PINSA, **67, A,** No. 2, March 2001, pp. 295-314

SUPERVISED PATTERN CLASSIFICATION BY SURFACE FITTING WITH GENETIC ALGORITHMS

S BANDYOPADHYAY, C A MURTHY AND S K PAL

Machine Intelligence Unit, Indian Statistical Institute, 203 BT Road, Calcutta - 700035 (India)

(Received 01 March 2000; Accepted 16 October 2000)

The principle of placing hyperplanes to model the class boundaries for pattern recognition problems using the searching capabilities of genetic algorithms (GAs) is explained. Various classifiers using fixed string length GAs, variable string length GAs and GAs with chromosome differentiation are described. The superiority of these classifiers over the Bayes classifier (with assumption of multivariate normal distribution), k-NN rule and multilayer perceptron is demonstrated for both speech and remotely sensed data. Some of the results reported here are taken from the existing literature.

Key Words: Genetic Algorithms; Boundary Approximation; Variable String Length; Restrictive Mating; Speech Recognition; Satellite Image Classification; Bayes Classifier; k-NN Rule; Multilayer Perceptron

1 Introduction

Genetic algorithms (GAs)[1-14] are randomized search and optimization techniques guided by the principles of evolution and natural genetic. The term was first mentioned by Bagley[15] when he devised a genetic algorithm based game playing program using some commonly used operators. He found that the GA was insensitive to the game non-linearity, and performed well over a range of environments. It was then with the pioneering work of Holland[11] that GAs were firmly established as an effective search and optimization strategy.

GAs mimic some of the processes observed in natural evolution, which included operations like selection, crossover and mutation. They perform multimodal search in complex landscapes and provide near optimal solution for objective of fitness function of an optimization problem. They are efficient, adaptive and robust search processes, with a large amount of implicit parallelism[9,11] Genetic algorithms are gradually finding widespread applications during the past decade in solving problems requiring efficient and effective search, in business, scientific and engineering circles[1,5,8,10,13,16,17]. Some of the applications of GAs, so far being made, include pattern classification and feature selection[16-18], image processing and scene recognition[19-22] rule generation and classifier systems[23-28] neural network design[29-36] scheduling problems[37,38] VLSI design[39], path planning[40] and the traveling salesman problem[41-43], graph colouring[3], and numerical optimization[44]. Moreover, several researchers are actively engaged in developing enhanced and more effective genetic operators and models, and analyzing their performance for different applications. Some such attempts are described below.

The issue of convergence of GAs to the globally optimal solution has been pursued in ref. [45], where GAs are again modelled as Markov chains having a finite number of states. A state is represented by a population together with a potential string. Irrespective of the choice of initial population, GAs have been proved to converge to the optimal string for infinite number of iterations, provided the conventional mutation operation is incorporated. Murthy *et al.*[46] have provided a stopping criterion, called ε-optimal stopping time, for the elitist model of the GAs. Subsequently, they have derved the ε-optimal stopping time for GAs with elitism under a 'practically valid assumption'.

An attempt to incorporate the ancestors influence into the fitness of individual chromosomes has been made in ref. [47]. This is based on the observations in nature where an individual is not an independent entity, but is highly influenced by the environment.

e-mail: {sanghami,murthy,sankar}@www.isical.ac.in

Ghosh *et al.*[48] have incorporated the concept of aging of individuals for measuring their suitability for participation in genetic operations, by combining both the functional value and the age of an individual for computing its effective fitness. Results have shown that this scheme provides enhanced performance and maintains more diversity in the population.

Bhandari *et al.*[49] have proposed a new mutation operator known as *directed mutation* which follows from the concept of induced mutation in biological systems[50]. This operation uses the information acquired in the previous generations rather than probabilistic decision rules. In certain environments, directed mutation will deterministically introduce a new point in the population. The new point is directed (guided) by the solutions obtained earlier, and therefore the technique is called *directed mutation*.

In ref. [33], Pal and Bhandari incorporated GAs to find out the optimal set of weights (biases) in a layered network. Weighted mean square error over the training examples has been used as the fitness measure. They introduced a new concept of selection, called *non-linear selection*, which enhances genetic homogeneity of the population and speeds up searching. Implementation results on both linearly separable and non-linearly separable pattern sets are also reported.

An attempt has also been made for evolving architectures of Hopfield type optimum neural networks for extracting object regions from gray images using GAs[51], where each binary chromosome represents a network architecture. The presence (or absence) of connectivity between neurons is represented by 1 (or 0). The proposed GA based technique has been able to evolve network architectures whose connectivity is about two-third of the requirement of the corresponding fixed fully connected ones in order to produce comparable segmented output. The optimized networks have been found to be more noise independent. Other attempts for evolving the architecure of neural networks using GAs can be found in refs. [36, 52, 53].

Many tasks involved in the process of recognizing a pattern need appropriate parameter selection and efficient search in complex and large spaces in order to attain optimal solutions. This makes the process not only computationally intensive, but also leads to a possibility of losing the exact solution. Therefore, the application of GAs for solving certain problems of pattern recognition, that require optimization of computation requirements, and robust, fast and close approximate solution, seems appropriate and natural. Additionally, the existence of the proof of convergence of GAs to the global optimal solution as the number of iterations goes to infinity[45], further strengthens the theoretical basis of its use in search problems. Significance of GAs to pattern recognition and image processing problems is adequately demonstrated in refs. [16, 17, 53-55]. Some of the investigations are mentioned below.

A method for determining the optimal enhancement operator for both bimodal and multimodal images is described by Pal *et al.* in ref. [21]. The algorithm does not need iterative visual interaction and prior knowledge of image statistics for this purpose. The fuzziness measures are use as fitness function.

Selection of a subset of principal components for classification using GAs is made in ref. [56]. Since the search space depends on the product of the number of classes and the number of original features, this selection process by conventional means may be computationally very expensive. Results on two data sets with small and large cardinalates are presented.

Murthy and Chowdhury[57] have used GAs for finding optimal clusters, without the need for searching all possible clusters. The experimental results show that the GA based scheme may improve the final output of the K-means algorithm[58] where an improvement is possible.

Another hybridization of the K-means algorithm with GAs (called, GKA) for partitional clustering is reported in ref. [59], where the superiority of GKA over other algorithms is demonstrated. The GKA is applied for codebook design that are used in image and speech coding. A class of representation schemes, called Voronoi Net-works, has also been proposed in ref. [59] and a new heuristic learning algorithm for them, called supervised K-means algorithm, is formulated.

One of the important and natural applications of GAs for supervised pattern classification is to search and appropriately place a number of surfaces in the feature space such that the decision boundary of a given data set is closely approximated. Attempts in this direction can be found in refs. [60, 61]. In ref. [59], Srikanth *et al.* described a genetic algorithmic approach to pattern classification, both crisp and fuzzy, where clusters in pattern space are approximated by

ellipsoids. A variable number of ellipsoids is searched for, which collectively classify a set of objects by minimizing a criteria based on the number of misclassified points and the fuzzy distance of a patterns from the surface of the ellipsoid.

The present article provides, in this direction, some key features of the results of investigation that has been carried out in Machine Intelligence Unit of Indian Statistical Institute, Calcutta, under the project Soft Computing in Pattern Recognition. This includes the *GA-classifier* (where the decision boundary is approximated by a fixed number of hyperplanes), the *VGA-classifier* (where variable string length GAs are used to approximate the decision boundary by a variable number of hyperplanes) and the *GACD-classifier* (where the concept of chromosome differentiation incorporated in designing GA based classifiers), along with some of their real life applications. Some of the results are taken from the existing literature.

A distinguishing feature of the above approach is that the boundaries need to be generated explicitly for making decisions. This is unlike the conventional methods or the multilayered perceptron (MLP) based approaches, where the generation of boundaries is a consequence of the respective decision making processes.

2 Overview of Genetic Algorithms

GAs are modelled on the principles of natural genetic systems, where the genetic information of each individual or potential solution is encoded in structures called *chromosomes*. They use some domain or problem dependent knowledge for directing the search in more promising areas: this is known as the *fitness function*. Each individual or chromosome has an associated fitness function, which indicates its degrees of goodness with respect to the solution it represents. Various biologically inspired operators like *selection, crossover* and *mutation* are applied on the chromosomes to yield potentially better solutions.

Since a GA works simultaneously on a set of coded solutions it has very little chance of getting stuck at a local optimum when used as an optimization technique. Again, the search space need not be continuous, and no auxiliary information, like derivative of the optimizing function, is required. Moreover, the resolution of the possible search space is increased by operating on coded (possible) solutions and not on the solutions themselves.

A schematic diagram of the basic structure of a genetic algorithm is shown is Fig. 1.

