Multi-objective GAs, Quantitative Indices and Pattern Classification

Sanghmitra Bandyopadhyay, Sankar K Pal, Fellow, IEEE, and B. Aruna
Machine Intelligence Unit
Indian Statistical Institute
Kolkata-700108, India
Email: {sanghami,sankar}@isical.ac.in

Published in IEEE Transactions on Systems, Man and Cybernetics - B, vol. 34, no. 5, pp. 2088-2099, 2004.

Abstract

The concept of multi-objective optimization (MOO) has been integrated with variable length chromosomes for the development of a non-parametric genetic classifier which can overcome the problems, like overfitting/overlearning and ignoring smaller classes, as faced by single objective classifiers. The classifier can efficiently approximate any kind of linear and/or non-linear class boundaries of a data set using an appropriate number of hyperplanes. While designing the classifier the aim is to simultaneously minimize the number of misclassified training points and the number of hyperplanes, and to maximize the product of class wise recognition scores. The concepts of validation set (in addition to training and test sets) and validation functional are introduced in the multi-objective classifier for selecting a solution from a set of non-dominated solutions provided by the MOO algorithm. This genetic classifier incorporates elitism and some domain specific constraints in the search process, and is called the *CEMOGA-Classifier* (constrained elitist multi-objective genetic algorithm based classifier). Two new quantitative indices, namely, the *purity* and *minimal spacing*, are developed for evaluating the performance of different MOO techniques. These are used, along with classification accuracy, required number of hyperplanes and the computation time, to compare the *CEMOGA-Classifier* with other related ones.

Keywords: genetic algorithms, multi-objective optimization, pattern recognition, hyperplane fitting, constrained elitism, pareto-optimality, minimal spacing, purity.

I. Introduction

Supervised pattern classification can be viewed as a problem of generating appropriate class boundaries which can successfully distinguish the various classes in the feature space. In real life problems, the boundaries between different classes are usually nonlinear. It is known that any non-linear surface can be approximated by using a number of hyperplanes. Hence the problem of classification can be viewed as searching for a number of linear surfaces that can appropriately model the class boundaries while providing minimum number of misclassified data points.

Genetic algorithms (*GAs*) [1] are randomized search and optimization techniques guided by the principals of natural selection. They are efficient, adaptive and robust search process having a large amount of implicit parallelism. *GAs* have been successfully applied in various domains including supervised pattern classification [2]. Here, the searching capability of *GAs* has been utilized for appropriately modelling the class boundaries of a data set by approximating them using a number of linear surfaces or hyperplanes. The concept of variable string length *GAs* has been used for this purpose, resulting in a variable length genetic algorithm based classifier, or the *VGA-Classifier* [2]. The *VGA-Classifier* evolves an appropriate number of hyperplanes which minimizes first the number of misclassified points $Miss$ and then the number of hyperplanes H, though the former is given primary/more importance. The superiority of the genetic classifier over the conventional *Bayes* maximum likelihood classifier and the *k-NN rule*, as well as some neural network based classifiers, was demonstrated for a wide variety of data sets.

It may be noted that as a result of the combination of the two objectives, $Miss$ and H, into a single objective function in the *VGA-Classifier*, a situation may result, where a large number of hyperplanes may be utilized for reducing $Miss$ by a small fraction. Consequently, although such a classifier would provide good abstraction of the training data, its generalization capability is likely to be poor.

This article describes a multi-objective *GA* based classifier for pattern recognition which deals with the aforesaid problems. The ability of *GAs* and other evolutionary algorithms for handling complex problems, involving features such as discontinuities, multi-modality, disjoint feasible spaces and noisy function evaluations, as well as their population based nature, makes them possibly well suited for optimizing multiple objectives. The characteristic feature of the genetic classifier is that the class boundaries (approximated by piece wise linear segments) are generated explicitly for making decisions. This is unlike the conventional methods or the multi-layered perceptron (MLP) based approaches, where the generation of the class boundaries is a consequence of the respective decision making processes.

Three multi-objective optimization (MOO) techniques, *NSGA-II* [3] and *PAES* [11], and *CEMOGA* (constrained elitist multi-objective GA), are used for designing the classifier. Of the three, *CEMOGA* is newly developed here. Unlike *NSGA-II*, *CEMOGA* utilizes some domain specific knowledge in the process that helps in increasing the diversity of the system; thereby leading to larger exploration of the search space. The classifiers simultaneously optimize three objectives,

namely, $Miss$, H and *ClassAccuracy*, described in detail in Section III-D. This ensures that overfitting/overlearning is avoided, while classes of smaller size are not ignored during training. Since the MOO algorithms will provide a set of solutions after training, the concepts of validation set (in addition to training and test sets) and validation functional are introduced for selecting one solution from the set. For comparing the performance of different MOO techniques, two new quantitative measures, viz., *purity* and *minimal spacing*, are defined. These two measures, along with the number of hyperplanes required, time required for training, percentage of correct classification, user's accuracy and kappa, are used for comparing the performance of different classifiers.

II. Multi Objective Optimization

Many real world problems involve multiple measures of performance, or objectives, which should be optimized simultaneously. Optimizing multiple objectives involves finding a solution, or possibly a set of solutions, which would provide the values of all the objective functions acceptable to the designer [4]. A general minimization problem of M objectives can be mathematically stated as:

$$\begin{aligned} \text{Minimize } f(x) = [f_i(x), i &= 1, \cdots, M] \\ \text{subject to:} & \\ g_j(x) \leq 0 \qquad j &= 1, 2, \cdots, J, \\ h_k(x) = 0 \qquad k &= 1, 2, \cdots, K, \end{aligned} \tag{1}$$

where $f_i(x)$ is the i^{th} objective function, $g_j(x)$ is the j^{th} inequality constraint and $h_k(x)$ is the k^{th} equality constraint. The MOO problem then reduces to finding x such that $f(x)$ is optimized [5].

In general, the objectives of the optimization problem are often conflicting. Optimal performance according to one objective, if such an optimum exists, often implies unacceptably low performance in one or more of the other objective dimensions, creating the need for a compromise to be reached. A suitable solution to such problems involving conflicting objectives should offer *acceptable*, though possibly sub-optimal in the single objective sense, performance in all objective dimensions, where acceptable is a problem dependent and ultimately subjective concept. An important concept in MOO is that of domination, where a solution x_i is said to dominate another solution x_j if both the following conditions are true:

- The solution x_i is not worse than x_j in all objectives.
- The solution x_i is strictly better than x_j in at least one objective.

This, in turn, leads to the definition of *Pareto-optimality*, where a decision vector $x_i \in U$, where U is the universe, is said to be *Pareto-optimal* if and only if there exists no x_j, $x_j \in U$, such that x_i is dominated by x_j. Solution x_i is said to be *non-dominated*. The set of all such non-dominated solutions is called the *Pareto-optimal set* or the *non-dominated set*. In general, MOO problems tend to achieve a family of alternatives which must be considered equivalent in the absence of information concerning the relevance of each objective relative to the others. Some of the important issues in MOO are designing a suite of test problems with varying degrees of difficulty [6] and defining a set of appropriate quantitative measures for comparing the performance of different MOO strategies [7].

A study establishing the effectiveness of some MOO techniques over their single objective counterparts for the multi-objective set covering problem with varying degrees of difficulty has been carried out in [13]. Theoretical analysis characterizing MOO algorithms [14], and investigating the possibility of generating a unified framework for them, such that individual multi-objective evolutionary algorithms could be generated by varying some parameter within this framework [15] have been taken up in recent studies. The next section describes how the characteristics of multi-objective GAs (MOGAs) may be exploited for developing a non-parametric classifier which models the class boundaries of a data set using a number of hyperplanes.

III. Multi-objective GA Based Classifier

In this section we describe different multi-objective classifiers based on CEMOGA, NSGAII and PAES. Note that we did not choose the earlier versions of MOGA methods e.g., NSGA and NPGA, since these are essentially non elitist in nature. We chose NSGA-II for the purpose of comparison since the operations of the proposed *CEMOGA-Classifier* are based on it. PAES has been chosen since it is a relatively new algorithm based on evolutionary strategies, another widely used evolutionary algorithm.

