\%,i}$ are modified according to equation (4) and equation (5) respectively.

$$F^{50\%,i}_{fitness}(f,i) = \begin{cases} \dfrac{|C^i_f|}{\sum\limits_{\substack{f\neq g \\ g\in\mathcal{F}_{50\%,i}}} |\rho_{fg}|}, & \text{if } f \in \mathcal{F}_{\text{eval}=50\%,i} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$F^{50\%,i}_{unfitness}(f,i) = e^{-\frac{F^{50\%,i}_{fitness}(f,i) - \min_f\{F^{50\%,i}_{fitness}\}}{\max_f\{F^{50\%,i}_{fitness}\} - \min_f\{F^{50\%,i}_{fitness}\}}}, \quad (5)$$

where ρ_{fg} is the Pearson's correlation between f^{th} and g^{th} features. Here we try to select features with higher relevance but reducing the redundancy in the set of selected features. The fitness, defined in equation (4), increases when the feature is highly correlated with class label. Similarly, it reduces when it is more correlated with other existent features. This is done to achieve maximum relevance minimum redundancy (MRMR).

After 75% function evaluations, again we take another snapshot of the population. Let the existent features for the population be $\mathcal{F}_{\text{eval}=75\%,i} \subseteq \mathcal{F}_{\text{eval}=50\%,i}$. Then, the fitness and unfitness values of the features in $\mathcal{F}_{\text{eval}=75\%,i}$ are defined in equation (6) and in equation (7) respectively.

$$F^{75\%,i}_{fitness}(f,i) = \begin{cases} F^{0\%}_{fitness}(f,i), & \text{if } f \in \mathcal{F}_{\text{eval}=75\%,i} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

$$F^{75\%,i}_{unfitness}(f,i) = 1.0 - F^{75\%,i}_{fitness}(f,i) \quad (7)$$

D. Population and Archive Initialization

We initialize each population using ramped-half-and-half method. While constructing the random trees, selection of terminal nodes has been made with a probability p_{var}. To insert a terminal node in a tree, a random number r_n is drawn in $[0,1]$. If $r_n < p_{var}$ then a feature node is added, otherwise a constant node is added. The function nodes are chosen from the set $\mathcal{F}$, with equal probability of inclusion for all functions. The feature nodes are selected using Roulette wheel selection on fitness $F^{0\%}_{fitness}$, defined in equation (2).

To initialize the archive from the initial population, we have used the multi-objective archive initialization scheme present in [72], [75]. It requires two parameters: maximum archive size (N_{max}) and minimum archive size (N_{min}).

E. Selection of Crossover or Mutation

Since ASMiGP is a steady state MOGP, in each generation we generate only one offspring. We use either crossover or mutation to do that. A random number r_c is drawn in $[0,1]$. if $r_c < p_c$ then crossover operator is selected otherwise the mutation operator is selected for that generation.

F. Mating Selection

ASMiGP uses crossover with male and female differentiation which needs one male and one female parent. We perform Roulette wheel selection using classification accuracy of the binary classifiers as fitness to select the female parent. Then the male parent is selected randomly from the remaining archive. The only condition to be satisfied is that the male and female parents must be distinct. For mutation, we need only one individual and it is selected in the same way as done for the female parent in case of crossover operator.

A choice of mating selection could have been the use of some bi-level selection operator, where Pareto dominance is preferred over diversity. In that case, solutions along the whole Pareto optimal solution set with good diversity would have been selected as the primary (female) parents. Note that, we have used crossover with male-female differentiation that tries to generate an off-spring near the primary parent in the hypothesis space. Consequently, this might cause generation of Pareto optimal binary classifiers along the whole Pareto optimal solution set. Though they are Pareto optimal, these binary classifiers may have poor accuracies. This is because of the implicit imbalance nature of the binary classification problems (due to conversion from multi-class classification problems) and different classifier sizes. An ensemble classifier, however, performs better when individual members of the ensemble are more accurate. Therefore, we use classification accuracy based mating selection. It guides the search to generate more accurate binary classifiers.

G. Crossover

In this study, we have used crossover operation with male and female differentiation. We want the off-spring to be near the female (acceptor) parent in the hypothesis space. The male (donor) parent is used to make the off-spring diverse from its mother. To do this, two random points (nodes) are selected from each of the parents. The probabilities of selection of terminal nodes and function nodes as a crossover point (node) are respectively p^c_t and $(1 - p^c_t)$. Then, the subtree rooted at the selected node of the mother tree is replaced with a similarly selected subtree from the father tree. If the off-spring is identical to any of its parents, the whole procedure is repeated (before evaluation/learning of the off-spring).

H. Mutation

In most of the GP based systems, for mutation a subtree rooted at a randomly selected node is replaced by a new randomly generated subtree. Though this kind of mutation explores more in the hypothesis space, it may be too disruptive in nature. Therefore, we intend to use less exploration by keeping the tree structure unaltered. During mutation we perform the following operations on a tree: (1) Each constant of the tree is replaced by another random constant with probability p_c^m. (2) Each function node of the tree is replaced by anther random function node with probability p_f^m. (3) Only one feature node of the tree is replaced by another feature node.

For feature nodes, to select the mutation point Roulette wheel selection is performed on the unfitness of the features which are present in the tree. Similarly, to insert a new feature at the mutation point, we select a feature using Roulette wheel selection based on the fitness values (probability proportional to fitness) of the features.

This restricted mutation scheme ensures that the tree structure of an equation remains the same. It also ensures that the variables in an equation do not change drastically - changing more than one variable would shift the solution (equation) in the hypothesis space by a larger amount.