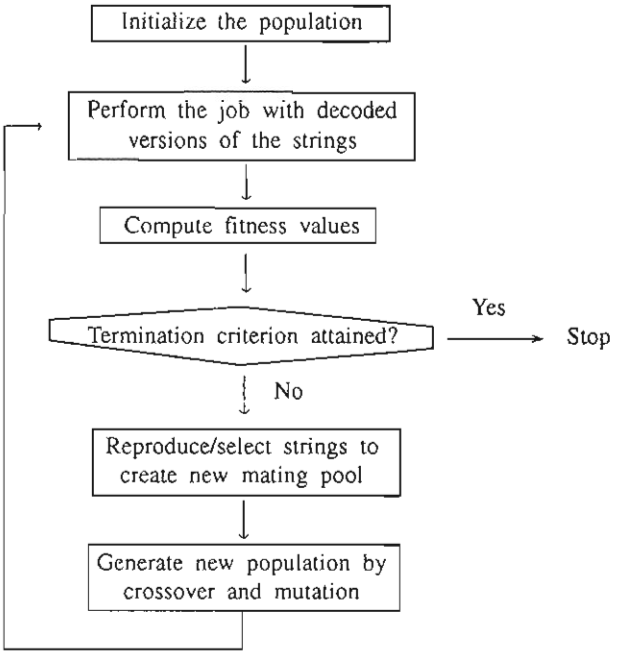

Fig. 1 Basic steps of genetic algorithm

The evolution starts from a set of chromosomes (representing a potential solution set for the function to be optimized) and proceeds from generation to generation through genetic operations. GAs require only a suitable objective function, which is a mapping from the chromosomal space to the solution space, in order to evaluate the suitability or fitness of the derived solutions.

To solve an optimization problem, GAs start with the chromosomal representation of a parameter set. The parameter set is to be coded as a finite length string over an alphabet of finite length. Usually, the chromosomes are strings of 0's and 1's. A set of such chromosomes in a generation is called a *population*, the size of which may be constant or may vary from one generation to another. A common practice is to choose the initial population randomly.

The fitness/objective function associated with a chromosome is chosen depending on the problem to be solved, in such a way that the strings (possible solutions) representing good points in the search space have high fitness values. This is the only information (also known as the payoff information) that GAs use while searching for possible solutions.

Subsequently, the selection/reproduction process copies individual strings (called parent chromosomes) into a tentative new population (known as mating pool) for genetic operations. The number of copies that an individual receives for the next generation is usually taken to be directly proportional to its fitness value; thereby mimicking the natural selection procedure to some extent. This scheme is commonly called the *proportional selection scheme*. A commonly used strategy known as the *elitist selection*[62] is adopted in GAs, thereby providing an *elitist GA* (EGA), where the best chromosome of the current generation is retained in the next generation.

The other two frequently used genetic operators applied on the population of chromosomes are crossover and mutation. The main purpose of crossover is to exchange information between randomly selected parent chromosomes by recombining parts of their corresponding strings. It recombines genetic material of two parent chromosomes to produce offspring for the next generation. *Single point crossover* is one of the most commonly used schemes.

The main aim of mutation is to introduce genetic diversity into the population. Sometimes, it helps to regain the information lost in earlier generations. In case of binary representation it negates the bit value and is known as bit mutation. Like natural genetic systems, mutation is GAs is usually performed occasionally. Here a random bit position of a randomly selected string is replaced by another character from the alphabet.

As shown in Fig. 1, cycle of selection crossover and mutation is repeated a number of times till one of the following occurs :

1. The average fitness value of a population becomes more or less constant over a specified number of generations,
2. A desired objective function value is attained by at least one string in the population,
3. The number of generations (or iterations) is greater than some threshold value.

3 Description of the Genetic Classifers

A) GA-Classifier: GA Based Classifier Using Fixed Number of Hyperplanes

The GA-*Classifier*[63] attempts to place H hyperplans in the feature space appropriately such that the number of misclassified training points is minimized. From elementary geometry, the equation of a hyperplane in N dimensional space $(X_1–X_2– \ldots X_N)$ is given by

$$x_N \cos \alpha_N + \beta_{N-1} \sin\alpha_{N-1} = d \quad \ldots. \quad (1)$$

where $\beta_{N-1} = x_{N-1} \cos \alpha_{N-2} + \beta_{N-2} \sin \alpha_{N-2}$

$$\beta_{N-2} = x_{N-2} \cos \alpha_{N-3} + \beta_{N-3} \sin \alpha_{N-3}$$

$$\vdots$$

$$\beta_1 = x_1 \cos \alpha_0 + \beta_0 \sin \alpha_0$$

The various parameters are as follows :

X_i : the i th feature of the training points.

$(x_1, x_2, \ldots, x_N)$: a point on the hyperplane

α_{N-1} : the angle that the unit normal to the hyperplane makes with the X_N axis.

α_{N-2} : the angle that the projection of the normal in the $(X_1–X_2– \ldots, – X_{N-1})$ space makes with the X_{N-1} axis.

$\vdots$

α_1 : the angle that the projection of the normal in the $(X_1–X_2)$ plane makes with the X_2 axis.

α_0 : the angle that the projection of the normal in the (X_1) plane makes with the X_1 axis = 0. Hence, $\beta_0 \sin \alpha_0 = 0$.

d : the perpendicular distance of the hyperplane from the origin.

Thus the N tuple $< \alpha_1, \alpha_2, \ldots, \alpha_{N-1}, d >$ specifies a hyperplane in N dimensional space.

Each angle α_j, $j = 1, 2, \ldots, N–1$ is allowed to vary in the range of 0 to 2π. If b_1 bits are used to represent an angle, then the possible values of α_j are

$$0, \delta * 2\pi, 2\delta * 2\pi, 3\delta * 2\pi, \ldots, (2^{b1}–1)\ \delta * 2\pi$$

where $\delta = \frac{1}{2^{b1}}$. Consequently, if the b_1 bits contain a binary string having the decimal value v_1, then the angle is given by $v_1 * \delta * 2\pi$.

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 (x_i^{min}, x_i^{max}) be the minimum and maximum values of feature X_i as obtained from the training points. Then the vertices of the enclosing hyper rectangle are given by

$$(x_1^{ch1}, \ x_1^{ch2}, \ \ldots, \ x_N^{chN})$$

where each ch_i, $i = 1,2, \ldots, N$ can be either *max or min*.

(Note that there will be 2^N vertices.) Let *diag* be the length of the diagonal of this hyper rectangle given by

$$diag = \sqrt{\left(x_1^{max} - x_1^{min}\right)^2 + \left(x_2^{max} - x_2^{min}\right)^2 + \ \ldots \ + \left(x_N^{max} - x_N^{min}\right)^2}$$

A hyperplane is designated as the *base hyperplane* with respect to a given orientation (*i.e.*, for some $\alpha_1, \alpha_2, \ldots, \alpha_{N-1}$) if

i) it has the same orientation

ii) it passes through one of the vertices of the enclosing rectangle

iii) 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 $d_{min} + \frac{diag}{2^{b_2}} * v_2$.

Thus each chromosome is of a fixed length of $l = H\,((N-1) * b_1 + b_2)$, where H denotes the number of hyperplanes. These are initially generated randomly for a population of size *Pop*.

Using the parameters of the hyperplanes encoded in a chromosome, the region in which each training pattern lies is determined based on eq (1). A region is said to provide the demarcation for class i, if among the points that lie in this region, majority belong to class i. Other points that lie in this region are considered to be misclassified. The misclassifications associated with all the regions (for these H hyperplanes) are summed up to provide the total misclassification, *miss*, for the string. Its fitness is defined as $(n - miss)$, where n is the size of the training data.

After computing the fitness, the genetic operators of selection, crossover and mutation are applied[9] to generate a new population of chromosomes. Elitism is incorporated in the process for preserving the best candidate found so far. Fitness computation followed by genetic operations are executed for a fixed number of generations at the end of which the best chromosome provides the set of hyperplanes constituting the final decision boundary. The flowchart for the *GA-classifier* is given in Fig. 2.

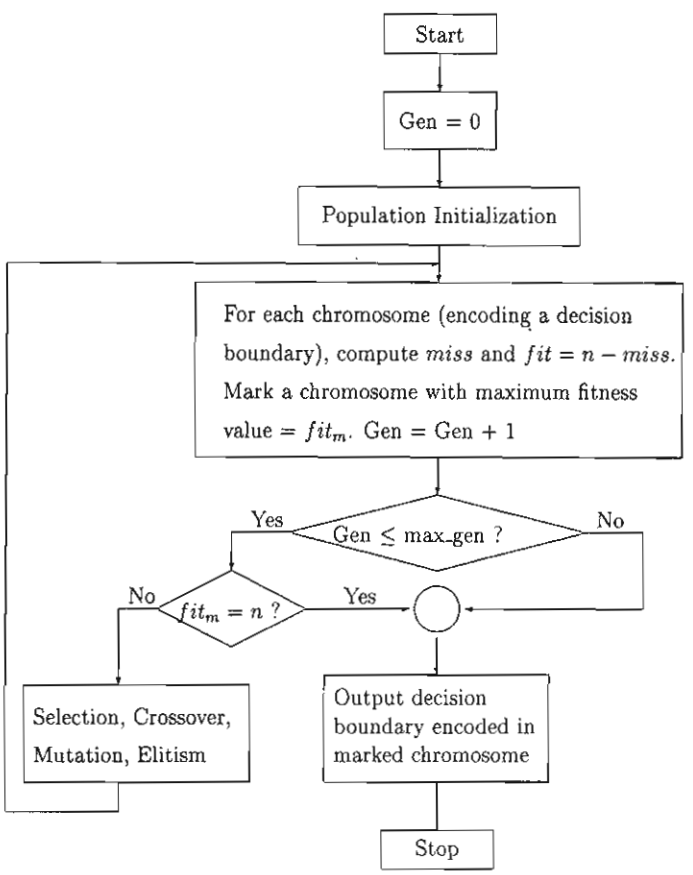

Fig. 2 Flowchart for the *GA-classifier*

B) Determination of Optimal H : VGA-Classifier

Since it is very difficult to estimate a proper value of H, the *GA-classifier* often suffered from the problem of over fitting of the data set, resulting from a conservative estimate of H. This also led to the presence of redundant hyperplanes in the final decision boundary. In order to overcome this limitation, the concept variable string lengths in GAs[64], encoding the parameters of a variable number of hyperplanes, was incorporated in the *GA- classifier*, thereby providing the *VGA-classifier*[65].