A. Why Multi-objective Optimization?

As already mentioned, the *VGA-Classifier* [2] used a single criterion, which gave primary importance to $Miss$, and thereafter to H, for optimization during training. The preference given to $Miss$ over H was user defined and adhoc. It could result in overfitting of the data, when a large number of hyperplanes could be used for providing maximum accuracy during training, resulting in reduced generalizability. Thus need for optimizing $Miss$ and H separately and simultaneously was felt.

Although $Miss$ provided a measure of the overall classification accuracy, it did not incorporate any class wise information. Therefore, a new term called $ClassAccuracy$, defined as the product of the class wise recognition scores, is taken as the third objective for optimization. It may be noted that minimizing $Miss$ does not automatically lead to maximizing $ClassAccuracy$, and vice verse. For example, consider a data set, having two classes, with 100 points in class 1 and 10 points in class 2. Assume that in one situation, 50% of both the classes are correctly recognized. Hence, $ClassAccuracy$ is 0.25 and $Miss = 55$. In another situation, let 60% and 40% of the classes be recognized respectively. Here the $ClassAccuracy$ becomes 0.24, while $Miss$ is 46. It is evident that although in the first case $ClassAccuracy$ is more, in the second $Miss$ is smaller. The question as to which one is better is therefore seen to be problem dependent, and cannot be resolved without any domain knowledge.

Since any combination of the three objectives would, in general, be adhoc, it is imperative to adopt an MOO strategy. An important consequence of this is that the decision maker will be provided with a set of possible alternative solutions, as well as intermediate solutions, which the decision maker may subsequently refine. The next section describes how the principles of multi objective GA for can be used for the placement of an appropriate number of hyperplanes such that $Miss$ (the number of misclassified points), $ClassAccuracy$ (the product of the rates of correct classification of each class), and H (the number of hyperplanes H required to model the class boundaries) can be simultaneously optimized.

B. Principle of Hyperplane Fitting

The multi objective genetic classifiers attempts to place H hyperplanes in the feature space appropriately such that all the above mentioned objective functions are optimized. As is known, a hyperplane in D dimensional space can be represented by $D-1$ angle variables α_j, $j = 1, 2, \ldots, D-1$, and a perpendicular distance variable d. Each angle variable α_j is allowed to vary in the range of 0 to 2π, the length of the intervals being determined by the number of bits, b_1, used to represent the angles. Once the angles are fixed, the orientation of the hyperplane becomes fixed. Now only d must be specified in order to specify the hyperplane. For this purpose the hyper rectangle enclosing the training points is considered. Let $diag$ be the length of the diagonal of this hyper rectangle. A hyperplane is designated as the *base hyperplane* with respect to a given orientation (i.e., for some $\alpha_1, \alpha_2, \ldots, \alpha_{D-1}$) if

- it has the same orientation
- it passes through one of the vertices of the enclosing rectangle
- its perpendicular distance from the origin is minimum (among the hyperplanes passing through the other vertices). Let this distance be d_{min}.

If b_2 bits are used to represent d, then a value of v_2 in these bits represent a hyperplane with the given orientation and for which d is given by $d_{min} + \frac{diag}{2^{b_2}} * v_2$. Details of these are available in [2].

C. Chromosome Representation

The chromosomes are represented by strings of 1, 0, and # (don't care), encoding the parameters of variable number of hyperplanes. If angle and distance bits are represented by b_1 and b_2 bits, respectively, then the length, l_i, of the ith chromosome encoding H_i ($1 \leq H_i \leq H_{Max}$, where H_{Max} represents the maximum number of hyperplanes, specified a *priori*) hyperplanes is

$$l_i = H_i * [(D-1) * b_1 + b_2] \qquad i \in (1, \cdots, H_{Max}). \tag{2}$$

Note that if the number of classes is represented by C, then the minimum number of hyperplanes, H_{Min}, required to generate those many regions is :

$$H_{Min} = \log_2 C \tag{3}$$

D. Fitness Computation

The fitness of a chromosome is assessed on the basis of $Miss$, H and $ClassAccuracy$.

Computing Miss :

The number of misclassified points for a string i encoding H_i hyperplanes is found as follows : Let the H_i hyperplanes provide M_i distinct regions which contain at least one training data point. (Note that although $M_i \leq 2^{H_i}$, in reality it is bounded by the size of the training data set.) For each such region and from the training data points that lie in this region, the class of the majority is determined, and the region is considered to represent (or be labeled by) the said class. Points of other classes that lie in this region are considered to be misclassified. The sum of the misclassifications for all the M_i regions constitutes the total misclassification $Miss_i$ associated with the string.

Computing ClassAccuracy :

For a data set having C classes, the fraction of each class, j, correctly classified, $ClassAccuracy_j$, is defined as :

$$ClassAccuracy_j = \frac{n_{j_c}}{n_j} \tag{4}$$

where n_j and n_{j_c} represent the number of points and the number of correctly classified points of class j, respectively. Overall class accuracy of a chromosome, considering all the classes is calculated as $ClassAccuracy = \Pi_{j=1}^{C} ClassAccuracy_j$.

Calculating H :

If the length of a chromosome is l_i, then the number of hyperplanes encoded in the chromosome is calculated as follows:

$$H_i = \frac{l_i}{(D-1) * b_1 + b_2} \tag{5}$$

Any chromosome with $H = 0$ is assigned $ClassAccuracy = 0$ so that it will be automatically eliminated during selection.

E. Selection

In MOO, the selection procedure differs significantly from the corresponding single objective version. This section describes, in brief, the selection schemes used in the *CEMOGA-Classifier* and *NSGAII-Classifier*[3] since the operations of *CEMOGA-Classifier* are based on those in *NSGAII-Classifier*. Details of the selection scheme may be found in [3], [16]. The *PAES-Classifier* is described separately in Section III-I.

The chromosomes in a population are first sorted based on their *domination* status using the procedure *Non-dominated sort* [3], [16]. Here the chromosomes that are not dominated by any other member of the population form the first front, and are put in rank 1. These chromosomes are then removed from consideration, and the chromosomes which thereafter become non-dominated are assigned rank 2. The process is repeated until all the chromosomes have been assigned a rank. The chromosomes are then put in a sorted order according to their ranks. The overall complexity of the *Non-dominated sort* algorithm described in detail in [16] is shown to be $O(MN^2)$, where M is the number of objectives and N is the number of chromosomes in the population. The first front represents a *non-dominated set* with respect to the current population, since none of the solutions in this front is dominated by any other solution in the population.

After applying the *Non-dominated sort* algorithm the chromosomes are assigned a *crowding distance* and selection is performed using the *crowded tournament selection*. The *crowding distance* is computed as the sum of the difference of the objectives values of the solutions preceding and following the current solution (corresponding to the chromosome under consideration) for each objective [3], [16]. This provides a measure of the density of the solutions surrounding a particular point in the population.

In *crowded tournament selection*, a pair of chromosomes is selected randomly, and the one with the lower rank is selected. If the ranks of the chromosomes are equal, then the chromosome with a larger crowding distance is chosen. The large crowding distance ensures that the solutions are spread along the *Pareto-optimal front*.

F. Crossover

The crossover operation is modified to be applicable to two strings of (possibly) different lengths. The lengths of the two is first made equal by padding the smaller chromosome with #s. Conventional single point crossover is then performed on them with a probability μ_c. The resultant chromosome may have some *valid* and some *invalid* hyperplanes. (A hyperplane is valid if all the bits representing this plane are either $defined$, i.e., 0s and 1s, or #, i.e., undefined; it is invalid otherwise.)

If a hyperplane is invalid, let u = number of defined bits in that hyperplane. Then, all its # bits are set to defined bits with a probability u/l_H. In case this is not permitted, all the defined bits are set to #s. After each hyperplane becomes valid, all the #s are pushed to the end of the chromosome so that we have all the hyperplanes in the beginning of the chromosome. Details are available in [2].