I. Environmental Selection

ASMiGP uses the archive truncation strategy used in [72], [75]. This multiobjective archive truncation strategy uses Pareto based fitness function, i.e., fitness is Pareto dominance rank. This scheme maintains an archive (ensemble) which has an adaptive and dynamic size. It does not allow the archive to fall below a minimum size ensuring diversity in the genotypic space. Moreover, the environmental selection diminishes the exploration in areas of objective space that are less likely to yield improved solutions [72], ensuring diversity in phenotypic space. Furthermore, we have made the following difference in the environmental selection. ASMiGP ensures that the infix expression of every off-spring (after mutation or crossover) is distinct from every member of the archive. To achieve this, before evaluating the offspring, ASMiGP converts it to its infix expression and then compares the expression with that of each individual of the archive. Only if the expression is unique, the off-spring is evaluated and added to the archive. Otherwise it is discarded and a new off-spring is generated. Another noticeable difference is that here the number of objectives is three. Note that the diversity maintenance in each ensemble both in phenotypic space and in genotypic space finds a diverse set of trees (bi-classifiers). Trees, being diverse, enhance the performance of the corresponding ensemble. In this context, it is worth mentioning that the archive, along with the archive truncation strategy, helps to realize a good Pareto front by explicitly maintaining diversity among the fit (according to rank) solutions. However, the archive alone is not sufficient to evolve a good Pareto front along all the objectives.

J. Decision Making

To determine the class label, we find the *net belongingness* of a point to each class. The *net belongingness* lies in $[0, 1]$. A higher value indicates more *net belongingness* to that class. A data point is assigned to that class for which it has the highest *net belongingness*.

After the learning, we obtain a set of c archives, $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_c\}$; $\forall i, 1 \leq |\mathcal{A}_i| \leq N_{max}$, where c is the number of classes, and $\mathcal{A}_i$ is the set of all binary classifiers for the i^{th} class. To determine the *net belongingness* of a data point $\mathbf{p}$ to class m, $\mathcal{B}_m^{\text{net}}(\mathbf{p})$, it is passed through all genetic programs of set $\mathcal{A}_m$. The *net belongingness*, $\mathcal{B}_m^{\text{net}}(\mathbf{p})$, of the point $\mathbf{p}$ for class m is computed using equation (8).

$$\mathcal{B}_m^{\text{net}}(\mathbf{p}) = \frac{1}{2}\left(\frac{1}{|\mathcal{A}_m|}\sum_{i=1}^{|\mathcal{A}_m|}\mathcal{B}_m^i(\mathbf{p}) + 1.0\right), \quad (8)$$

where $\mathcal{B}_m^i$ is defined as

$$\mathcal{B}_m^i(\mathbf{p}) = \begin{cases} +\left(1.0 - \dfrac{FP_m^i}{FP_m^{max}}\right), & \text{if } \mathcal{A}_m^i(\mathbf{p}) > 0 \\ -\left(1.0 - \dfrac{FN_m^i}{FN_m^{max}}\right), & \text{otherwise.} \end{cases} \quad (9)$$

In equation (9), FP_m^i and FN_m^i respectively represent the number of FPs and FNs made by the i^{th} individual in set $\mathcal{A}_m$ on the training data. FP_m^{max} and FN_m^{max} are respectively the maximum possible FP and the maximum possible FN for the m^{th} class; and $\mathcal{A}_m^i(\mathbf{p})$ is the output from i^{th} individual of $\mathcal{A}_m$ for input data point $\mathbf{p}$. Finally, $\mathbf{p}$ is assigned the class k, when $\mathcal{B}_k^{\text{net}} = \max_{m=1}^{c}\{\mathcal{B}_m^{\text{net}}\}$. Note that FP_m^{max} and FP_m^{min} are determined by the training data.

The concept of *net belongingness* is inspired by the concept of negative voting. Negative voting has been widely used in diverse applications [76]–[78]. It is more effective when circumstances unfavourable to the preferences invoke stronger electoral responses than the similar favourable responses, as well as the behaviours of the voters are well defined [79]. In our scheme, the learners for the i^{th} class learn to vote *yes* for the points of the i^{th} class and *no* for the points which do not belong to the i^{th} class. For multi-class problems, it is more likely that a binary classifier learns to say *no* than to say *yes* for a much higher number of points. Therefore, we found negative voting to be more suitable in this context. However, we have used a weighted negative voting scheme. The accuracies for the responses *yes* and *no* of the i^{th} binary classifier of m^{th} class are respectively $\left(1.0 - \dfrac{FP_m^i}{FP_m^{max}}\right)$ and $\left(1.0 - \dfrac{FN_m^i}{FN_m^{max}}\right)$. These values have been used as corresponding weights for the responses *yes* and *no* of the i^{th} binary classifier of the m^{th} class. We have used the positive and the negative signs to indicate acceptance and rejection of the data point for the m^{th} class respectively.

IV. Experimentation

A. Experimental Settings

We have repeated 10-fold cross validation of the proposed method for 10 times. Table I shows the parameter settings that we have used for this purpose. The training data is Z-score normalized. Furthermore, based on the means and the

TABLE I
PARAMETER SETTINGS FOR THE PROPOSED METHOD

Parameter	Value
Set of functions ($\mathcal{F}$)	$\{+,-,\times,\div\}$
Range of initial values of constants ($\mathcal{C}$)	[0, 2]
Maximum depth of tree during initialization	6
Maximum allowable depth of tree	10
Maximum archive size (N_{max})	50
Minimum archive size (N_{min})	30
Initial Probability of feature nodes (p_{var})	0.8
Probability of crossover (p_c)	0.8
Probability of crossover for terminal nodes (p_t^c)	0.2
Probability of mutation for constants (p_c^m)	0.3
Probability of mutation for function nodes (p_f^m)	0.1
Function evaluations for each binary classifier	400000

TABLE II
SUMMARY OF MICROARRAY DATA SETS

	Data Set	Features (F)	Samples (S)	Classes (C)	$\left(\frac{F}{S}\right)$
1	Colon	2000	62	2	32.26
2	TOX-171	5748	171	4	33.61
3	Leukemia 1	7129	34	2	209.68
4	Leukemia 2	7129	38	2	187.61
5	CLL-SUB-111	11340	111	3	102.16
6	GCM	16063	144	14	111.55
7	SMK-CAN-187	19993	187	2	106.91
8	GLA-BRA-180	49151	180	4	273.06

standard deviations of the features of the training data, the test data was Z-score normalized. Note that except N_{max} and N_{min}, all parameters are standard parameters used in any GP simulations, while N_{max} and N_{min} are needed for archive maintenance. Although all the parameters can be chosen using a cross validation mechanism, because of huge computational overhead, we could not do that. Based on a few experiments we selected these parameters. These choices are not optimal. However, since the same set of values are used for widely different types of data sets, it demonstrates the effectiveness of the proposed scheme. The proposed method has been implemented in Java with the help of jMetal [80], [81].