In the *VGA-classifier*, the chromosomes are represented by strings of 1, 0 and # (don't care), encoding the parameters of variable number of hyperplanes. Let H_{max} represent the maximum number of hyperplanes that may be required to model the decision boundary of a given data set. It is specified *a priori.*

Fitness Computation

For each string i encoding H_i hyperplanes, the number of misclassified points $miss_i$, is found as in the case for *GA-classifer*. If n is the size of the training data, then the fitness of the ith string, fit_i, is defined as

$$fit_i = (n - miss_i) - \alpha H_i$$

where $\alpha = \frac{1}{H_{max}}$ and H_i is the number of hyerplanes encoded in the string. A string with zero hyperplane is defined to have zero fitness. Maximization of the fitness function ensures the minimization of, primarily, the number of misclassified points and then the number of hyperplanes.

Genetic Operators

Since the strings have variable length, the operators crossover and mutation were newly defined as follows.

Crossover

Two strings, i and j having length l_i and l_j respectively are selected from the mating pool. Let $l_i \leq l_j$. Then string i is padded with #s so as to make the two lengths equal. Conventional crossover like single point crossover, two point crossover[9] is now performed over these two strings with probability μ_c. The following two cases may now arise :

1) All the hyperplanes in the offspring are complete (A hyperplane is a string is called *complete* if all the bits corresponding to it are either defined (i.e., 0s and 1s) or #s. Otherwise it is incomplete).
2) Some hyperplanes are incomplete.

In the second case let u = number of defined bits (either 0 or 1) and t = total number of bits per hyperplane. Then, for each incomplete hyperplane, all the #s are set to defined bits (either 0 or 1 randomly) with probability $\frac{u}{t}$. In case this is not permitted, all the defined bits are set to #. Thus each hyperplane in the string becomes complete. Subsequently, the string is rearranged so that all the #s are pushed to the end.

Mutation

In order to introduce greater flexibility in the method, the mutation operator is defined is such a way that it can both increase and decrease the string length. For this, the strings are padded with #s such that the resultant length becomes equal to l_{max}. Now for each defined bit position, it is determined whether conventional mutation[9] can be applied or not with probability μ_m. Otherwise, the position is set to # with probability μ_{m1}. Each undefined position is set to a defined bit (randomly chosen) according to another mutation probability μ_{m2}.

Note that mutation may result in some incomplete hyperplanes, and these are handled in a manner, as done for crossover operation. Also, mutation may yield strings having all #s indicating that no hyperplanes are encoded in it. Consequently, this string will have fitness = 0 and will be automatically eliminated during selection.

C) Theoretical Studies of the Genetic Classifiers

Theoretical analyses of the above mentioned GA based classifiers show that for infinitely large number of iterations it will provide the minimum misclassification error during training; at the same time the number of hyperplanes required to model the decision boundary for providing the minimum number of misclassified points will also be the minimum.

It is known from the literature that Bayes classifier[58] is the best possible classifier if the class conditional densities and *a priori* probabilities are known. No classifier can provide better performance than Bayes classifier under such conditions. In practice, it is difficult to use Bayes classifier because the class conditional densities and *a priori* probabilities may not be known. Hence new classifiers are devised and their performances are compared to that of the

Bayes classifier. The desirable property of any classifier is that its performance should approximate or approach that of the Bayes classifier under limiting conditions. There are many ways in which the performance of a classifier is compared to that of the Bayer classifier. One such way is to investigate the behavior of the error rate (defined as the ratio of the number of misclassified points to the size of the data set) as the size of the training data goes to infinity, and check whether the limiting error rate is equal to the Bayers error probability. Such an investigation[65] establishes that this is true for the aforesaid genetic classifiers when the number of iterations goes to infinity. In other words, the decision boundary provided by the genetic classifiers approaches the Bayes decision boundary as the number of training data points and the number of iterations approaches infinity.

D) Experimental Results

This section has two parts. In the first part, some experimental results are presented for the genetic classifiers on *Vowel* data set. This includes a description of the data set, variation of the recognition scores of the *GA- classifier* during testing for different values of H, performance of the *VGA-classifier* and its comparison to the Bayes maximum likelihood classifier and k- NN rule. In the second part, the above genetic classifiers are used for pixel classification of satellite images of parts of the cities of Calcutta and Bombay for locating different landcover regions.

For the GA based classifiers the numbers of bits used to represent an angle and the perpendicular distance are 8 and 16 respectively. *Roulette wheel* selection is adopted to implement *the proportional selection strategy, Single point* crossover is applied with a fixed crossover probability (μ_c) value of 0.8. The mutation operation is performed on a bit by bit basis for a varying mutation probability value (μ_m) in the range [0.015, 0.333]. The form of the variation of μ_m with the number of generations is shown in Fig. 3. The range is divided into eight equispaced values. μ_m is slowly decreased in steps from 0.333 to 0.015, and then increased again. This ensures that initially, a random search is performed through the feature space. The randomness is gradually decreased with the passing of generations so that now the algorithm performs a detailed search in the vicinity of promising solutions obtained so far. In spite of this, the algorithm may still get stuck at a local optimum. This problem is overcome by increasing the mutation probability to a high value, thereby making the search more random once again. 100 and 200 iterations are executed with each value of μ_m for the *GA-classifier* and the *VGA-classifier* respectively. (Note that since the search space is larger for the *VGA-classifier*, it is allowed more time to execute). The algorithm is terminated if the population contains at least one string with no misclassified points. Otherwise, the algorithm is executed for a prespecified number of generations.

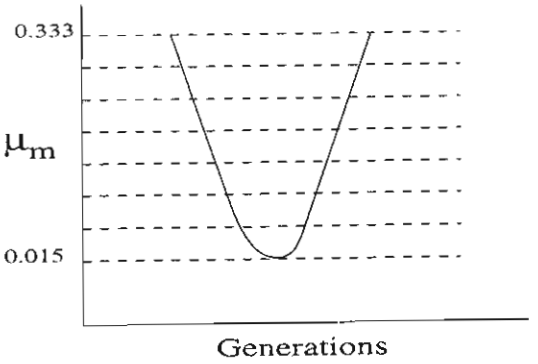

Fig. 3 Variation of mutation probability with the number of generations

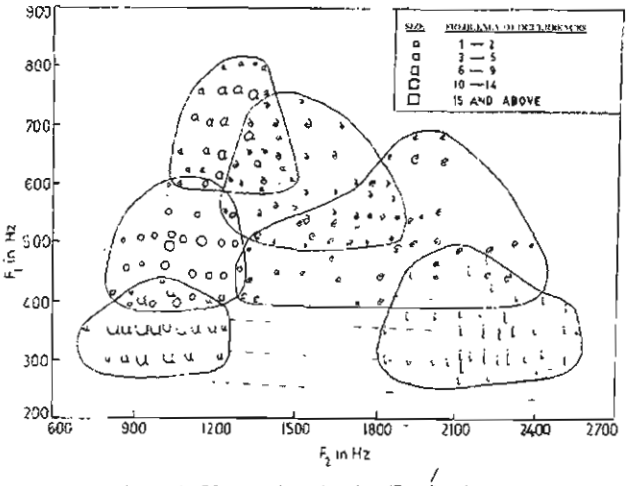

Fig. 4 Vowel data in the F_1–F_2 plane

D.1 Results on the Vowel Data

Vowel data consists of 871 Indian Telugu vowel sounds[67]. These were uttered by three male speakers in the age group of 30-35 years. The data set has three features F_1, F_2 and F_3, corresponding to the first, second and third vowel formant frequencies, and six classes $\{\delta,a,i,u,e,o\}$. Fig. 4 shows the distribution of the six classes in the F_1–F_2 plan. (It

is known[67] that these two features are more important in characterizing the classes than F_3). Note that the boundaries of the classes are seen to be very ill-defined and overlapping. The scores provided here are the average values obtained over five different runs of the algorithms.

Table I presents the test recognition scores of the *GA-classifier* for different values of H. The scores are found to improve with the value of H upto H =7 which provides the maximum score. Low values of H *viz.*, 2 and 3 are seen to provide quite low recognition scores indicating that they are insufficient for modelling the overlapping class boundaries appropriately. Even H = 4 is not a proper choice since in this case one class was not recognized at all (class *a*). Interestingly, although it was found that the recognition scores during training of the *GA-classifier* consistently improved with the increase of H from 2 to 8, a decrease in the test recognition score is observed from H = 7 to H = 8. The reason for this is that as H is increased upto a certain point (from 2 to 7), the *GA- classifier* is able to surround the data points more easily during training thereby providing improved scores. However increasing H beyond a certain point (in this case 7) results in overfitting of the training data points at the cost of reduced generalization capability. Therefore, although the training scores improve even further, the recognition scores during testing degrades.