G. Mutation

As in crossover, the mutation operation is also modified to tackle variable length strings. The length of a chromosome is first made equal to the maximum chromosome length by appropriately padding with #s. Then for each defined bit position, mutation is carried out with *mutation probability* μ_m. μ_m is varied with the number of iterations such that it goes though a complete cycle of high values, then low values, then high followed by low values once again. Note that when μ_m is high, greater exploration of the search space results, while exploitation is given more importance for low values of μ_m. By alternating this cycle, we intend to provide a better balance between exploration and exploitation of the search space; a crucial factor in the good performance of GAs. The variation of μ_m with the number of generations is depicted in Fig. 1 [17].

If mutation cannot be carried out on a defined bit, it is set to # with a probability μ_{m1}. Each undefined bit is set to a defined bit (chosen randomly) with probability μ_{m2}. The resultant chromosome may again have some *invalid* hyperplanes which are handled in the same way as in crossover. Note that mutation operation on defined bit may result in a decrease in the chromosome length, while the operation on # positions may result in an increase in the chromosome length. In this article we have fixed the values μ_{m1} and μ_{m2}, and have kept them equal so that the probability of increasing and decreasing the string length is the same.

H. Incorporating Elitism

In this section we describe how elitism is incorporated in the *NSGAII-Classifier* and *CEMOGA-Classifier*. As for selection, these are again mentioned together since they are somewhat similar, though the procedure in *CEMOGA-Classifier* is so designed such that some domain specific constraints are incorporation.

H.1 Elitism in NSGAII-Classifier

Here the parent population and the child population, obtained using selection, crossover and mutation on the parent population, are first combined together to form a combined generation of size $2N$. The combined population is sorted and the best N chromosomes are selected from this combined generation to constitute the new parent generation. Chromosomes from rank 1 get selected first, followed by rank 2 chromosomes and so on till the total number of chromosomes exceeds N. Thereafter the last front chromosomes are removed, and they are selected one by one on the basis of their crowding distance, so that the total number of chromosomes in the new parent generation is N. This new parent generation again undergoes crowded tournament selection, crossover and mutation to form the child generation. This process is repeated for a maximum number of generations, fixed a *priori*.

H.2 Elitism in CEMOGA-Classifier

As in the *NSGAII-Classifier*, an initial population of size N is created. This is also called the *combined generation* for the sake of convenience. Note that, although in the initial stage, the size of the *combined generation* is equal to N, for subsequent generations this will be equal to $N+N_C$, where N_C is the size of the *constrained generation*, described below. Now two sets of operations are invoked on the same *combined generation*. Firstly, crowded tournament selection is used to select N chromosomes from the *combined generation*, after which crossover and mutation operators are applied to form a generation of size N, which is called the *current generation*. Simultaneously, a *constrained generation* is formed from the *combined generation* by selecting chromosomes which satisfy the following constraints:

1. H is greater than H_{Min} (Eqn. 3).
2. *Miss* is less than $\alpha_{m1} * S$, for the first θ_1 generations and less than $\alpha_{m2} * S$ for later generations, where S is the size of the data sets and α_{m1}, $\alpha_{m1} < 1$.
3. *ClassAccuracy* is greater than zero for the first θ_2 generations, and greater than α_{c1} for the later generations, where $0 < \alpha_{c1} < 1$.

Note that the size of *constrained generation*, N_C, may vary, depending upon the number of chromosomes that satisfy the constraints, but is restricted to $\delta *$ N, where $\delta > 0$. If the number of chromosomes that satisfy the constraints exceeds $\delta *$ N, then in a process analogous to the one described in Section III-H.1, the chromosomes from the first front onwards, and which satisfy the constraints, are added to the *constrained generation* one after the other till the size of the *constrained generation* exceeds the limit. Thereafter, the last front chromosomes are removed from the *constrained generation*, and crowded tournament selection is applied to select the appropriate number of chromosomes from this front. We call this process *Crowded ConstrainedGen Formation*. The *current generation* and the *constrained generation* are combined to form the *combined generation*, which will now be of size $N + N_C$. The entire process is continued for a specified number of generations.

Note that in contrast to the elitism procedure for *NSGAII-Classifier*, where better ranked chromosomes are retained, even though they may not satisfy the practical constraints of the problem, this is not the case in *CEMOGA-Classifier*. Here, the *constrained generation* is formed in such a way that even lower ranked chromosomes have a chance of surviving, provided they satisfy the constraints. This, in turn, leads to a better diversity, and hence larger exploration of the search space, which is likely to result in improved performance. Fig. 2 illustrates the $CEMOGA$ procedure.

*I. The Classifier Based on Pareto Archived Evolution Strategy (*PAES-Classifier*)*

PAES-Classifier utilizes the Pareto-archived evolution strategy ($PAES$) [11] as the underlying MOO technique. PAES uses a $(1 + 1)$ evolutionary strategy (ES) [18]. This algorithm performs a local search by using a small change operator to move from one solution to a nearby solution. At any generation t, a parent solution (P_t), offspring solution (C_t), and a number of best solutions called *Archive* are maintained. The offspring (C_t) is obtained by applying the mutation operator on a copy of (P_t). The parent (P_t) and offspring (C_t) are compared for domination, and based on different scenarios (C_t) may be rejected or accepted, and the *Archive* may or may not be updated.

To decide who qualifies as parent (as well as to decide a victim for deletion when the archive overflows), the density of solutions in the neighbourhood is checked and the one residing in the least crowded area qualifies as parent. (In the case of selecting a victim for deletion, one residing in the most crowded area is selected.) For determining density, each objective space is divided into 2^d equal divisions where d is a user defined *depth* parameter. Therefore, the entire search space is divided into $(2^d)^M$ unique, equal sized M-dimensional hypercubes. The number of solutions in each hypercube is counted. This is used as a measure of the density of the solutions in the archive.

J. Validation and Testing

In contrast to the single objective *VGA-Classifier* [2] which provides only a single final solution, the multi-objective classifiers described in this article provide a set of solutions. These are the chromosomes in the first front (or, rank 1) of the final population for the *NSGAII-Classifier* and *CEMOGA-Classifier*, while for the *PAES-Classifier* these are found in the archive. A validation function therefore needs to be defined to select a chromosome which is best suited for the problem at hand; and which can be subsequently used for designing the final classifier. Accordingly, the data set has to be divided into three parts: training, validation and test sets.

In the present investigation, the validation function, applied on the validation set, is defined as follows:

$$Validity = 0.5 * ClassAccuracy + 0.3 * \left(\frac{V - Miss}{V}\right)^C - 0.2 * \left(\frac{H}{H_{Max}}\right)^C \tag{6}$$

where V and C are the size of validation set and the number of classes, respectively. From Eqn. 4, it is found that for individual classes, the value of $ClassAccuracy_j$ can range from 0 to 1. The overall $ClassAccuracy$ is a product of the C such $ClassAccuracy_j$ values corresponding to C classes. Therefore, in order to combine the three objectives into a single validity function, the terms corresponding to $Miss$ and H are so defined that these values also range from 0 to 1, and thereafter these are raised to the power of C. The coefficients reflect the relative importance of the three objectives. It may be noted that these values may be changed depending on the problem domain and other user specific considerations. Moreover, the validity function may be defined in some other form as well.

The classifiers designed by the chromosomes of the first front of the last generation are subjected to the validation phase. The validation function ($Validity$) is then computed corresponding to each classifier. The classifier providing the highest value of $Validity$ is taken as the solution.

The final classifier thus obtained from the validation phase is then used for testing. For each unknown pattern in the test set, the region in which it lies is first determined. The pattern is then labeled by the class associated with that region.

IV. Indices for Comparing MO Solutions

In general, the solutions obtained using any MOO algorithm should be as close as possible to the Pareto-optimal front. Moreover, the solutions should be distributed uniformly over the solution space. In other words, a maximal set of solutions over the Pareto-optimal front is desired which must cover the entire gamut of the front, and also be uniformly distributed over it. Some measures of comparing the non-dominated solutions provided by different MOO strategies are discussed in this section. We also define two new measures in this context.

A. Measures Based on Position of Non-dominated Front

One of the ways of evaluating the performance of MOO algorithms is based on the position of the non-dominated front (surface formed by the non-dominated solutions). Some such measures are the error ratio [19] and set coverage metric [20]. The first measure, error ratio, assumes that a knowledge of the Pareto-optimal set is available, an information that may not be readily available in real-life problems. The other measure, set coverage, essentially computes the relative goodness of only two MOO strategies. Another measure is now formulated that can be used to compare the solutions obtained by not only two but several MOO strategies (which is in contrast to the set coverage measure), while not assuming any knowledge about the Pareto-optimal front (which is in contrast to the error ratio). Such a new measure, called *purity*, is described as follows.