We have used eight microarray and eleven text data sets which are summarized in Table II and in Table III, respectively.

TABLE III
SUMMARY OF TEXT DATA SETS

	Data Set	Features (F)	Samples (S)	Classes (C)	$\left(\frac{F}{S}\right)$
1	oh0.wc	3182	1003	10	3.17
2	oh10.wc	3238	1050	10	3.08
3	tr12.wc	5804	313	8	18.54
4	tr23.wc	5832	204	6	28.59
5	tr11.wc	6429	414	9	15.53
6	tr21.wc	7902	336	6	23.52
7	wap.wc	8460	1560	20	5.42
8	ohscal.wc	11465	11162	10	1.03
9	la2s.wc	12432	3075	6	4.04
10	la1s.wc	13195	3204	6	4.12
11	new3s.wc	26832	9558	44	2.81

B. Importance of FS and Rule Size Reduction

To demonstrate the importance of the proposed FS and rule size reduction scheme, we have compared our method with GP without FS and no restriction to equation size. To do that, we have made the following changes in the proposed method: (1) No explicit feature selection as described in Section III-C is done. (2) There are only two objectives, FP and FN. All other parts of the algorithm and the parameter values remain unchanged. Similar to the proposed scheme, we have also executed 10-fold cross validation for 10 times for all the data sets. Along with results (mean values of the corresponding 10 runs) obtained using the proposed method, the results obtained using this scheme for microarray and for text data sets are respectively summarized in Table IV and in Table V. From these two tables, we observe the following.

1) In all cases, the test accuracy is much higher for the proposed method. Especially, for GCM having 14 classes, the difference between the test accuracies are remarkably high.
2) The tree size increases significantly when the third objective, i.e., the restriction on rule size, is not used.
3) For most of the data sets the proposed method selects smaller number of distinct features per tree as well as for per classifier.

These observations clearly demonstrate the importance of the FS as well as constraining the rule size. Since our FS scheme discards features with poor relevance and uses features with good discriminating power yet avoiding use of redundant features, it not only makes the discovery of useful rules easier but also implicitly constrains the rule length. Thus, FS plays a very important role having two positive impacts: it makes identification of useful rules easier and it promotes the minimization of the third objective.

C. Comparing with Other Methods

To compare the performance of the proposed method we have used the experimental results reported in [82] (see Table 4, Table 5, Table 6, and Table 7 of [82]). In [82] the authors have used four different types of classification algorithms: (1) probability based Naive Bayes (NB), (2) tree based C4.5, (3) instance based lazy learning algorithm IB1, and (4) rule based RIPPER both before and after feature selection. Along with the full feature set, they have used six feature selection algorithms in their experiment: (1) FAST [82], (2) FCBF [83], [84], (3) CFS [85], (4) ReliefF [86], (5) Consist [87], and (6) FOCUS-SF [88]. Use of all features can be viewed as the seventh feature selection algorithm. To make this paper comprehensive we are not discussing the experimental settings used in [82]. Note that accuracies for few data sets for few pairs of feature selection schemes are not available in [82].

1) Results with Microarray Data Sets: Table IV presents the results of the proposed method on microarray data sets. We have already stated that for each classifier in [82] the authors have used seven FS schemes (six feature selection method as well as the set of all features). For each feature selection method five repetitions of the 10-fold cross validation experiment were done in [82]. And then, for each feature

TABLE IV
EXPERIMENTAL RESULTS ON MICROARRAY DATA SETS (MEAN VALUES OF 10 RUNS OF 10-FOLD CROSS VALIDATION)

Data Set	%TA[a]		FS[b]		TS[c]		(F/T)[d]		%F[e]		$(\%F/T)$[f]	
	PM[g]	$W^h_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$
Colon	85.12 (2.07)	71.93 (4.12)	195.0	182.5	6.76	123.60	3.11	8.56	9.75	9.13	0.16	0.43
TOX-171	81.07 (3.42)	58.52 (4.57)	407.6	540.9	8.83	148.44	3.53	11.97	7.09	9.41	0.06	0.21
Leukemia1	93.50 (2.85)	74.08 (6.42)	188.6	276.6	3.01	76.09	2.00	6.52	2.65	3.88	0.03	0.09
Leukemia2	91.58 (3.67)	77.33 (6.84)	147.8	229.0	3.50	74.16	2.02	5.72	2.07	3.21	0.03	0.08
CLL-SUB-111	80.52 (2.60)	54.51 (3.51)	335.3	465.3	7.29	134.28	3.20	10.75	2.96	4.10	0.03	0.09
GCM	69.35 (1.96)	33.97 (2.60)	854.3	1086.0	4.70	105.42	1.91	6.22	5.32	6.76	0.01	0.04
SMK-CAN-187	68.68 (1.18)	62.12 (2.30)	319.9	435.8	22.23	135.88	6.41	15.33	1.60	2.18	0.03	0.08
GLA-BRA-180	68.22 (2.25)	58.89 (2.75)	784.4	569.9	11.30	150.81	4.33	11.40	1.60	1.16	0.01	0.02

[a]Test Accuracy (standard deviation is provided within parenthesis), [b]Number of Features Selected per Classifier, [c]Tree Size, [d]Number of Features per Tree, [e]Percentage of Features Selected, [f]Percentage of Features Selected per Tree, [g]Proposed Method, [h]GP without FS and Rule Size Reduction

TABLE V
EXPERIMENTAL RESULTS ON TEXT DATA SETS (MEAN VALUES OF 10 RUNS OF 10-FOLD CROSS VALIDATION)