Table I

Variation of overall recognition scores (%) during testing with H for Vowel data with perc = 10

Number of hyperplanes	Overall recognition score
8	74.56
7	74.68
6	71.99
5	71.37
4	69.21
3	57.50
2	53.30

Table II shows the number of hyperplanes H_{VGA} as determined automatically by the *VGA-classifier* for modelling the class boundaries of *Vowel*. Two different values of H_{max}, *viz.*, 6 and 10, are used for this purpose. The overall recognition scores obtained during testing of the *VGA-classifier* along with their comparison with those of the corresponding fixed length version (*i.e., GA-classifier* with H = 6 and 10) are also shown. The purpose of this exercise is to compare the performance of the *VGA-classifier* and *GA-classifier* starting with the same number of hyperplanes, i.e., H_{max} for *VGA-classifier* and H for *GA-classifier.*

Table II

H_{VGA} and the comparative overall recognition scores (%) during testing (when 10% of the data set is used for training and the remaining 90% for testing)

VGA-classifier			Score for *GA-classifier*
H_{max}	H_{VGA}	Score(%)	when $H = H_{max}$ (%)
6	6	71.19	71.99
10	6	73.66	69.21

It is found from the table that the *VGA-classifier* automatically reduces the number of hyperplanes to 6 when it was initiated with a larger value (=10). When initiating the algorithm with H_{max} = 6, it was not able to eliminate any hyperplane. Interestingly, although its recognition score on the test data set is found to be higher than that of the *GA-classifier* for H_{max} = 10 (where finally six hyperplanes are utilized), this is not the case for H_{max} = 6. This may be due to the fact that with ten hyperplanes the *VGA-classifier* has more flexibility of placing a smaller number of hyperplanes appropriately than in the case when H_{max} = 6. Therefore on termination of the algorithm after a fixed number of generations, the former is able to better approximate the decision boundary than the latter. However, it may be noted that the recognition scores may have improved further if more iterations of the classifier are executed.

Regarding the time taken for training of the classifiers, the *VGA-classifier* is found to take longer than the *GA-classifier* even for a fixed number of generations. As is mentioned earlier, this is because the search space is larger for the *VGA-classifier* (where the number of hyperplanes varies in the range [1, H_{max}]) than for *GA-classifier* (where the number of hyperplanes is fixed).

For the purpose of comparing the performance of the *VGA-classifier* we have used Bayes maximum likelihood classifier (which is well known for discriminating over-lapping classes), and k-NN classifier and the multilayer perceptron (both of which are well known for discriminating non-overlapping, non-linear regions by generating piecewise linear boundaries).

Recognition scores on the 90% test data when the remaining 10% of the data set was used for training

were 77.73%, 70.35% and 68.48% for Bayes maximum likelihood classifier (with assumption of normal distribution, and estimation of the covariance matrices and *a priori* probabilities from the data set), k-NN rule (for k= $\sqrt{n}$) and MLP (with two hidden layers and 20 hidden nodes per layer) respectively. As can be found from comparison with the results in Table II, Bayes Maximum likelihood classifier performs the best for *Vowel* (which conforms to earlier findings made in ref. 67), followed by the score of the *VGA-classifier* for H_{max} = 10. MLP is found to perform the poorest for this data. Detailed results considering other parameters are provided in ref. [65].

Table III provides a comparison of the performance of our concept of using variable string length in GAs[65] with that used in ref. [60]. Before providing the results, let us briefly describe the method of incorporating variable string lengths in GAs as in ref. [60].

The initial population is created randomly such that each string encodes the parameters of only one hyperplane. The fitness of a string is characterized by just the number of training points it classifies correctly, irrespective of the number of hyperplanes encoded in it. Among the genetic operators, traditional selection and mutation are used. A new form of crossover, called *modulo crossover*, is used which keeps the sum of the lengths of the two chromosomes constant both before and after crossover.

Two other operators are used in conjunction with the *modulo crossover* for the purpose of faster recombination and juxtaposition. These are the *insertion* and *deletion* operators. During *insertion*, a portion of the genetic material from one chromosome is inserted at a random *insert-location* in the other chromosome. Conversely, during *deletion*, a portion of a chromosome is deleted to result in a shorter chromosome.

Table III shows the comparative overall recognition scores during both training and testing of the *VGA-classifier* (H_{max} = 10) for *Vowel* when 10% and 50% of the data set are used for training. For keeping parity, the VGA of Srikanth *et al.*[60] is implemented such that no more than 10 hyperplanes are used for modelling the decision boundary of the data sets. The table also shows the number of hyperplanes. H_{VGA}, generated by the two methods for one particular run. Since the method in ref. [60] does not take care of the minimization of the number of hyperplanes while maximizing the fitness function, the H_{VGA} is usually higher than that of our method.

As is evident from the tables, the performance of the classifier during training is better for the VGA in ref. [60] than the one in ref. [65]. The former, in general, uses more hyperplanes (of which many were found to be redundant on investigation), which results in an increase in the execution time. From the training performance, it appears that the operators used by Srikanth *et al.*, are better able to recombine the subsolution blocks into appropriate larger bloks. However this is seen, in general, to result in comparatively poorer scores during testing. To consider a typical example in one of the cases for the vowel data set when 10% data is used for training, 10 hyperplanes were used to provide a training recognition score of 97. 47% while the recognition score during testing reduced to 68.95%.

Table III

Comparative classification performance of VGA-classifier for H_{max} = 10 using two types of variable string lengths for different percentages of the training data

	VGA[65]			VGA[60]		
Perc	Training score (%)	Test score (%)	H_{VGA}	Training score (%)	Test score (%)	H_{VGA}
10%	80.00	73.66	6	97.36	70.22	9
50%	79.73	78.26	6	85.48	78.37	9

D.2 Pixel Classification of Spot and IRS Images

In this section, the utility of the genetic classifiers for classification of pixels for partitioning different landcover regions in satellite images is investigated. Note that satellite images usually have a large number of classes with overlapping and nonlinear class boundaries. Fig. 5 shows, as a typical example, the complexity in scatter plot of 932 points belonging to seven classes which are taken from the *SPOT* image of a part of the city of Calcutta. Therefore, for appropriate modeling of such non-linear and overlapping class boundaries, the utility of an efficient search technique like GAs is evident. Moreover, it is desirable that the search technique does not need to assume any particular distribution of the data set and/or class a *priori* probabilities.

SPOT Image of a Part of Calcutta

The image considered in this experiment has three bands. These are :

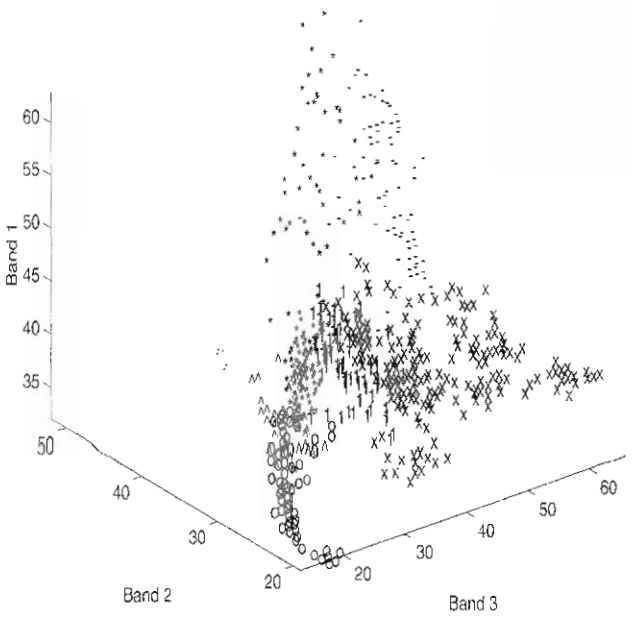

Fig. 5 Scatter plot for the training points in Calcutta image

Band 1– green band of wavelength 0.50-0.59 μm,

Band 2– red band of wavelength 0.61-0.68 μm, and

Band 3– near infra red band of wavelength 0.79-0.89 μm.

The training set comprises 932 points belonging to seven classes, with three features corresponding to the above mentioned bands. The seven classes are *turbid water* (TW), *pond water* (PW), *concrete* (Concr.), *vegetation* (Veg), *habitation* (Hab), *open space* (OS) and *roads* (including bridges) (B/R). Some important landcovers of Calcutta can be identified, from a knowledge about the area, more easily in Band 3 of the image (Fig. 6 shows the image with 75% stretching to make it more prominent). These are the following : The prominent black stretch across the figure is the river *Hooghly*. Portions of a bridge (referred to as the *second bridge*), which was under construction when the picture was taken, protrude into the *Hooghly* near its bend around the center of the image. There are two distinct black, elongated patches below the river, on the left side of the image. These are water bodies, the one to the left being *Garden Reach lake* and the one to the right being *Khidirpore dockyard*. Just to the right of these water bodies, there is a very thin line, starting from the right bank of the river, and going to the bottom edge of the picture. This is a canal called the *Talis nala*. Above the *Talis nala*, on the right side of the picture, there is a triangular patch, the *race course*, On the top, right hand side of the image, there is a thin line, stretching from the top edge, and ending on the middle, left edge. This is the *Beleghata canal* with a road by its side. There are several roads on the right side of the image, near the middle and top portions. These are not very obvious from the images. A bridge cuts the river near the top of the image. This is called the *Rabindra Setu*.

IRS Image of Bombay

The Image considered here has three bands. These are :

Band 1– green band of wavelength 0.52 – 0.59 μm,

Band 2– red band of wavelength 0.62 – 0.68 μm, and

Band 3– near infra red band of wavelength 0.77 – 0.86 μm.

Some important landcovers of Bombay, as seen more prominently from Band 3 (Fig. 7 shows the image with 75% stretching to make it more prominent), are as follows : The elongated city area is surrounded by the Arabian sea. There is a concrete structure (on the right side top corner) connecting Bombay to New Bombay. On the Southern part of the city, there are several islands, including the well known *Elephanta islands*. The dockyard is situated on the south eastern part of Bombay, which can be seen as a set of three finger like structure. On the upper part of the image, towards left, there is a distinct criscrossed structure. This is the *Santa Cruz airport*.