Suppose there are N, $N > 2$, MOO strategies applied to a problem. Let $r_i = |R_1^i|$, $i = 1, 2, \ldots, N$ be the number of rank one solutions obtained from each MOO strategy. Compute the union of all these solutions as $R^* = \bigcup_{i=1}^{N} \{R_1^i\}$. Thereafter a ranking procedure is applied on R^* and obtain the new rank one solutions, called R_1^*. Let r_i^* be the number of rank one solutions which are present in R_1^*. That is,

$$r_i^* = |\{\gamma | \gamma \in R_1^i \text{ and } \gamma \in R_1^*\}|. \tag{7}$$

Then the purity measure for the ith MOO strategy, P_i, is defined as

$$P_i = \frac{r_i^*}{r_i}, \qquad i = 1, 2, \ldots, N. \tag{8}$$

Note that the purity value may lie between $[0, 1]$, where a value nearer to 1 indicates a better performance.

B. Measures Based on Diversity of the Solutions

Schott [21] suggested a measure called spacing which reflects the uniformity of the distribution of the solutions over the non-dominated front. Spacing (S) between solutions is calculated as

$$S = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - d)^2}, \tag{9}$$

where $d_i = \min_{k \in Q \text{ and } k \neq i} \sum_{m=1}^{M} |f_m^i - f_m^k|$ and f_m^i (or, f_m^k) is the mth objective value of the ith (or, kth) solution in the final non-dominated solution set Q. d is the mean value of all the d_is. Note that a value of S nearer to 0 is desirable since it indicates that the solutions are uniformly distributed over the Pareto-optimal front. This diversity measure can be calculated for solution sets where there are more than two solutions in the set. Figure 3 (a) and (b) demonstrates a common scenario when this measure is expected to fail. In Fig. 3 (a), since point a has b as its closest neighbor and vice versa, the value of S will be low, indicating wrongly a uniform spread over the Pareto-optimal front. This measure is unable to indicate that a large gap exists between b and c. Note that with this assumption the value of S will be low in Fig. 3 (b) also. It is obvious that the situation in Fig. 3 (b) if preferable, since it shows a more uniform distribution over the Pareto-optimal front. However, the measure S is unable to indicate this. We overcome the above limitation of S by defining here a modified measure called *minimal spacing*, S_m. The essence of this measure is to consider the distance from a solution to its nearest neighbor which has not already been considered. The way S_m is calculated among the solutions in the non-dominated set is as follows:

Initially consider all solutions as unmarked. Take a solution and call it the seed. Mark the seed. Start computing the nearest distance from the last marked solution (initially the seed). Each time while calculating the minimum distance between two solutions, consider only those solutions which are still unmarked. Once a solution is included in the distance computation process, make it as marked. Continue this process until all solutions are considered and keep track of the sum of the distances.

Repeat the above process by considering each solution as seed and find out the overall minimum sum of distances. The sequence of solutions and the corresponding distance values are used for the computation of the S_m, where again Eqn. 9 is used with $|Q|$ replaced by $|Q|$-1 (since here we have $|Q|$-1 distances). As for S, a lower value of S_m indicates better performance of the corresponding MOO technique. Note that with this definition, the value of S_m in Fig. 3 (b) will be lower than that in Fig. 3 (a), indicating that the former solution set is better. In this regard, it may be mentioned that the range of values of the different objectives often varies widely. Consequently, they are generally normalized while computing the distances. In other words, when computing the d_is, the term $|f_m^i - f_m^k|$ is divided by $|F_m^{max} - F_m^{min}|$ in order to normalize it, where F_m^{max} and F_m^{min} are the maximum and minimum values respectively of the mth objective.

V. Simulation and Results

Several data sets, with the number of dimensions varying from two to eighteen and number of classes varying from two to six, were used to demonstrate the effectiveness of the afore mentioned genetic classifiers. Each data was divided into three different groups of *training, validation and test* which comprised 20%, 30% and 50% of the original set respectively.

In the following subsections we first describe the different data sets. The experimental parameters adopted for the implementation of the classifiers are provided in the next subsection, followed by a a detailed description of the comparative results.

A. Data Sets

Vowel: This consists of 871 Indian Telugu vowel sounds and three features F_1, F_2 and F_3, corresponding to the first, second and third vowel formant frequencies, and six classes $\{\delta, a, i, u, e, o\}$ [22]. Figure 4 shows the data in $F_1 - F_2$ plane with six different symbols representing the six classes. As can be seen from the figure, the boundaries of the classes are ill-defined and overlapping.

Iris: This data represents three different categories of the flower, iris namely, Setosa, Versicolor and Virginica, and is characterized by four feature values [23]. It has 150 points with 50 samples per class. It is known that two classes Versicolor and Virginica have a large amount of overlap while the class Setosa is linearly separable from the other two.

Cancer: This breast cancer database, obtained from the University of Wisconsin Hospital, Madison, has 683 breast mass samples belonging to two classes *Benign* and *Malignant*, in nine dimensional feature space. The nine feature values correspond to *clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli* and *mitoses*. The data set is available at [http://www.ics.uci.edu/∼mlearn/MLRepository.html].

Landsat: The Landsat data, obtained by the satellite *Landsat-V*, represents satellite imagery data of five classes of rocks, vegetation and soil, namely, Manda Granite, Romapahari Granite, Vegetation, Black Phillite and Alluvium. The

data set with 795 samples and two features is shown in Fig. 5. The details of extraction procedure are available elsewhere [24], [25].

Mango: Mango data set [26] consists of eighteen measurements taken of the leaves of three different kinds of mango trees. It has 166 samples with eighteen features each. It has three classes representing three kinds of mango. The feature set consists of measurements like Z-value, area, perimeter, maximum length, maximum breadth, petiole, K-value, S-value, shape index, upper midrib/ lower midrib and perimeter upper half/ perimeter lower half. The terms *upper* and *lower* are used with respect to the maximum breadth position.

Crude Oil: This data set has 56 samples with three classes and five features [27]. The three classes, consisting of 7, 11 and 38 patterns respectively, correspond to three types of oil. The five features are vanadium (in percent ash), iron (in percent ash), beryllium (in percent ash), saturated hydrocarbons (in percent area) and aromatic hydrocarbons (in percent area).

B. Parameter Values

For *CEMOGA-Classifier* and *NSGAII-Classifier*, the population size and crossover probability were kept fixed at 20 and 0.85, respectively while the conventional mutation probability was varied in the range [0.45,0.002]. μ_{m1} and μ_{m2} were fixed at 0.95. The values of b_1 and b_2 were chosen to be 8 and 16, respectively. Maximum number of generations and maximum number of hyperplanes were kept fixed at 3000 and 20, respectively. The values of θ_1 and θ_2, as described in Section III-H, were set to 50. The values of α_{m1}, α_{m2} and α_{c1} were chosen to be 0.50, 0.35 and 0.05, respectively. Since the *PAES-Classifier* operates on only one string at a time, in order to keep the comparison fair, it was executed for 60,000 generations (equal to the population size (20) times the maximum number of generations (3000) for the genetic classifiers). The maximum archive size was fixed at 30, and the depth parameter was kept equal to 3.

C. Results

This section is divided into two parts. In the first part, a comparison of the performance of all the classifiers is provided for the eight data sets using different classification measures. The non-dominated solutions obtained by the three multi-objective classifiers are compared in the second part of this section, using the measures, *purity* and *minimal spacing*, defined in Section IV.

C.1 Comparison of the Classification Performance

The classification performance of the classifiers are compared using several measures, namely, number of hyperplanes, percentage recognition scores, class accuracy, user's accuracy and kappa [28], [29]. User accuracy (U_a) is a measure of confidence that a classifier attributes to a region as belonging to a class. It is defined as $U_a = n_{i_c}/n_i^p$, where n_{i_c} is defined earlier in Eqn. 4 and n_i^p is the number of points classified into class i.