Data Set	%TA[a]		FS[b]		TS[c]		(F/T)[d]		%F[e]		$(\%F/T)$[f]	
	PM[g]	$W^h_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$	PM	$W_{F\&R}$
oh0.wc	87.75 (0.51)	70.53 (1.17)	484.1	2080.7	17.06	145.38	6.46	25.90	15.21	65.39	0.20	0.81
oh10.wc	81.06 (0.61)	66.11 (1.52)	439.6	2166.8	24.11	140.67	7.12	26.03	13.58	66.92	0.22	0.80
tr12.wc	87.86 (1.50)	60.55 (2.36)	603.7	1697.3	8.67	123.91	4.17	17.22	10.40	29.24	0.07	0.30
tr23.wc	93.95 (1.19)	64.90 (2.14)	396.5	1154.2	6.10	117.15	3.09	14.81	6.80	19.79	0.05	0.25
tr11.wc	86.08 (0.72)	64.63 (2.56)	631.6	1785.8	8.82	128.57	4.03	17.03	9.82	27.78	0.06	0.26
tr21.wc	89.64 (1.14)	71.46 (2.62)	362.0	1219.2	9.51	114.51	4.17	15.00	4.58	15.43	0.05	0.19
wap.wc	79.60 (0.60)	58.96 (1.64)	1199.8	4377.5	17.38	125.53	6.07	21.02	14.18	51.74	0.07	0.25
ohscal.wc	73.45 (0.23)	62.59 (1.21)	232.9	3853.8	63.48	111.52	7.78	24.87	2.03	33.61	0.07	0.22
la2s.wc	83.95 (0.62)	67.32 (0.99)	354.6	2369.1	53.17	133.76	11.41	28.65	2.85	19.06	0.09	0.23
la1s.wc	83.06 (0.36)	65.62 (1.05)	380.6	2372.0	56.40	133.21	12.30	28.35	2.88	17.98	0.09	0.21
new3s.wc	81.32 (0.27)	58.96 (1.64)	1674.2	4377.5	33.93	125.53	7.10	21.02	6.24	16.31	0.03	0.08

[a]Test Accuracy (standard deviation is provided within parenthesis), [b]Number of Features Selected per Classifier, [c]Tree Size, [d]Number of Features per Tree, [e]Percentage of Features Selected, [f]Percentage of Features Selected per Tree, [g]Proposed Method, [h]GP without FS and Rule Size Reduction

selection method, the average accuracy over the five repetitions is reported. We compare this average accuracy with the *average* accuracy that we have obtained by our method over the 10 repetitions of the 10-fold cross validation experiments. In particular, we count the number of cases (each case refers to one feature selection scheme) in which our algorithm outperforms. Note that, for some combination of data set and classifier, the total number of cases is less than seven as for those combinations some results are not available. Table VI reports these counts. To elaborate Table VI, consider the entry corresponding to column IB1 and row CLL-SUB-111. For the data set CLL-SUB-111, in Table 6 of [82] authors report the performance of the algorithm IB1 for six different feature selection algorithms. Our algorithm is found to perform better than five of the six feature selection algorithms and hence the entry for row CLL-SUB-111 and column IB1 is $5/6$. Note that, for this data set, authors of [82] did not report performance of IB1 using all features. All but the entries in the last column of Table VI are generated in the same manner. The last column of Table VI, which reports the row total, reveals that our method performs the best for 61.40% (132 out of 215) test cases.

If we compute the percentage of features selected by the ensembles, it may not be very small for some data sets, like Colon. But, if we consider the number of features selected per binary classifier (tree), we find that this number is quite small, e.g., the maximum value is 0.16% for Colon. We assume that binary classifiers (binary trees) having less than twenty nodes are concise enough. We need to remember that we are talking

TABLE VI
COMPARISON WITH NB, C4.5, IB1, AND RIPPER ON MICROARRAY DATA SETS

Data Set	NB	C4.5	IB1	RIPPER	Total
Colon	4/7	2/7	4/7	6/7	16/28
TOX-171	4/7	7/7	3/7	7/7	21/28
Leukemia1	2/7	2/7	2/7	2/7	8/28
Leukemia2	5/7	6/7	6/7	5/7	22/28
CLL-SUB-111	4/7	6/7	5/6	4/7	19/27
GCM	4/7	7/7	6/7	7/7	24/28
SMK-CAN-187	2/6	2/6	2/6	2/6	8/24
GLA-BRA-180	3/6	5/6	1/6	5/6	14/24
Total	28/54	37/54	29/53	38/54	132/215

about raw rules (equations) directly obtained from GP which are most likely affected by bloating. Simplification of these rules may lead to reduction in their sizes. Based on this, in all but one data set we could find easy to analyze rules. For SMK-CAN-187 the extracted rules are more complex possibly because of complex structure of the data.

In Table S-I (see supplementary materials), we present the best rules (or binary classifiers or GPs) we found for each class of each microarray data set. These GPs have the highest training accuracy among the GPs obtained from the first fold of the first 10-fold cross validation. If there are more than one rule having the same maximum training accuracy, the rule with the smallest length has been reported. The features in the equations are indexed starting from '0'. From Table S-I we can observe that for several classes the proposed method could find substantially small rules. Noticeably, for four, six, eight,

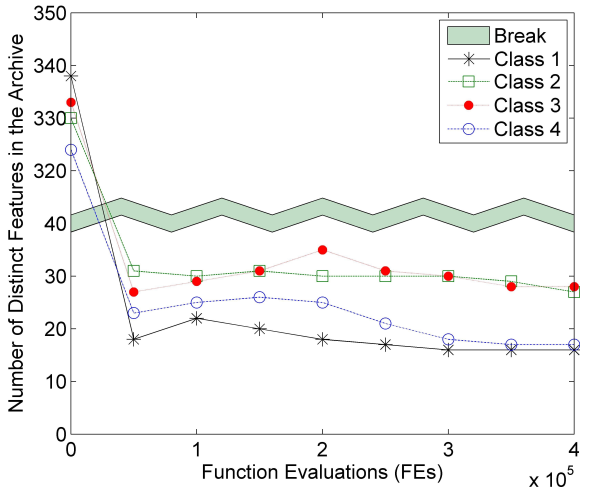

Fig. 1. Showing the number of distinct features present in the archive with respect to function evaluations for GLA-BRA-180 data set.

and three classes, the proposed method could find the best binary classifier having only one, two, three, and four distinct variables, respectively. So, 21 out of 33 (i.e., 63.64%) cases, we could find considerably small (and hence easy to interpret) rules. For some classes, the rules having the highest training accuracy were not very comprehensible because the length of the rule was not very small. If we think that $(2.0 \times x_i)$ is more easy to understand than $(x_i + x_i)$, though both equations have the same *size*, we observe that for 21 out of 33 (i.e., 63.64%) cases the proposed method could find rules having highest accuracy, which are not at all affected by bloating and are comprehensible without simplification.