The training set comprises 198 point belonging to five classes, with three features corresponding to the above mentioned three bands. The five classes are labelled *turbid water* (TW), concrete (Concr.) *habitation* (Hab), *vegetation* (Veg) and *open space (OS.)*

Issue of Large Value of H

In view of the complexity of the data sets, high values of H like 15 and 20 for the GA based classifiers were considered. Since the maximum number of regions provided by H hyperplanes is equal to 2^H, the aforesaid high values of H, make the number of regions ($=2^H$) also very large. This leads to a practical limitation of the method. However, an important point that needs to be taken into consideration is that the possible number of regions can never be larger than

Fig. 6 Band 3 of the Calcutta image with 75% stretching

Fig. 7 Band 3 of the Bombay image with 75% stretching

the number of points n in the training data set. Also, $n << 2^H$ for large H. Thus we need to consider atmost n regions while tackling this problem. In fact, the number of regions for this problem was found to be considerably less than n as well.

Results for SPOT *image of Calcutta*

Figs. 8 and 9 present the output classified images of the *GA-classifier* and *VGA-classifier* for H and H_{max} = 15 respectively. Note that for the GA based classifiers, some of the pixels remain unclassified in the output images. These are labelled as 'Uncls' Figs. 10 and 11 present the output classified images corresponding to the Bayes classifier and k-NN rule (for $k=\sqrt{n}$). Extensive experiments with other parameters of the different classifiers have been carried out, but here we present only one result for each classifier.

From the results it is found that all the classifiers able to identify *Hooghly, Rabindra Setu, Garden Reach lake, Khidirpore dockyard* and *race course* properly. Overall, the prevalence of concrete on the right bank of *Hooghly* (corresponding to the Calcutta metropolis area), and vegetation, open space and habitation on the left bank (corresponding to Howrah region) are correct. Also, the pixels corresponding to the *second bridge*, are identified as those of either concrete or road class.

Although the performance of the Bayes classifier is found to be quite good, it appears to over estimate a large road class on both the sides of the *Hooghly*. For the *VGA-classifier,* which succeeds in reducing the number of hyperplanes from 15 to 9, most of the road pixels on the right hand side of the image have been identified as members of the class PW and Concr rather than B/R. This is because of a large overlap between the Classes Concr and B/R on one hand (in fact, the latter class has been extracted from the former) and PW and B/R on the other hand. These are evident from Figs. 12 and 13 respectively. The performance of the k- NN rule was found to depend heavily on the value of k. The performance of the *GA-classifier* was also found to be poor for H=10, which improved for H=15, and then again degraded

Fig. 8 Classified Calcutta image using *GA-classifier* for H = 15

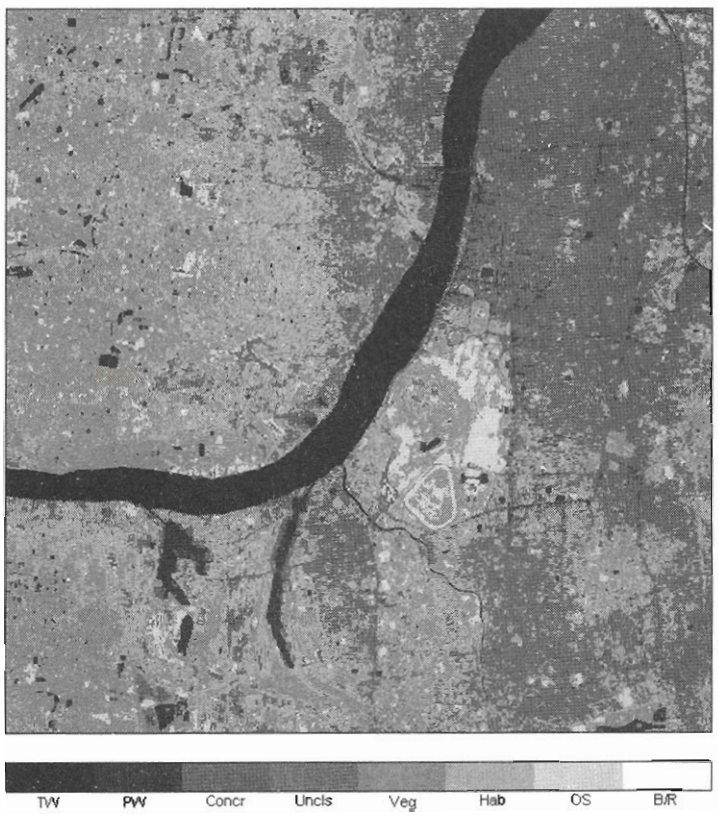

Fig. 9 Classified Calcutta image *VGA-classifier* for H_{max} = 15. Final Value of H_{opt} = 9

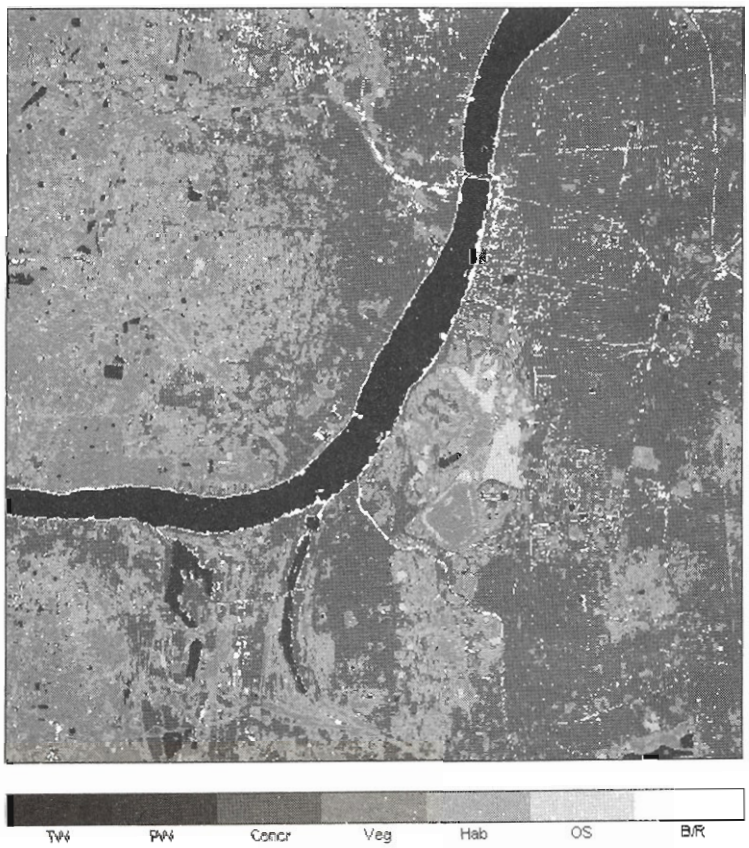

Fig. 10 Classified Calcutta image using Bayes classifier

Fig. 11 Classified Calcutta image using k-NN rule for $k=\sqrt{n}$

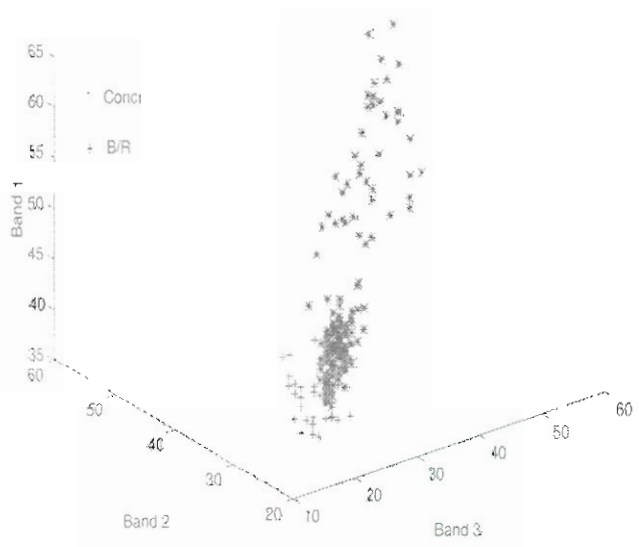

Fig. 12 Scatter plot for the classes Concr and B/R for the training data of Calcutta image

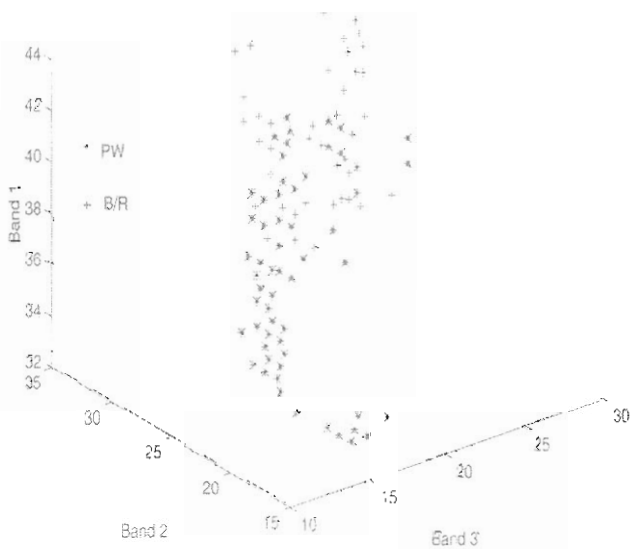

Fig. 13 Scatter plot for the classes PW and B/R for the training data of Calcutta image

for H=20. Overfitting of the data set and the subsequent reduction in the generalization capability is responsible for the degradation of performance from H=15 to H = 20, while H = 10 appears to be too small for classifying this complex data set. We have provided the results corresponding to H = 15 here.