The coefficient of agreement (*kappa*) measures the relationship of beyond chance agreement to expected disagreement. The estimate of kappa for class i (κ_i) is defined as

$$\kappa_i = \frac{n * n_{i_c} - n_i * n_i^p}{n * n_i^p - n_i * n_i^p} \tag{10}$$

The numerator and denominator of the overall kappa are obtained by summing the respective numerators and denominators of κ_i separately over all classes i.

Tables I and II show the comparative results of the three multi-objective classifiers on different data sets during training and testing, respectively. As can be seen from Table I, the *CEMOGA-Classifier* uses a smaller number of hyperplanes than the remaining classifiers for approximating the class boundaries for all the data sets, except *Mango*. In general, using a large number of hyerplanes for approximating the class boundaries is likely to lead to overfitting of the data; thereby leading to better performance during training (better abstraction), but poorer generalization capability. This is supported by the results provided in Tables I and II where it is seen that although the percentage recognition scores during training of the *CEMOGA-Classifier* is not the best for all the data sets, its performance during testing is consistently better than those of the other classifiers for all the data sets.

The only departure with regard to the number of hyperplanes is observed for *Mango*, where the *NSGAII-Classifier* uses only three hyperplanes, while the *CEMOGA-Classifier* uses six. However, it can be seen from Table II that the *NSGAII-Classifier* (as also the *PAES-Classifier*) provides a *ClassAccuracy* values are zero, indicating that at least one class has not been recognized at all. In contrast, the *CEMOGA-Classifier* uses six hyperplanes to recognize all the three classes to a reasonable extent, thereby providing a *ClassAccuracy* value of 0.28. Table III which shows the classwise percentage recognition scores for all the three classifiers during testing also demonstrates this observation. As can be seen, only *CEMOGA-Classifier* is able to recognize, to a large extent, all the three classes. In contrast, the *VGA-Classifier* fails to recognize class 2, while the *NSGAII-Classifier* and *PAES-Classifier* fail to recognize class 3. In fact in both these cases it was found that there were no patterns that were either correctly or incorrectly classified to

the respective classes. (It may be mentioned here that the *VGA-Classifier* was found to be unable to recognize class 2 while providing an overall recognition score of 55.43%.)

As is evident from the kappa values provided in Table II the *CEMOGA-Classifier* consistently provides the best result for all the data sets, once again highlighting its superior performance. In order to study the classwise user's accuracy, the *Vowel* data set was chosen, as an example, since it has been used extensively in other experiments with the *GA* based classifiers [30]. It was found that the user's accuracy and kappa values were, in general, relatively poorer for Class 1 (δ) for all the classifiers. The reason for this is evident from Fig. 4, where it is found that this class is largely overlapped with its neighboring classes. Again, the user's accuracy values for classes 3 (i) was found to be the highest for *CEMOGA-Classifier* and *PAES-Classifier*, while for *NSGAII-Classifier*, this was highest for class 4 (u). It may be noted in this context that both these classes have only a little overlap with the neighboring classes, and hence obtaining high user's accuracy for them is expected.

Regarding the time taken for training the classifiers, it was found that as expected, the multi-objective classifiers took more time than the single-objective *VGA-Classifier*. In general, the *NSGAII-Classifier* and *CEMOGA-Classifier* took similar time since their operations are similar, except that *CEMOGA-Classifier* requires an additional step of checking for the constraints, while *PAES-Classifier* took more time to train. As an illustration, for *Vowel* data, the time taken by the *CEMOGA-Classifier*, *NSGAII-Classifier* and *PAES-Classifier* were 6.15 min, 5.604 min and 19.816 min respectively when executed on SUN workstation.

C.2 Comparision with Respect to Obtained Non-Dominated Solutions

Table IV demonstrates the number of non-dominated solutions and the corresponding purity values obtained by the three MOO strategies for the six data sets. Interestingly, in all the cases, *CEMOGA-Classifier* attains a purity value of 1, indicating that all the non-dominated solutions obtained by it are indeed non-dominated even when the solutions of all the three algorithms are combined. On the other hand, almost none of the solutions obtained using *PAES-Classifier* was found to be actually non-dominated in the combined scenario, while *NSGAII-Classifier* lies some where in between. In this regard it may be noted that *CEMOGA-Classifier*, in general, finds a smaller number of better non-dominated solutions than the other two algorithms. For the purpose of demonstration, the non-dominated solutions obtained using these three algorithms are shown in Fig. 6 for the *Vowel* data set. Note that when the solutions of all the three classifiers are combined and they are re-ranked it is found that all the *CEMOGA-Classifier* solutions actually belong to rank 1, none of the *PAES-Classifier* solutions belong to rank 1, while five of the *NSGAII-Classifier* solutions (encircled in the figure as $\oplus$ for convenience) belong to rank 1.

Table V depicts the values of minimal spacing (S_m) obtained by the three methods. This is reported for both the cases; one when all the respective solutions are considered, and second when only actual rank 1 solutions (obtained after combination of the solutions) are considered. The result in the second category is denoted by S_m (R1) in the table. Interestingly, the two values are exactly the same for *CEMOGA-Classifier*, for all the data sets, since all the solutions obtained by this classifier actually belong to rank 1, as indicated by the purity values (Table IV). As can be noted from Table V, in several cases (denoted by '-') it was not possible (or meaningful) to calculate S_m, since the number of solutions was less than or equal to two. In cases where it was possible to compute this value, *CEMOGA-Classifier* provided significantly better performance (i.e., smaller values), in general. The only exception is observed for *Crude Oil* where the value of S_m, provided by *NSGAII-Classifier*, is seen to be much better. However, when only those solutions that are actually non-dominated (denoted by R1) are considered, the *CEMOGA-Classifier* once again outperforms *NSGAII-Classifier*. Note that for *PAES-Classifier*, it was not possible to compute minimal spacing (R1) values for any of the data sets, since in all but one case (*Landsat* data set) the purity values are 0, and in the case of *Landsat*, only one solution is found to finally belong to rank 1.

VI. Discussion and Conclusions

In this article, a pattern classification methodology based on multi-objective variable string length genetic algorithm has been developed. The problem of generating class boundaries for distinguishing the different classes is posed as one of optimizing multiple objectives. Since the class boundaries are approximated by a variable number of hyperplanes, variable string length MOGA is used as the underlying search tool. Three objectives, $Miss$ (total number of misclassified samples), H (number of hyperplanes) and $ClassAccuracy$ (product of the classwise correct recognition rates), are used as the optimizing criteria. The first two objectives need to be minimized, whereas the third one is to be maximized simultaneously. Three techniques of MOO, viz., NSGA-II and PAES, and CEMOGA, have been used for developing the classifiers. Of these, CEMOGA is newly developed here. Unlike NSGA-II, CEMOGA uses some domain specific constraints in the process that helps in making the search more efficient. Since the multi-objective classifiers, in general, provide a set of non-dominated solutions, a validation phase is used after training in order to select one of the chromosomes (which provided the largest value of the validation function) to be used for testing.

The superiority of the *CEMOGA-Classifier* as compared to *NSGAII-Classifier* and *PAES-Classifier* has been demonstrated for several real life data sets. Comparison is made with respect to both the number of required hyperplanes and

classification performance, as measured by the recognition score, user's accuracy and kappa value. It is found that, in general, the *CEMOGA-Classifier* is able to approximate the class boundaries using a smaller number of hyperplanes; thereby providing superior generalization capability. The only point of departure was for the *Mango* data, where although the remaining classifiers required smaller number of hyperplanes as compared to the *CEMOGA-Classifier*, they were unable to recognize one of the three classes. In contrast, only the *CEMOGA-Classifier* was able to recognize all the three classes, and thus provided superior results during testing.