To investigate how the number of distinct features changes in the archive, we have executed our algorithm with 400000 function evaluations using the entire GLA-BRA-180 data set. In Fig. 1 we have plotted the number of distinct features after initial and after each 12.5% of maximum function evaluations. For all four classes of GLA-BRA-180 data set, the number of distinct features in the archive initially falls drastically depicting the impact of feature selection. After some generations, it becomes almost constant. Moreover, the number of final distinct features for each class is quite small. This indeed reveals a desirable behavior of the our scheme.

To show how the individual binary classifiers' accuracies change with function evolutions, we have plotted each individuals' FPs and FNs for all four classes of GLA-BRA-180 data set in Fig. S-1 and in Fig. S-2 (see supplementary materials). For this part of the experiment also, we have used the entire data set for training. From the figures it is observed that with the increase in number of evolutions, the average accuracy of the solutions tends to increase. However, for class four, some solutions having lower level of accuracies (having higher FPs and FNs) are there even after 400000 function evaluations. This is because, even after such high number of function evaluations, the algorithm was still searching for more concise solutions, and could find some trees of comparatively smaller size.

TABLE VII
COMPARISON WITH NB, C4.5, IB1, AND RIPPER ON TEXT DATA SETS

Data Set	NB	C4.5	IB1	RIPPER	Total
oh0.wc	7/7	7/7	7/7	6/6	27/27
oh10.wc	7/7	7/7	7/7	6/6	27/27
tr12.wc	7/7	7/7	7/7	7/7	28/28
tr23.wc	7/7	5/7	7/7	4/7	23/28
tr11.wc	7/7	7/7	7/7	6/6	27/27
tr21.wc	6/7	6/7	6/7	3/6	21/27
wap.wc	7/7	6/6	6/6	6/6	25/25
ohscal.wc	4/4	4/4	4/4	4/4	16/16
la2s.wc	6/6	5/5	5/5	5/5	21/21
la1s.wc	6/6	5/5	5/5	5/5	21/21
new3s.wc	3/3	3/3	3/3	3/3	12/12
Total	67/68	62/65	64/65	55/61	248/259

2) Results with Text Data Sets: Similar to Table VI with microarray data sets, we present the same result (number of test cases for which our algorithm provides the best results for each classifier and data set pair) with text data sets in Table VII. Table VII reveals that for text data sets the proposed scheme performs the best for 95.75% (248 out of 259) cases.

If we consider the same criteria that binary classifiers (trees) having less than twenty nodes are concise enough, then our method could not find interpretable rules for five (oh10.wc, ohoscal.wc, la2s.wc, la1s.wc, and new3s.wc) out of the eleven text data sets. The number of selected features is not at all small for most of the text data sets. Some reasons behind this may be that the number of classes is high for the text data sets and the existence of more complex class structure, which is defined by the keywords and relation of keywords to the imposed classes is usually not as straightforward as genes have to cancers. However, the percentage of features selected per binary classifiers (trees) is quite small. Noticeably, the accuracy obtained for new3s.wc (having 44 classes), is much higher than that of the other methods (for accuracies of other methods, see [82]). We can also observe that for data sets having more than or equal to ten classes, our methods performs comparatively better than other methods.

D. Statistical Significance Testing

To compare the proposed method with existing feature selection and classification methods, we consider four classification and six feature selection methods as well as with the full feature set. Thus we compare our algorithm with 28 feature selection and classification algorithm pairs. We have performed Wilcoxon signed ranks test to show that the performance of the proposed algorithm is significantly different over 28 pairs of feature selection and classification algorithm. For this, we have considered both the microarray and the text data sets together and have used the average test accuracies achieved with different algorithms on these data sets. Moreover, we have removed the cases for which accuracies are not known (see Table 4, Table 5, Table 6, and Table 7 of [82]). Table VIII shows that out of the 28 cases, only in one case the null hypothesis H_0 was accepted. Here the null hypothesis is that there is no significant difference in performance between our algorithm and a comparing algorithm. We have also used Friedman test [89], [90] to check if all the 29 (28 existing

TABLE VIII
WILCOXON SIGNED RANKS TEST (1-TAILED) FOR PAIRWISE COMPARISONS WITH THE PROPOSED METHOD AT $\alpha = 0.05$

	NB	C4.5	IB1	RIPPER
FAST	A	R	R	R
FCBF	R	R	R	R
CFS	R	R	R	R
ReliefF	R	R	R	R
Consist	R	R	R	R
FOCUS-SF	R	R	R	R
Full Set	R	R	R	R

A: H_0 Accepted, R: H_0 Rejected.

and our proposed) algorithms perform similarly over seven data sets: Colon, TOX-171, Leukemia1, Leukemia2, GCM, tr12.wc, and tr23.wc (see Table 4, Table 5, Table 6, and Table 7 of [82], some accuracies for other data sets are not available). The obtained Friedman statistics is 60.13, which is significant at $\alpha = 0.001$, as the corresponding statistic χ^2 with 28 degrees of freedom is 56.90. Thus Friedman test suggests existence of significant difference among the algorithms.

V. CONCLUSIONS AND DISCUSSIONS

In this work we have used a multiobjective genetic programming, called ASMiGP, to evolve diverse sets of binary classifiers to solve multi-class classification problems. Simultaneous feature selection and rule extraction are performed during the genetic evolution. An important objective of this work is to find simple classification rules that may be human understandable, which in the given context translates to rules with simple operations and short length. The proposed method is found to achieve its goal.

Ensembles perform better when weighted voting is used, the members of the ensembles are diverse enough, and the classifiers are accurate [21]. Here we have created c sets of diverse ensembles. Unlike other ensemble based method, here each ensemble represents diverse classifiers for just one class. The number of features used by the ensembles is high with respect to the number of features selected per (binary classifier) tree. This suggests that the binary classifiers are diverse enough. In our algorithm, we evolve c distinct species in parallel, which try to learn distinct patterns and no inter-species gene exchange is ever allowed. This property makes each species different from the other species and the system becomes less vulnerable, especially when each species has successfully learnt its designated patterns.