Results for IRS Image of Bombay

Figs. 14 and 15 provide the output classified Bombay image for the GA-*Classifier* and *VGA*-classifier for H and H_{max} = 15 respectively. Figs. 16 and 17 present the output classified images corresponding to Bayes classifier and k- NN rule

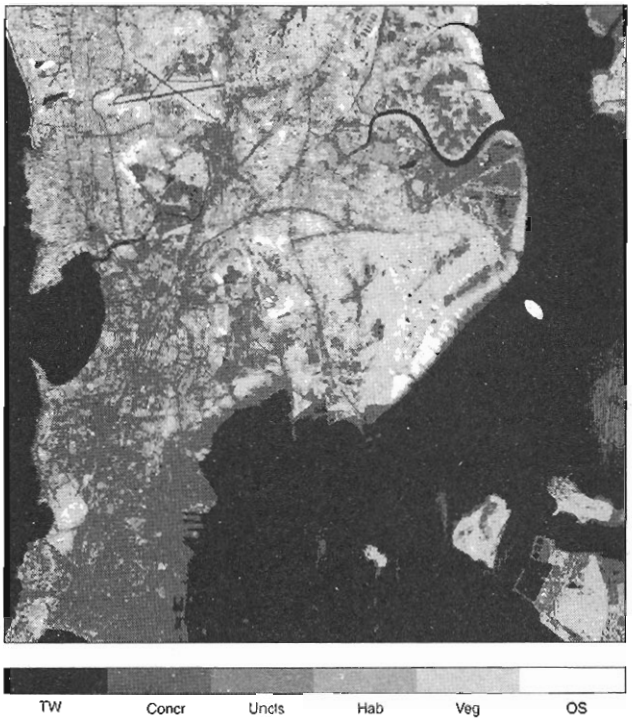

Fig. 14 Classified Bombay image using *GA-classifier* for H = 15

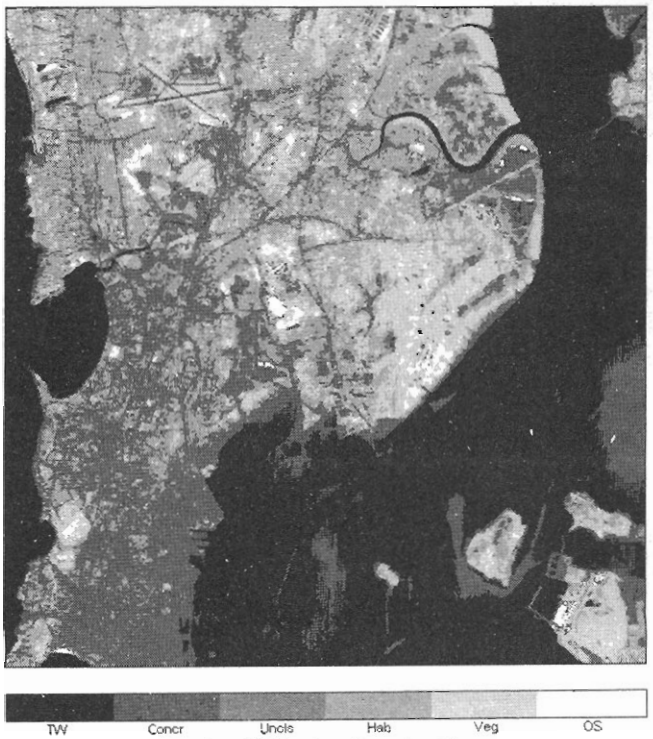

Fig. 15 Classified Bombay image using *VGA-classifier* for H_{max} = 15, Final value of H_{opt} = 11

(for $k = \sqrt{n}$) for the Bombay image. Like the Calcutta image, extensive experiments with other parameters of the different classifiers had been carried out for the Bombay images as well, but here we present only one result for each classifier.

Overall, from most of the output images one can identify the huge water body of *Arabian sea*, several road structures, the airport, several islands, dockyard and the bridge connecting Bombay to new Bombay. The southern part of Bombay is found to be identified as having large concrete area. This is known to be the most industrialized part of Bombay. As in the case of Calcutta image, it was found that the performance of the *GA-classifier* was better for $H = 15$ than with $H = 10$, but poorer for $H = 20$. We have provided the result for $H = 15$ here. The *VGA-Classifier* (Fig. 15) is found to provide good classification while reducing automatically the number of hyperplares to 11. The structure of the dockyard has come out distinctly. Only a small number of points (= 266) have remained unclassified in the image. The Bayes classifier is found to identify an unusually large portion of the image as belonging to the concrete class. The bridge connecting Bombay to New Bombay has come out as a more or less continuous structure. This is in contrast to the other classifiers where it has come out as a discontinuous structure. However, because of the predominance of the concrete class, the shape of the dockyard is found to be lost.

4 Incorporating Chromosome Differentiation in Genetic Algorithms

In this section, we investigate the effect of incorporating chromosome discrimination, thereby providing a methodology called GACD[68], on the performance of the said *GA-classifier*.