The non-dominated solutions obtained by the three MOO algorithms, NSGA-II, PAES and CEMOGA, are also compared with respect to the position of the corresponding fronts as well as the diversity of the solutions. For the former, a measure, called *purity*, has been defined that computes the fraction of solutions that remain non-dominated when the solutions resulting from several MOO techniques are combined. In order to measure the diversity, or spread, of the solutions on the Pareto-optimal front, an index called *minimal spacing* has been defined. This is an improved version of an existing measure called *spacing*. Interestingly, it is found the CEMOGA consistently provides a *purity* value of 1 for all the data sets, NSGA-II does so for only two of the six data sets, while PAES is unable to do so even once. With respect to *minimal spacing*, in general, CEMOGA again outperforms the other MOO strategies. Only for *Crude Oil*, NSGA-II outperforms CEMOGA. However, when only the truly non-dominated solutions of NSGA-II are considered for computing *minimal spacing*, CEMOGA is once again found to be superior.

The major focus of this article has been to lay the empirical foundation for developing an efficient, non-parametric MOGA based supervised pattern classifier. In future, a theoretical analysis of the classifier needs to be performed. Further, sensitivity analysis of the classifier with respect to different genetic parameters should be carried out. Utility of other recent MOO techniques like particle swarm optimization [31] needs to be studied. Finally, the application of the classifier in other domains, viz., image patterns, symbolic data, needs to be studied.

References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley, 1989.

[2] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "*VGA-Classifier*: Design and applications," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 30, no. 6, pp. 890–895, 2000.

[3] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, (Paris, France), pp. 849–858, Springer. Lecture Notes in Computer Science No. 1917, 2000.

[4] C. A. Coello Coello and A. D. Christiansen, "Multiobjective optimization of trusses using genetic algorithms," *Computers and Structures*, vol. 75, no. 6, pp. 647–660, 2000.

[5] C. A. Coello Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil Engineering and Environmental Systems*, vol. 17, pp. 319–346, 2000.

[6] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of the Congress on Evolutionary Computation*, (Honolulu, USA), pp. 825–830, IEEE Press, 2002.

[7] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[8] C. A. Coello Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information Systems*, vol. 1, no. 3, pp. 129–156, 1999.

[9] N. Srinivas and K. Deb, "Multi objective optimization using nondominated sorting in genetic algorithm," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[10] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched pareto genetic algorithm.," Tech. Rep. IlliGAL Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[11] J. D. Knowles and D. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[12] J. D. Knowles, M. J. Oates, and D. W. Corne, "Advanced multiobjective evolutionary algorithms applied to two problems in telecommunications," *BT Technology Journal*, vol. 18, no. 4, pp. 51–65, 2000.

[13] A. Jaszkiewicz, "Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem," *IEEE. Trans. Evolutionary Computation*, vol. 7, no. 2, pp. 133–143, 2003.

[14] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multi-objective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

[15] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE. Trans. Evolutionary Computation*, vol. 7, no. 2, pp. 174–188, 2003.

[16] K. Deb, *Multi-objective Optimization Using Evolutionary Algorithms*. England: John Wiley and Sons, Ltd, 2001.

[17] S. K. Pal, S. Bandyopadhyay, and C. A. Murthy, "Genetic algorithms for generation of class boundaries," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 6, pp. 816–828, 1998.

[18] H. P. Schwefel, "Collective phaenomena in evolutionary systems," in *31st Ann. Meet. Intl. Soc. for General System Research*, (Budapest), pp. 1025–1033, 1987.

[19] D. V. Veldhuizen, *Multiobjective Evolutionary algorithms: Classification, Analyses, and new Innovations*. PhD thesis, Air Force Institute of Technology, Dayton, OH, 1999.

[20] E. Zitzler, *Evolutionary algorithms for multiobjective optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

[21] J. R. Schott, *Fault tolerant design using single and multi-criteria genetic algorithms*. PhD thesis, Dept. of Aeronautics and Astronautics, Massachussets Institute of Technology, Boston, MA, 1995.

[22] S. K. Pal and D. D. Majumder, "Fuzzy sets and decision making approaches in vowel and speaker recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 625–629, 1977.

[23] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 3, pp. 179–188, 1936.

[24] A. Pal, *On a Class of Stochastic Approximation-Type Parameter-Learning Algorithms for Pattern Recognition*. PhD thesis, Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta, India, 1990.

[25] A. Pal, "Some applications of GGA for automatic learning of class parameters in the presence of wrong samples," *Inform. Sci.*, vol. 67, pp. 189–208, 1993.

[26] S. K. Pal, "Fuzzy set theoretic measures for automatic feature evaluation-II," *Inform. Sci.*, vol. 64, pp. 165–179, 1992.

[27] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Prentice-Hall, 1982.

[28] R. G. Congalton, R. G. Oderwald, and R. A. Mead, "Assessing landsat classification accuracy using discrete multivariate analysis statistical techniques," *Photogrammetric Engineering and Remote Sensing*, vol. 49, pp. 1671–1678, 1983.

[29] G. H. Rosenfield and K. Fitzpatrik-Lins, "Coefficient of agreement as a measure of thematic classification accuracy," *Photogrametic Engineering and Remote Sensing*, vol. 52, pp. 223–227, 1986.

[30] G. Weiss and F. Provost, "The effect of class distribution on classifier learning," Tech. Rep. ML-TR 43, Department of Computer Science, Rutgers University, 2001.

[31] S. Mostaghim and J. Teich, "Strategies for finding good local guides in multi-objective particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 26–33, 2003.

Biographical Information of Dr. Sanghamitra Bandyopadhyay

Sanghamitra Bandyopadhyay did her Bachelors in Physics and Computer Science in 1988 and 1991 respectively. Subsequently, she did her Masters in Computer Science from Indian Institute of Technology (IIT), Kharagpur in 1993 and Ph.D in Computer Science from Indian Statistical Institute, Calcutta in 1998. Currently she is a faculty member at Indian Statistical Institute, Calcutta, India. Dr. Bandyopadhyay is the first recipient of *Dr. Shanker Dayal Sharma Gold Medal* and *Institute Silver Medal* for being adjudged the best all round post graduate performer in IIT, Kharagpur in 1994. She has worked in Los Alamos National Laboratory, Los Alamos, USA, in 1997, as a graduate research assistant, in the University of New South Wales, Sydney, Australia, in 1999, as a post doctoral fellow, and in the Department of Computer Science and Engineering, University of Texas at Arlington, USA, in 2001 as a faculty and researcher. Dr. Bandyopadhyay received the Indian National Science Academy (INSA) and the Indian Science Congress Association (ISCA) *Young Scientist Awards* in 2000, as well as the Indian National Academy of Engineering (INAE) *Young Engineers' Award* in 2002. She has published over seventy articles in international journals, conference and workshop proceedings, edited books and journal special issues and served on the committees of several conferences. Her research interests include Pattern Recognition, Data Mining, Evolutionary and Soft Computation, Bioinformatics and Parallel & Distributed Systems.

Biographical Information of Dr. Sankar K. Pal

Sankar K. Pal is a Professor and Distinguished Scientist at the Indian Statistical Institute, Calcutta. He is also the Founding Head of Machine Intelligence Unit. He received the M. Tech. and Ph.D. degrees in Radio physics and Electronics in 1974 and 1979 respectively, from the University of Calcutta. In 1982 he received another Ph.D. in Electrical Engineering along with DIC from Imperial College, University of London. During 1986-87, he worked at the University of California, Berkeley and the University of Maryland, College Park, as a Fulbright Post-doctoral Visiting Fellow; and during 1990-92 & in 1994 at the NASA Johnson Space Center, Houston, Texas as an NRC Guest Investigator. He served as a Distinguished Visitor of IEEE Computer Society (USA) for the Asia-Pacific Region during 1997-99, and he held several visiting positions in Hong Kong and Australian universities during 1999-2002. Prof. Pal is a Fellow of the IEEE, USA, Third World Academy of Sciences, Italy, International Association for Pattern recognition, USA, and all the four National Academies for Science/Engineering in India. His research interests include Pattern Recognition and Machine Learning, Image Processing, Data Mining, Soft Computing, Neural Nets, Genetic Algorithms, Fuzzy Sets, and Rough Sets. He is a co-author of ten books and about three hundred research publications. He has received the 1990 S. S. Bhatnagar Prize (which is the most coveted award for a scientist in India), and many prestigious awards in India and abroad including the 1993 Jawaharlal Nehru Fellowship, 1993 Vikram Sarabhai Research Award, 1993 NASA Tech Brief Award (USA), 1994 IEEE Trans. Neural Networks Outstanding Paper Award (USA), 1995 NASA Patent Application Award (USA), 1997 IETE - Ram Lal Wadhwa Gold Medal, 1998 Om Bhasin Foundation Award, 1999 G. D. Birla Award for Scientific Research, 2000 Khwarizmi International Award (1st winner) from the Islamic Republic of Iran, the 2001 INSA-Syed Husain Zaheer Medal, and the FICCI Award 2000-2001 in Engineering and Technology. Prof. Pal served/serving as an Associate Editor and a Guest Editor of IEEE Trans. Pattern Analysis and Machine Intelligence, IEEE Trans. Neural Networks, IEEE Computer, Pattern Recognition Letters, Neurocomputing, Applied Intelligence, Information Sciences, Fuzzy Sets and Systems, Fundamenta Informaticae and Int. J. Computational Intelligence and Applications; and a Member, Executive Advisory Editorial Board, IEEE Trans. Fuzzy Systems, Int. Journal on Image and Graphics, and Int. Journal of Approximate Reasoning.