Our method has been tested on nineteen (eight microarray and eleven text) data sets. The experimental results show that we could find easy-to-understand rules for overall 63.2% (87.5% for microarray and 54.5% for text) data sets. We compared our method with four classification methods in conjunction with seven feature selection methods including use of the all-feature set. For 80.17% (61.40% for microarray and 95.75% for text) cases our method outperforms others. The improvement in performance is shown statistically significant compared to the others.

The overall performance of the proposed method is better on text data sets compared to that of microarray data sets. Two important differences between these two groups of data sets are that (i) microarray data sets have comparatively large feature-to-sample ratios and may not have enough number of points in each class for MOGP to learn the structure of the data sets, and (ii) the text data sets have comparatively large number of classes as well as instances. Our limited experiments suggest that when there are enough points in each class so that each species can successfully learn its target pattern, the proposed method works better. We found that for text data our method performs noticeably better than the other methods particularly when number of classes is high (ten or more than that).

We have shown that our method is very effective when the dimension of the data sets are as high as 49151. But in all our data sets, number of samples were not very big. If we apply the proposed method on a data set with a really large number samples, it may require a significant amount of time. The method may also take a substantial amount of time when the number of classes is high and we have limited parallel processing capability. With the today's high performance computing technologies, however, these are not really crucial shortcomings. Yet, we have not applied it to truly big data. This work can be adapted to deal with big data, especially using Hadoop.

Our future research interest is to use MOGP to classify multi-class text data, where each text may belong to more than one category (class) and to use *net belongingness* as a fuzzy membership of the documents to the corresponding category. We also intend to modify the proposed learning method to stepwise learning, so that we can use it for big text data.

ACKNOWLEDGMENT

Authors gratefully acknowledge the computing support obtained from the Networked Communications Program of Chunghwa Telecom Co, Taiwan. Kaustuv Nag is thankful to the Department of Science and Technology, India for providing support in the form of INSPIRE Fellowship (code no. IF120686).

REFERENCES

[1] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 2, pp. 121–144, 2010.
[2] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992.
[3] ——, *Genetic programming II: automatic discovery of reusable programs*. MIT press, 1994.
[4] ——, "Genetic programming as a means for programming computers by natural selection," *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, 1994.
[5] J. R. Koza, F. H. Bennett III, and O. Stiffelman, *Genetic programming as a Darwinian invention machine*. Springer, 1999.
[6] P. J. Rauss, J. M. Daida, and S. Chaudhary, "Classification of spectral imagery using genetic programming," *Ann Arbor*, vol. 1001, p. 48109, 2000.
[7] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Evolutionary Programming VII*. Springer, 1998, pp. 735–744.
[8] I. De Falco, A. Della Cioppa, and E. Tarantino, "Discovering interesting classification rules with genetic programming," *Applied Soft Computing*, vol. 1, no. 4, pp. 257–269, 2002.
[9] C. C. Bojarczuk, H. S. Lopes, and A. A. Freitas, "Genetic programming for knowledge discovery in chest-pain diagnosis," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 19, no. 4, pp. 38–44, 2000.

[10] H. Gray, R. Maxwell, I. Martinez-Perez, C. Arus, and S. Cerdan, "Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra," *Genetic Programming*, p. 424, 1996.

[11] D. Hope, E. Munday, and S. Smith, "Evolutionary algorithms in the classification of mammograms," in *Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on*. IEEE, 2007, pp. 258–265.

[12] G. Wilson and M. Heywood, "Introducing probabilistic adaptive mapping developmental genetic programming with redundant mappings," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 187–220, 2007.

[13] J. Kishore, L. M. Patnaik, V. Mani, and V. Agrawal, "Application of genetic programming for multicategory pattern classification," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 3, pp. 242–258, 2000.

[14] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Designing a classifier by a layered multi-population genetic programming approach," *Pattern recognition*, vol. 40, no. 8, pp. 2211–2225, 2007.

[15] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 183–196, 2004.

[16] M. Zhang and P. Wong, "Genetic programming for medical classification: a program simplification approach," *Genetic Programming and Evolvable Machines*, vol. 9, no. 3, pp. 229–255, 2008.

[17] M. Zhang and W. Smart, "Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification," *Pattern Recognition Letters*, vol. 27, no. 11, pp. 1266–1274, 2006.

[18] D. P. Muni, N. R. Pal, and J. Das, "Genetic programming for simultaneous feature selection and classifier design," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 1, pp. 106–117, 2006.

[19] K.-H. Liu and C.-G. Xu, "A genetic programming-based approach to the classification of multiclass microarray datasets," *Bioinformatics*, vol. 25, no. 3, pp. 331–337, 2009.

[20] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*. Springer, 2000, pp. 1–15.

[21] L. K. Hansen and P. Salamon, "Neural network ensembles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 10, pp. 993–1001, 1990.

[22] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 3, pp. 368–386, 2013.

[23] W. B. Langdon and R. Poli, *Foundations of genetic programming*. Springer, 2002.

[24] K. Neshatian, M. Zhang, and P. Andreae, "A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming," *IEEE transactions on evolutionary computation*, vol. 16, no. 5, pp. 645–661, 2012.

[25] P. Nordin, "A compiling genetic programming system that directly manipulates the machine code," *Advances in genetic programming*, vol. 1, pp. 311–331, 1994.

[26] M. O'Neill and C. Ryan, "Grammatical evolution," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 4, pp. 349–358, 2001.

[27] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Genetic Programming*. Springer, 2000, pp. 121–132.

[28] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[29] Y.-C. Chen, N. R. Pal, and I.-F. Chung, "An integrated mechanism for feature selection and fuzzy rule extraction for classification," *Fuzzy Systems, IEEE Transactions on*, vol. 20, no. 4, pp. 683–698, 2012.

[30] N. Pal and K. Chintalapudi, "A connectionist system for feature selection," *Neural Parallel and Scientific Computations*, vol. 5, pp. 359–382, 1997.