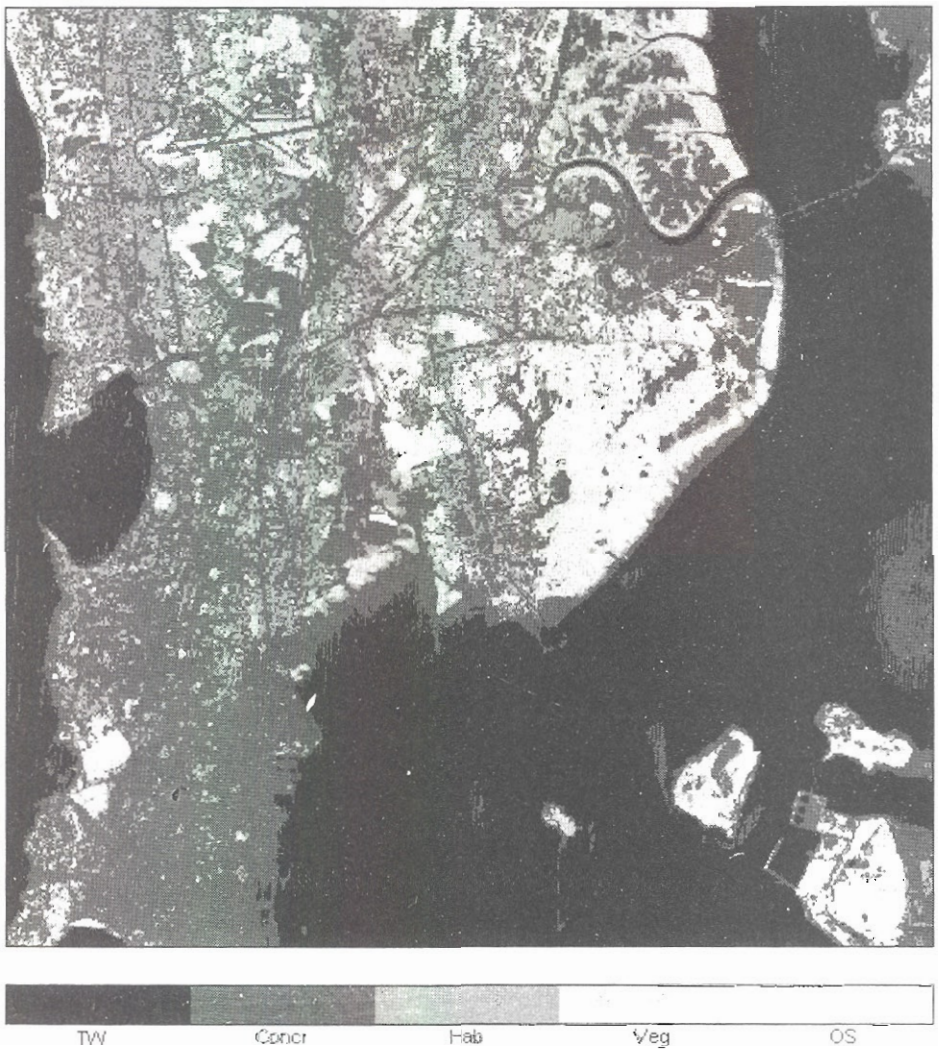

Fig. 16 Classified Bombay image using Bayes classifier

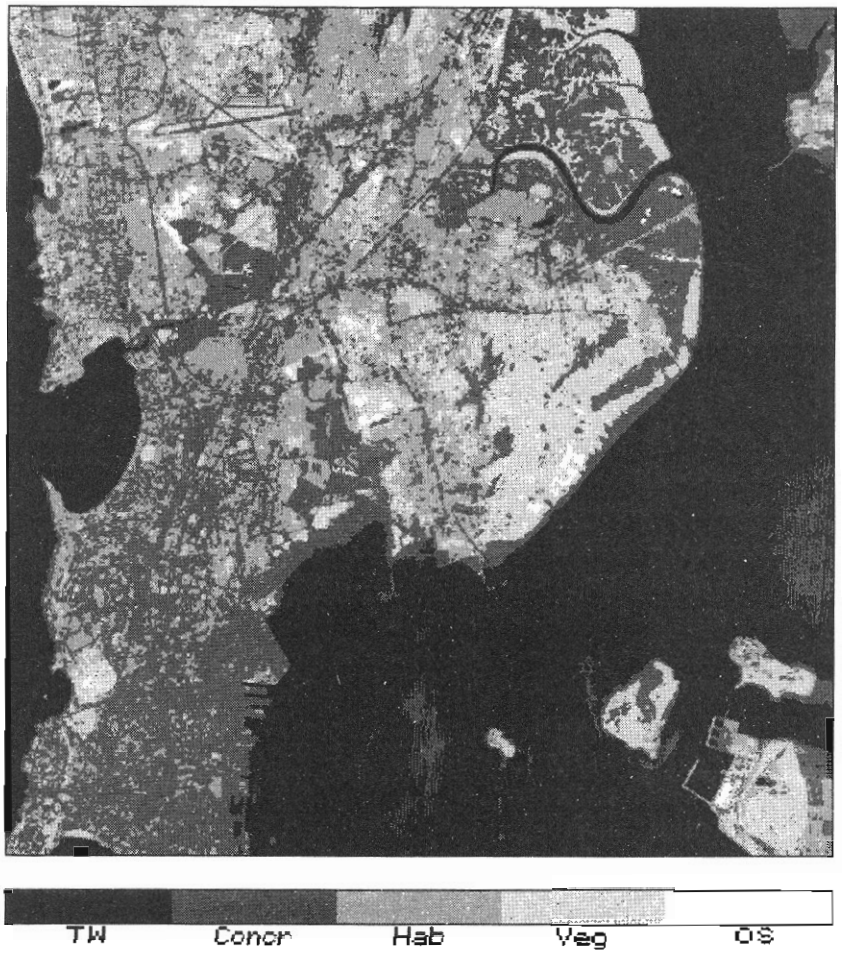

Fig. 17 Classified Bombay image using k-NN rule for k= $\sqrt{n}$

A) Description of GACD

In GACD (GA with chromosome differentiation), the chromosomes are distinguished into 2 classes, M and F. The structure of a chromosome of GACD is shown in Fig. 18. Here the l bits, termed the *data bits* encode parameters of the problem. The initial two bits, termed the *class bits* indicate the class (M, when the class bits are either 01 or 10, F, when the class bits are 00) of the chromosome. The M and F populations are initially generated in such a way that the hamming distance between the two is maximized. Two separate populations, one containing the M chromosomes (M population) and the other containing the F chromosomes (F population), are maintained over the generations. The sizes of these two populations, p_m and p_f respectively, may vary. Let $p_m + p_f = p$, where p is fixed (equivalent to the population size of conventional GA). Intially $p_m = p_f = \frac{p}{2}$. The data bits for each M chromosome are first generated randomly. One of the two *class bits*, chosen randomly, is inititalized to 0 and the other to 1. The data bits of the F chromosomes are initially generated in such a way that the hamming distance between the two populations (in terms of the data bits) is maximum. The hamming distance between two chromosomes c_1, and c_2, denoted by h (c_1, c_2), is defined as the number of bit positions in which the two chromosomes differ. Hamming distance between two populations, P_1 and P_2, denoted by h (P_1, P_2), is defined as follows :

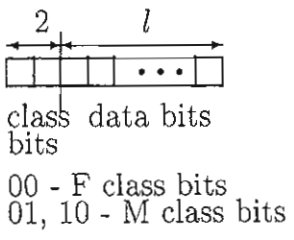

Fig. 18 Structure of a chromosome in GACD

$$h(P_1, P_2) = \sum_i \sum_j h(c_i, c_j), \quad \forall c_i \in P_1, \quad \forall c_j \in P_2.$$

Only the l data bits are used to compute the fitness for the chromosomes in a problem specific manner. Selection is performed over all the p $(= p_m + p_f)$ chromosomes, (i.e. disregarding the class information) using their fitness values. In other words, all the chromosomes compete with one another for survival. The selected chromosomes are placed in the mating pool. Crossover is applied probabilistically between an M and an F parent chromosome. Each parent contributes one class bit to the offspring. Since the F parent can only contribute a 0 (its class bits being 00), the class of the child is primarily determined by the M parent which can contribute a 1 (yielding an M child) or a 0 (yielding an F child) depending upon the bit position (among the two class bits) of the M parent chosen. This process is performed for both the offspring whereby either two M or two F or one M and one F offspring will be generated. Bit by bit mutation is performed over the data bits only with probability μ_m. The class *bits* are not mutated. Elitism is incorporated by preserving the best chromosome, among both the M and F chromosomes, seen till the current generation either inside or outside the population

B) Schema Analysis for GACD

A schema, of same length as the chromosomes being considered, is a string over the alphabet {0, 1,#} (where # represents the don't care symbol). It represents the subset of all the binary strings that match the schema at positions where the latter has 1s and Os. The schema theorem[9] which is of fundamental importance in GA, states that short, low order, above average (characterized by fitness value) schemata will receive exponentially increasing number of trials in subsequent generations. It has been proved in ref. [68] that the schema theorem holds good for GACD. It is also shown that in certain cases the lower bound of the number of schema sampled by GACD is greater than or equal to those of CGA. Or,

$$lower\_bound\ (m_{GACD}\ (h,t+1)) \geq lower\_bound\ (m_{CGA}\ (h,t+1)),$$

where $m_{GACD}(h,t+1)$ and $m_{CGA}(h,t+1)$ denote the number of instances of schema h at instant $t + 1$ obtained by GACD and CGA respectively.

C) Results

The concept of chromosome differentiation and restricted mating is incorporated in the above mentioned *GA-classifier* to construct the *GACD-classifier*. The recognition scores of the *GACD-classifier* for *Vowel* are provided in Table IV for two values of H. The corresponding scores of the *GA-classifier* are also provided in the table for the convenience of readers. The *GACD-classifier* is found to provide a performance superior to that of *GA-classifier* for both the values of H. The recognition scores during training of the *GACD-classifier* (*viz.*, 94.12% and 95.29% for H=6 and 7 respectively) were also found to be significantly superior to those of the *GA-classifier* (*viz.*, 82.35% and 88.23% for H = 6 and 7 respectively).

Table IV

Variation of the average recognition score (%) during testing for Vowel data with H for μ_c = 0.8

H	Recognition Score (%)	
	GACD-classifier	*GA-classifier*
6	75.19	71.99
7	74.94	74.68

5 Discussion and Conclusions

This article dealt with the development of several pattern classifiers, along with their theoretical and practical aspects, using both conventional genetic algorithms and some of their modifications/ enhancements. The search and optimization capability of genetic algorithms has been exploited for the placement of an appropriate number of surfaces in the feature space such that the associated number of misclassified points is minimized. The classifiers store the parameters of the surfaces constituting the final decision boundary, and the region-class associations. These are later used for determining the region and hence the class of an unknown pattern. Various versions of the genetic classifier e.g., *GA-classifier* using fixed number of surfaces, *VGA-classifier* using variable number of surfaces. *GACD-classifier* incorporating the concept of chromosome differentiation have been formulated. Some theoretical results have been provided.

The effectiveness of these genetic classifiers and their comparison with Bayes maximum likelihood classifier (which is well known for discriminating overlapping classes), k-NN rule and MLP (both of which are well known for discriminating non-overlapping, non-convex regions by generating

piecewise linear boundaries) are demonstrated on a speech data *Vowel*. The way of incorporating the concept of variable string length in VGA-*classifier* is also compared with that of Srikanth *et al.*[60] in a part of the experiment. Besides these, the problem of pixel classification from satellite images for partitioning various landcover types with ill-defined boundaries is considered as another real life application.

In this regard one may note the analogy between the classification principles of the GA based classifiers and MLP with hard limiting neurons. It is known in the literature[69] that the latter approximates the decision boundary by piecewise linear surfaces. The parameters of these surfaces are encoded in the connection weights and threshold biases of the network. Similarly, the *VGA-classifier* also generates decision boundaries by appropriately fitting a number of hyperplanes in the feature space. The parameters are encoded in the chromosomes. The obvious advantages of the latter over the MLP is that it performs concurrent search for a number of sets of hyperplanes, each representing a different classification in the feature space. On the other hand, the MLP deals with only one such set. Thus it has a greater chance of getting stuck at a local optimum, which the *VGA-classifier* can overcome. Moreover, *VGA-classifier* does not assume any fixed value of the number of hyperplanes, while MLP assumes a fixed number of hidden nodes and layers. This results in the problem of over fitting with an associated loss of generalization capability for MLP. As a consequence of the above discussion, an attempt to determine the architecture of MLP automatically, based on the results of the *VGA-classifier*, was reported in ref. [53]. This constitutes another interesting application of the *VGA-classifier*.

In this article, binary representation of chromosomes in linear form has been used, primarily because it is well studied in the literature, and it maximizes the number of schemata sampled by each member of the population; thereby making the implicit parallelism of GAs to be used to the fullest. However, in many practical situations, this may not be a natural choice. Thus, other kinds of representation, like floating point, tree, matrix representation[12,70] may be studied.

A modification of the fitness function by incorporating the information on relative position of the boundary from the training data may constitute another part of further investigation. The classification methodology presented here, considers only the total number of misclassified points as the optimizing criterion. It does not take the classwise recognition scores into account. This may sometimes lead to an undesirable situation where the overall recognition score is high, but some classes are totally ignored. In order to tackle this, an investigation may be undertaken where the classwise weighted scores constitute the fitness criterion of the chromosomes. The weighting factor may take some a priori class information, like the class probabilities, into account.