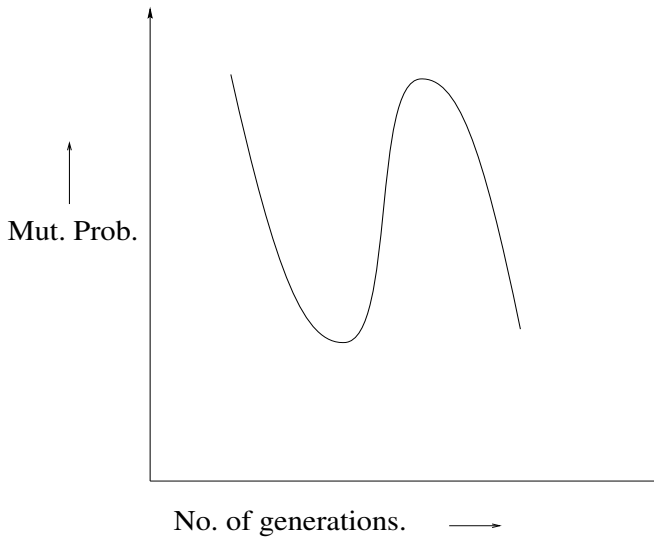

Fig. 1. Mutation probability graph

TABLE I
TRAINING RESULTS ON VARIOUS DATA SETS

	NSGAII		PAES		CEMOGA	
	H	Score%	H	Score%	H	Score%
Vowel	8	75.86	13	70.11	7	76.43
Iris	2	100	6	100	2	100
Cancer	3	98.82	11	98.23	1	96.47
Landsat	10	87.42	9	71.70	8	86.79
Mango	3	78.79	6	81.82	6	84.84
Crude Oil	6	100.00	11	95.45	4	100.00

TABLE II
TEST RESULTS ON VARIOUS DATA SETS (HERE CLACC STANDS FOR *Class Accuracy*)

	NSGAII			PAES			CEMOGA		
	ClAcc	Score%	Kappa	ClAcc	Score%	Kappa	ClAcc	Score%	Kappa
Vowel	0.06	74.48	0.664	0.07	69.19	0.635	0.15	79.08	0.731
Iris	0.83	94.67	0.919	0.86	94.67	0.919	0.89	96.00	0.939
Cancer	0.92	92.44	0.899	0.87	94.42	0.911	0.94	97.06	0.937
Landsat	0.25	80.85	0.753	0.21	72.29	0.641	0.29	84.13	0.795
Mango	0.00	60.24	0.328	0.00	63.85	0.367	0.28	66.26	0.418
Crude Oil	0.75	95.83	0.912	0.71	91.67	0.833	0.75	95.83	0.912