[31] D. Chakraborty and N. R. Pal, "Selecting useful groups of features in a connectionist framework," *Neural Networks, IEEE Transactions on*, vol. 19, no. 3, pp. 381–396, 2008.

[32] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[33] H. Jabeen and A. R. Baig, "Review of classification using genetic programming," *International journal of engineering science and technology*, vol. 2, no. 2, pp. 94–103, 2010.

[34] M. Muharram and G. D. Smith, "Evolutionary constructive induction," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 11, pp. 1518–1528, 2005.

[35] K. Neshatian and M. Zhang, "Genetic programming and class-wise orthogonal transformation for dimension reduction in classification problems," in *Genetic Programming*. Springer, 2008, pp. 242–253.

[36] H. Guo and A. K. Nandi, "Breast cancer diagnosis using genetic programming generated feature," *Pattern Recognition*, vol. 39, no. 5, pp. 980–987, 2006.

[37] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 1, pp. 89–99, 2005.

[38] K. Krawiec, "Genetic programming-based construction of features for machine learning and knowledge discovery tasks," *Genetic Programming and Evolvable Machines*, vol. 3, no. 4, pp. 329–343, 2002.

[39] X. Tan, B. Bhanu, and Y. Lin, "Fingerprint classification based on learned features," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 35, no. 3, pp. 287–300, 2005.

[40] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, 2005.

[41] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 35, no. 2, pp. 156–171, 2005.

[42] J. R. Koza, "Concept formation and decision tree induction using the genetic programming paradigm," in *Parallel Problem Solving from Nature*. Springer, 1991, pp. 124–128.

[43] G. Folino, C. Pizzuti, and G. Spezzano, "Genetic programming and simulated annealing: A hybrid method to evolve decision trees," in *Genetic Programming*. Springer, 2000, pp. 294–303.

[44] J. Eggermont, "Evolving fuzzy decision trees with genetic programming and clustering," in *Genetic Programming*. Springer, 2002, pp. 71–82.

[45] H. Zhao, "A multi-objective genetic programming approach to developing pareto optimal decision trees," *Decision Support Systems*, vol. 43, no. 3, pp. 809–826, 2007.

[46] T. M. Khoshgoftaar and Y. Liu, "A multi-objective software quality classification model using genetic programming," *Reliability, IEEE Transactions on*, vol. 56, no. 2, pp. 237–245, 2007.

[47] A. Tsakonas, "A comparison of classification accuracy of four genetic programming-evolved intelligent structures," *Information Sciences*, vol. 176, no. 6, pp. 691–724, 2006.

[48] S. Sakprasat and M. C. Sinclair, "Classification rule mining for automatic credit approval using genetic programming," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 548–555.

[49] C. Qing-Shan, Z. De-Fu, W. Li-Jun, and C. Huo-Wang, "A modified genetic programming for behavior scoring problem," in *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*. IEEE, 2007, pp. 535–539.

[50] E. Carreno, G. Leguizamón, and N. Wagner, "Evolution of classification rules for comprehensible knowledge discovery," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 1261–1268.

[51] R. R. Mendes, F. B. de Voznika, A. A. Freitas, and J. C. Nievola, "Discovering fuzzy classification rules with genetic programming and co-evolution," in *Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 314–325.

[52] A. L. Garcia-Almanza and E. P. Tsang, "Evolving decision rules to predict investment opportunities," *International Journal of Automation and Computing*, vol. 5, no. 1, pp. 22–31, 2008.

[53] U. Bhowan, M. Johnston, and M. Zhang, "Developing new fitness functions in genetic programming for classification with unbalanced data," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 406–421, 2012.

[54] W. Fu, M. Johnston, and M. Zhang, "Low-level feature extraction for edge detection using genetic programming," *Cybernetics, IEEE Transactions on*, vol. 44, no. 8, pp. 1459–1472, Aug 2014.

[55] P. Wang, K. Tang, T. Weise, E. Tsang, and X. Yao, "Multiobjective genetic programming for maximizing roc performance," *Neurocomputing*, vol. 125, pp. 102–118, 2014.

[56] P. Wang, M. Emmerich, R. Li, K. Tang, T. Baeck, and X. Yao, "Convex hull-based multi-objective genetic programming for maximizing receiver operating characteristic performance," *Evolutionary Computation, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[57] P. A. Whigham and G. Dick, "Implicitly controlling bloat in genetic programming," *Evolutionary Computation, IEEE Transactions on*, vol. 14, no. 2, pp. 173–190, 2010.

[58] S. Luke and L. Panait, “A comparison of bloat control methods for genetic programming,” *Evolutionary Computation*, vol. 14, no. 3, pp. 309–344, 2006.

[59] R. Poli, “A simple but theoretically-motivated method to control bloat in genetic programming,” in *Genetic Programming*. Springer, 2003, pp. 204–217.

[60] M. J. Streeter, “The root causes of code growth in genetic programming,” in *Genetic programming*. Springer, 2003, pp. 443–454.

[61] S. Luke and L. Panait, “Fighting bloat with nonparametric parsimony pressure,” in *Parallel Problem Solving from NaturePPSN VII*. Springer, 2002, pp. 411–421.

[62] F. Fernández-de Vega, G. G. Gil, J. A. G. Pulido, and J. L. Guisado, “Control of bloat in genetic programming by means of the island model,” in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 263–271.

[63] F. Fernandez, G. Galeano, and J. Gomez, “Comparing synchronous and asynchronous parallel and distributed genetic programming models,” in *Genetic Programming*. Springer, 2002, pp. 326–335.

[64] M. Castelli, L. Vanneschi, and S. Silva, “Semantic search-based genetic programming and the effect of intron deletion,” *Cybernetics, IEEE Transactions on*, vol. 44, no. 1, pp. 103–113, Jan 2014.

[65] D. Rochat, M. Tomassini, and L. Vanneschi, “Dynamic size populations in distributed genetic programming,” in *Genetic Programming*. Springer, 2005, pp. 50–61.

[66] F. Fernandez, L. Vanneschi, and M. Tomassini, “The effect of plagues in genetic programming: A study of variable-size populations,” in *Genetic Programming*. Springer, 2003, pp. 317–326.