In GACD, the chromosomes have been differentiated into two classes and a form of restricted mating has been applied. As a part of future work, the effect of differentiating the chromosomes into more than two categories may be investigated. Moreover, the effect of incorporating variable string lengths in GACD, thereby yielding a methodology called VGACD, needs to be studied as well. Results of initial investigations in this direction are encouraging.

An investigation needs to be performed for the selection of appropriate kinds of surfaces (i.e., linear, higher order) for modelling the class boundaries. The classification scheme developed here is a supervised one, necessitating the presence of training data. An unsupervised scheme, based on the similar principle of placement of hyperplanes, may be developed for partitioning of unlabelled data sets.

References

1 R K Belew and J B Booker (Eds) *Proc 4th Int Conf Genetic Algorithms* San Meteo Morgan Kaufmann (1991)

2 B P Buckles and F E Petry (Eds) *Genetic Algorithms* Los Almitos IEEE Computer Society Press (1994)

3 L Davis (Ed) *Handbook of Genetic Algorithms* New York Van Nostrand Reinhold (1991)

4 K A D Jong *Machine Learning* **3** (1988) 121

5 L J Eshelman (Eds) *proc 6th Int Conf Genetic Algorithms* San Mateo Morgan Kaufmann (1995)

6 J L R Filho, P C Treleaven and C Alipi *IEEE Computer* (1994) 28

7 D B Fogel *Evolutionary Computation Toward a New Philosophy of Machine Intelligence* New York IEEE Press (1995)

8 S Forrest (Ed) *Proc 5th Int Conf Genetic Algorithms* San Mateo Morgan Kaufmann (1993)

9 D E Goldberg *Genetic Algorithms in Search, Optimization and Machine Learning* New York Addison-Wesley (1989)

10 J J Grefenstette (Ed) *Proc 1st Int Conf Genetic Algorithms* Hillsdale Lawrence Erlbaum Associates (1985)

11 J H Holland *Adaptation in Natural and Artificial Systems* Ann Arbor The University of Michigan Press (1975)

12 Z Michalewicz *Genetic Algorithms + Data Structures = Evolution Programs* New York Springer-Verlag (1992)

13 *Proc 1995 IEEE Int Conf Evolutionary Computation* Perth Australia (1995)

14 M Srinivas and L M Patnaik *IEEE Computer* (1994) 17

15 J D Bagley *The Behaviour of Adaptive System which Employ Genetic and Correlation Algorithms PhD Thesis* University of Michigan Ann Arbor (1967)

16 E S Gelsema (Ed) *Special Issue on Genetic Algorithms, Pattern Recognition Letters* **16** (1995) 8

17 S K Pal and P P Wang (Eds.) *Genetic Algorithms for Pattern Recognition* Boca Raton CRC Press (1996)

18 W Siedlecki and J Sklansky *Pattern Recog. Lett* **10** (1989) 335

19 C A Ankenbrandt, BP Buckles and F.E Petry *Pattern Recog. Lett* **11** (1990) 285

20 A Hill and C J Taylor *Image and Vision Comput* **10** (1992) 295

21 S K Pal, D Bhandari and M K Kundu *Pattern Recog Lett* **15** (1994) 261

22 G Seetharaman, A Narasimahan and L Stor *Proc* SPIE Conf *Stochastics and Neural Methods in Signal Processing and Computer Vision* **1569** (1991)

23 L B Booker, D E Goldberg and J H Holland *Art Intell* **40** (1989) 235

24 H Ishibuchi, T Murata and H Tanaka *Genetic Algorithms for Pattern Recognition* (Eds S K Pal and P P Wang) Boca Raton: CRC Press (1996) 227

25 H Ishibuchi, M Nii and T Murata *Proc IEEE Int Conf Neural Networks* (1997) 2390

26 H Ishibuchi, K Nozaki, N Yamamoto and H Tanaka *Proc 3rd IEEE Int Conf Fuzzy Syst* (1994) 1963

27 H Ishibuci, K Nozaki, N Yamamoto and H Tanaka *IEEE Trans Fuzzy Systs* **3** (1995) 260

28 C J Janikow *Genetic Algorithms for Pattern Recognition Eds:* S K Pal and P P Wang Boca Raton: CRC Press (1996) 253

29 S Bornholdt and D Graudenz *Neural Networks* **5** (1992) 327

30 S A Harp and T Samad *Handbook of Genetic Algorithms* (Ed L Davis) New York: Van Nostrand Reinhold (1991) 202

31 V Maniezzo *IEEE Trans Neural Networks* 5 (1994) 39

32. D J Montana and L Davis *Proc 11th Int Joint Conf Intell* (Ed N S Sridharan) San Mateo: Morgan Kaufman (1989) 792

33 S K Pal and D Bhandari *Inform Sci* **80** (1994) 213

34 S Saha and J P Christensen *inform Sci* 79 (1994) 191

35 J D Schaffer, R A Caruana and L J Eshelman *Physica D* **42** (1990) 244

36 D Whitley, T Starkweather and C Bogart *Parallel Computing* **14** (1990) 347

37 F A Cleveland and S F Smith *Proc 3rd Int Conf Genetic Algorithms* (Ed J D Schaffer) San Mateo: Morgan Kaufmann (1989) 160

38 L Davis *Proc 1st Conf Genetic Algorithms* (Ed. J J Grefenstette) Hillsdale Lawrence Erlbaum Associates (1985) 136

39 B W Wah, A leumwannonthachai and Y Li *Genetic Algorithms for Pattern Recognition* (Eds S K Pal and P P Wang) Boca Raton CRC Press (1996) 87

40 T Cleghorn, P Baffes and L Wang *Proc SOAR* Houston (1988) 81

41 J J Grefenstette, R Gopal, B Rosmaita and D Van Gucht *Proc 1st Conf Genetic Algorithms* (Ed J J Grefenstette) Hillsdale Lawrence Erlbaum Associates (1985) 160

42 P Jog, Y Suh, and D V Gucht *Proc 3rd Int Conf Genetic Algorithms* (Ed J D Schaffer) San Mateo Morgan Kaufmam (1989) 110

43 I M Oliver, D J Smith and J R C Holland *Proc 2nd Int Conf. Genetic Algorithms* Hillsdale Lawrence Erlbaum Associates (1987) 224

44 Z Michalewicz and C Z Janikow *Statistics and Computing* **1** (1991) 75

45 D Bhandari, C A Murthy and S K Pal *Int J Pattern Recog Art Intell* **10** (1996) 731

46 C A Murthy, D Bhandari and S K Pal *Fundamenta Informaticae* **35** (1998) 4

47 S De, A Ghosh and S K Pal *Genetic Algorithms for Pattern Recognition* (Eds S K Pal and P P Wang) Boca Raton CRC Press (1996) 1

48 A Ghosh, S Tsutsui and H Tanaka *Int J Knowledge-based Intell Eng Syst* **1** (1997) 86

49 D Bhandari, N R Pal and S K Pal *Inform Sci* **79** (1994) 251

50 H J Muller (Ed) *Studies in Genetics-Selected papers* Bloomington Indiana University Press (1992)

51 S K Pal, S De, and A Ghosh *Int J Patt Recog Art Intell* **11** (1997) 447

52 S G Romaniuk *Genetic Algorithms for Patt Recognition* (Eds S K Pal and P P Wang) Boca Raton CRC Press (1996) 179

53 S Bandyopadhyay and S K Pal *Fundamental Informaticae* **37** (1999) 177

54 S K Pal, A Ghosh and M K Kundu (Eds) *Soft Computing for Image Processing* Heidelberg Physica Verlag (1999)

55 S K Pal and S Mitra *Neuro-fuzzy Pattern Recognition: Methods in Soft Computing Paradigm* New York John Wiley (1999)

56 M Prakash and M N Murty *Pattern Recog Lett* **16** (1995) 781

57 C A Murthy and N Chowdhury *Pattern Recog Lett* **17** (1996) 825

58 J T Tou and R C Gonzalez *Pattern Recognition Priniciples* Reading Addison-Wesley (1974)

59 K Krishna *Hybrid Evolutionary Algorithms for Supervised and Unsupervised Learning* Ph D Thesis Department of Electrical Engineering Indian Institute of Science Bangalore India (1998)

60 R Srikanth, R George, N Warsi, D Prabhu, F E Petry and B P Buckles *Pattern Recog Lett* **16** (1995) 789

61 S Bandyopadhyay *Pattern Classification Using Genetic Algorithms* PhD Thesis Machine Intelligence Unit Indian Statistical Institute Calcutta India (1998)

62 J J Grefenste *IEEE Trans Sys Man Cyberns* **16** (1986) 122

63 S K Pal, S Bandyopadhyay and C A Murthy *IEEE Trans Syst Man Cyberns* **28** (1998) 816

64 D E Goiberg, K Deb and B Korb *Complex Systs* **3** (1989) 483

65 S Bandyopandhyay, C A Murthy and SK Pal Pattern Recog *Lett.* **19** (1998) 1171

66 S Bandyopadhyay, C A Murthy and S K Pal *J Franklim Institute* **336** (1999) 387

67 S K Pal and D D Majumder *IEEE Trans Syst Man Cyberns* **SMC-7** (1977) 625

68 S Bandyopadhyay, S K Pal and U Maulik *Inform Sci* **104** (1998) 293

69 R P Lippmann *IEEE ASSP Magazine* **4** (1987) 4

70 H V Hove and A Verschoren *Genetic Algorithms for Pattern Recognition* (Eds. (S K Pal and P P Wang) Boca Raton CRC Press (1996) 145