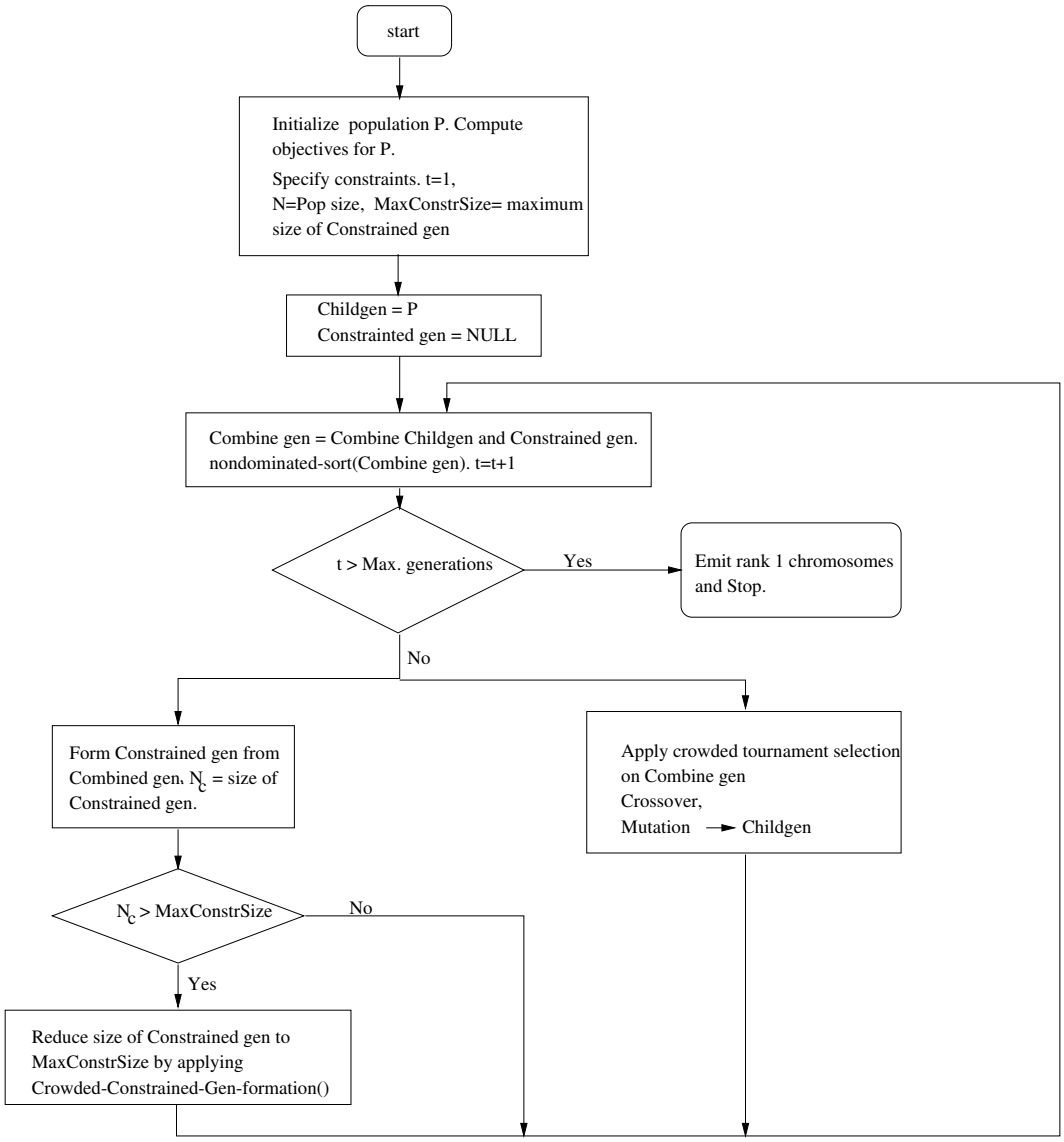

Fig. 2. Flowchart of CEMOGA-Classifier

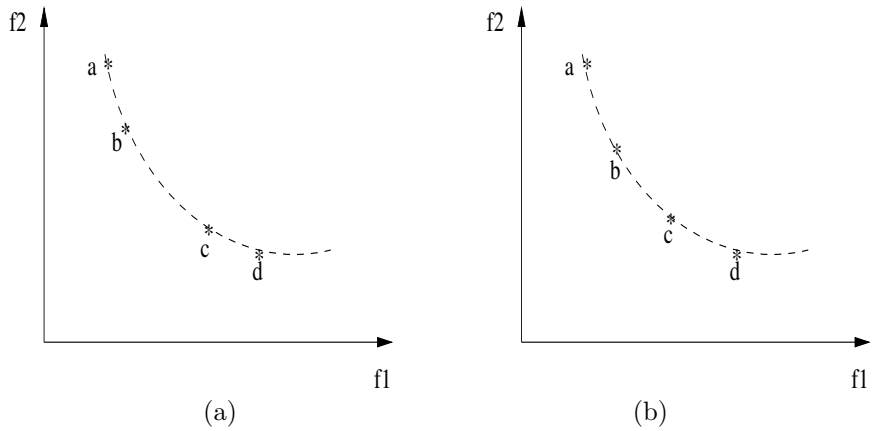

Fig. 3. Example of a set of non-dominated solutions

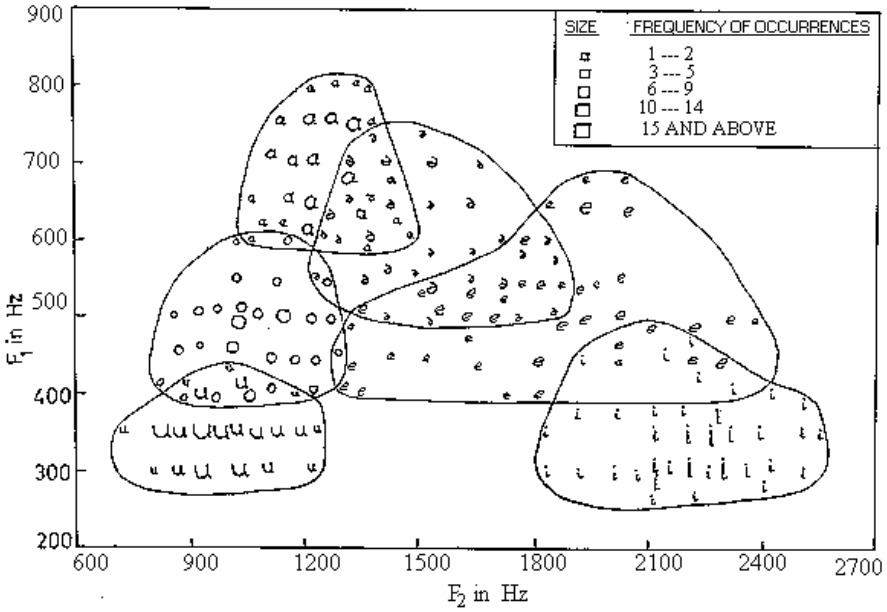

Fig. 4. Vowel data set

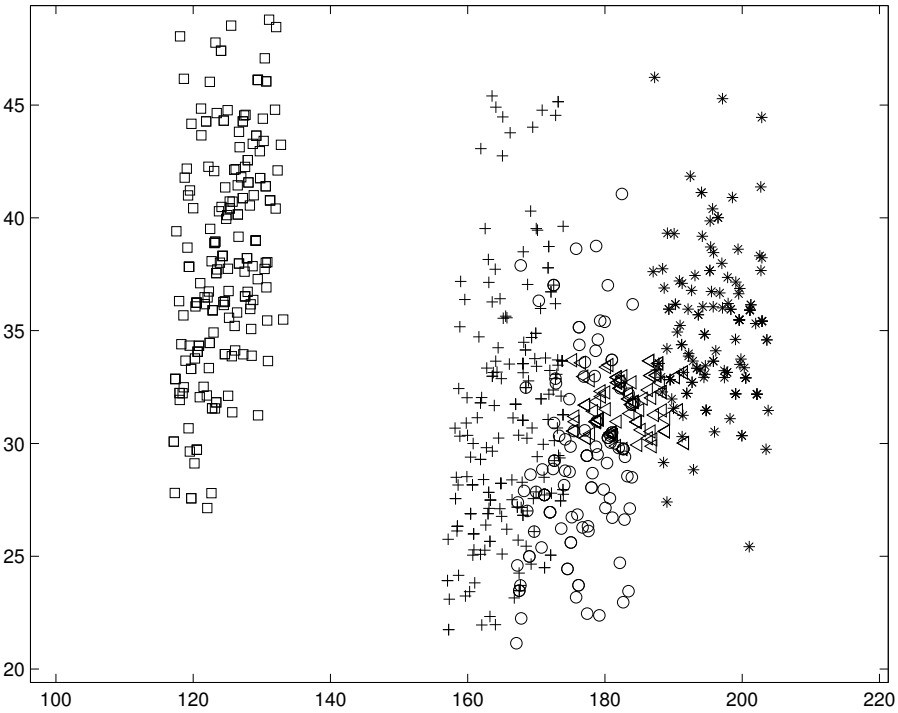

Fig. 5. Landsat data set ('□': Vegetation, '+': Black Phillite, 'o': Romapahari Granite, '◁': Alluvium, '*': Manda Granite respectively)

TABLE III
CLASSWISE PERCENTAGE RECOGNITION SCORES FOR MANGO USING THE GENETIC CLASSIFIERS

Class	Recognition Scores (%) Using		
	NSGAII	PAES	CEMOGA
C1	82.93	90.24	90.24
C2	66.67	65.21	50.00
C3	0.0	0.0	33.33

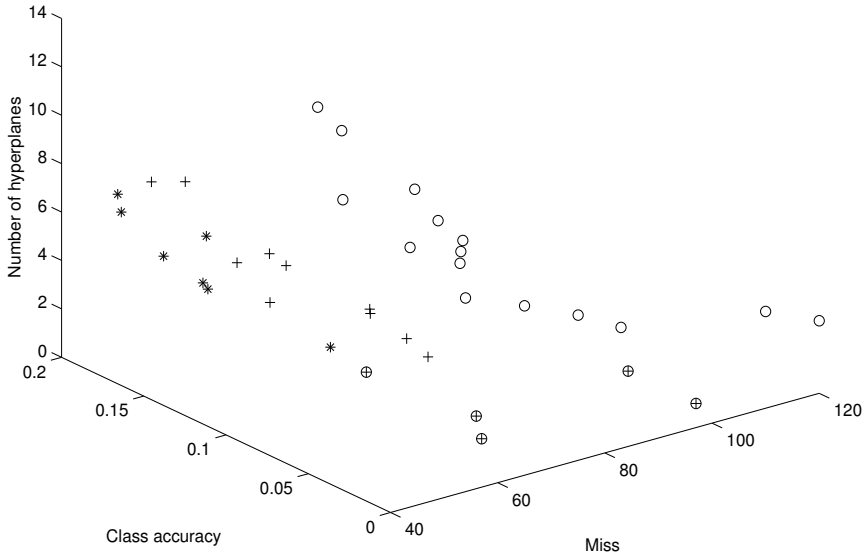

Fig. 6. Non-dominated solutions obtained by *CEMOGA-Classifier* (denoted by '*'), *PAES-Classifier* (denoted by 'o') and *NSGAII-Classifier* (denoted by '+') for the *Vowel* data set. Actual rank 1 solutions for *NSGAII-Classifier* are encircled.

TABLE IV
PURITY MEASURE ON VARIOUS DATA SETS

	NSGAII		PAES		CEMOGA	
	# sol.	Purity	# sol.	Purity	# sol.	Purity
Vowel	15	0.33	15	0.00	7	1.00
Iris	2	1.00	4	0.00	1	1.00
Cancer	3	0.67	6	0.00	2	1.00
Landsat	11	0.54	11	0.09	3	1.00
Mango	2	1.00	3	0.33	4	1.00
Crude Oil	8	0.37	7	0.00	3	1.00

TABLE V
MINIMAL SPACING (S_m) MEASURE ON VARIOUS DATA SETS

	NSGAII		PAES		CEMOGA	
	S_m	S_m (R1)	S_m	S_m (R1)	S_m	S_m (R1)
Vowel	0.088	0.117	0.084	-	0.042	0.042
Iris	-	-	0.474	-	-	-
Cancer	0.433	-	2.085	-	-	-
Landsat	0.095	0.151	0.072	-	0.018	0.018
Mango	-	-	0.089	-	0.056	0.056
Crude Oil	0.061	0.25	0.119	-	0.173	0.173