[67] A. Song, Q. Shi, and W. Yin, “Understanding of gp-evolved motion detectors,” *Computational Intelligence Magazine, IEEE*, vol. 8, no. 1, pp. 46–55, 2013.

[68] J. Luna, J. Romero, C. Romero, and S. Ventura, “On the use of genetic programming for mining comprehensible rules in subgroup discovery,” *Cybernetics, IEEE Transactions on*, vol. 44, no. 12, pp. 2329–2341, Dec 2014.

[69] S. Pang, D. Kim, and S. Y. Bang, “Membership authentication in the dynamic group by face classification using svm ensemble,” *Pattern Recognition Letters*, vol. 24, no. 1, pp. 215–225, 2003.

[70] H. Ishibuchi and T. Yamamoto, “Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers,” in *Genetic and Evolutionary ComputationGECCO 2003*. Springer, 2003, pp. 1077–1088.

[71] P. Yang, Y. Hwa Yang, B. B Zhou, and A. Y Zomaya, “A review of ensemble methods in bioinformatics,” *Current Bioinformatics*, vol. 5, no. 4, pp. 296–308, 2010.

[72] K. Nag, T. Pal, and N. Pal, “ASMiGA: An archive-based steady-state micro genetic algorithm,” *Cybernetics, IEEE Transactions on*, vol. 45, no. 1, pp. 40–52, Jan 2015.

[73] R. ENACHE, “Steady state evolutionary algorithms,” Honda Research Institute Europe GmbH, Tech. Rep. HRI-EU Report 04-02, January 2004.

[74] J.-H. Hong and S.-B. Cho, “Gene boosting for cancer classification based on gene expression profiles,” *Pattern Recognition*, vol. 42, no. 9, pp. 1761–1767, 2009.

[75] K. Nag and T. Pal, “A new archive based steady state genetic algorithm,” in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–7.

[76] S. Kernell, “Presidential popularity and negative voting: An alternative explanation of the midterm congressional decline of the president’s party,” *The American Political Science Review*, vol. 71, no. 1, pp. 44–66, 1977.

[77] M. Fang, H. Takauj, S. Kaneko, and H. Watanabe, “Robust optical flow estimation for underwater image,” in *Optomechatronic Technologies, 2009. ISOT 2009. International Symposium on*. IEEE, 2009, pp. 185–190.

[78] M. Fang, H. Takauji, and S. Kaneko, “Rapid computation of robust optical flow by efficient complementary voting,” in *World Automation Congress (WAC), 2010*. IEEE, 2010, pp. 1–6.

[79] M. P. Fiorina and K. A. Shepsle, “Is negative voting an artifact?” *American Journal of Political Science*, pp. 423–439, 1989.

[80] J. Durillo, A. Nebro, and E. Alba, “The jmetal framework for multi-objective optimization: Design and architecture,” in *CEC 2010*, Barcelona, Spain, July 2010, pp. 4138–4325.

[81] J. J. Durillo and A. J. Nebro, “jmetal: A java framework for multi-objective optimization,” *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997811001219

[82] Q. Song, J. Ni, and G. Wang, “A fast clustering-based feature subset selection algorithm for high-dimensional data,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 1, pp. 1–14, 2013.

[83] L. Yu and H. Liu, “Feature selection for high-dimensional data: A fast correlation-based filter solution,” in *Proc. 20th Intl Conf. Machine Leaning*, vol. 20, no. 2, 2003, pp. 856–863.

[84] ——, “Efficient feature selection via analysis of relevance and redundancy,” *The Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[85] M. A. Hall, “Correlation-based feature selection for machine learning,” Ph.D. dissertation, The University of Waikato, 1999.

[86] M. Robnik-Šikonja and I. Kononenko, “Theoretical and empirical analysis of relieff and rrelieff,” *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[87] M. Dash, H. Liu, and H. Motoda, “Consistency based feature selection,” in *Knowledge Discovery and Data Mining. Current Issues and New Applications*. Springer, 2000, pp. 98–109.

[88] H. Almuallim and T. G. Dietterich, “Learning boolean concepts in the presence of many irrelevant features,” *Artificial Intelligence*, vol. 69, no. 1, pp. 279–305, 1994.

[89] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[90] ——, “A comparison of alternative tests of significance for the problem of m rankings,” *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

Kaustuv Nag received the B.Tech. degree in computer science and engineering from the West Bengal University of Technology, Kolkata, India, and the M.Tech. degree in computer science and engineering from the National Institute of Technology, Durgapur, India, in 2010 and 2012, respectively, and is currently pursuing the Ph.D. degree from Jadavpur University, Kolkata.

He was a Visiting Researcher at Indian Statistical Institute, Kolkata. His current research interests include genetic algorithm, genetic programming, and artificial neural networks. Mr. Nag is a recipient of INSPIRE Fellowship.

Nikhil R. Pal (M’91-SM’00-F’05) is a Professor with the Electronics and Communication Sciences Unit of the Indian Statistical Institute. His current research interest includes bioinformatics, brain science, fuzzy logic, pattern analysis, neural networks, and evolutionary computation.

Dr. Pal was the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS from January 2005 to December 2010. He has served/been serving on the editorial/advisory board/steering committee of several journals, including the International Journal of Approximate Reasoning, Applied Soft Computing, International Journal of Knowledge-Based Intelligent Engineering Systems, International Journal of Neural Systems, Fuzzy Sets and Systems, International Journal of Intelligent Computing in Medical Sciences and Image Processing, Fuzzy Information and Engineering: An International Journal, IEEE TRANSACTIONS ON FUZZY SYSTEMS, and the IEEE TRANSACTIONS ON CYBERNETICS. He has given several plenary/keynote speeches in different premier international conferences in the area of computational intelligence. He has served as the General Chair, Program Chair, and Co-Program Chair of several conferences. He was a Distinguished Lecturer of the IEEE Computational Intelligence Society (CIS) and was a member of the Administrative Committee of the IEEE CIS. He is currently the Vice President for Publications of the IEEE CIS. He is a fellow of the National Academy of Sciences, India, a fellow of the Indian National Academy of Engineering, a fellow of the Indian National Science Academy, and a fellow of the International Fuzzy Systems Association.