In search of optimal clusters using genetic algorithms

C.A. Murthy [a,*], Nirmalya Chowdhury [b]

[a] *Machine Intelligence Unit, Indian Statistical Institute, 203 Barrackpore Trunk Road, Calcutta 700 035, India*

[b] *Ramakrishna Mission Shilpapitha, Belgharia, Calcutta 700 056, India*

Received 18 January 1995; revised 18 March 1996

Abstract

Genetic Algorithms (GAs) are generally portrayed as search procedures which can optimize functions based on a limited sample of function values. In this paper, GAs have been used in an attempt to optimize a specified objective function related to a clustering problem. Several experiments on synthetic and real life data sets show the utility of the proposed method. K-Means is one of the most popular methods adopted to solve the clustering problem. Analysis of the experimental results shows that the proposed method may improve the final output of K-Means where an improvement is possible.

Keywords: Pattern recognition; Clustering; K-means; Genetic algorithms

1. Introduction

Clustering is an important technique used in discovering inherent structure present in a set of objects. More specifically, the purpose of cluster analysis (Anderberg, 1973; Devijver and Kittler, 1982; Jain and Dubes, 1988; Tou and Gonzalez, 1974) is to group a set of patterns, usually vectors in a multi-dimensional space, into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. We assume that the given patterns belong to an n-dimensional Euclidean space $\mathbb{R}^n$ and that the dissimilarity measure is the Euclidean distance.

Let the set of patterns M be $\{x_1, x_2, \ldots, x_m\}$, where x_i is the ith pattern vector. Let the number of clusters be k. If the clusters are represented by $C_1, C_2, \ldots, C_k$, then

P1. $C_i \neq \emptyset$, for $i = 1, \ldots, k$,
P2. $C_i \cap C_j = \emptyset$ for $i \neq j$, and
P3. $\bigcup_{i=1}^{k} C_i = M$.

Clustering techniques may broadly be divided into two categories: hierarchical and non-hierarchical (Anderberg, 1973). Among the non-hierarchical clustering techniques, the K-means (or C-means or basic Isodata) algorithm has been one of the more widely used algorithms. The K-means algorithm is based on the optimization of a specified objective function. It attempts to minimize the sum of squared Euclidean distances between patterns and cluster centers. It was shown in (Selim and Ismail, 1984) that this algorithm may converge to a local minimum solution.

Global solutions of large problems cannot be found with a reasonable amount of computational effort (Spath, 1980). This suggested the development of several approximate methods to solve the underlying optimization problem. Most of these techniques arrive at a local minimum solution which does not

necessarily coincide with the desired global minimum (Selim and Ismail, 1984). In this paper, we have used Genetic Algorithms (GAs) in an attempt to reach the optimal solution for the clustering problem. Results of the experiments with synthetic data as well as with real-life data are reported.

Section 2 of this paper presents the statement of the problem. Section 3 provides the details of the proposed clustering method with GAs. Experimental results and analysis are reported in Section 4.

2. Statement of the problem

There are several ways in which a given data set can be clustered. One of the principles used for clustering is to minimize the sum of squared Euclidean distances of the data points from their respective cluster means. Mathematically this principle is stated below.

1. Let $C_1, C_2, \ldots, C_k$ be a set of k clusters of M.
2. Let $z_j = (\sum_{x \in C_j} x)/\#C_j$ for $j = 1, 2, \ldots, k$, where x is a pattern vector in C_j and $\# C_j$ represents the number of points in C_j.
3. Let $f(C_1, C_2, \ldots, C_k) = \sum_{j=1}^{k} \sum_{x \in C_j} |x - z_j|^2$. We shall refer to $f(C_1, C_2, \ldots, C_k)$ as the objective function of the clustering $C_1, C_2, \ldots, C_k$.
4. Minimize $f(C_1, C_2, \ldots, C_k)$ over all $C_1, C_2, \ldots, C_k$ satisfying P1, P2 and P3 stated in Section 1.

The objective function f is non-convex and hence the problem may have local minimum solutions which are not necessarily optimal (Selim and Ismail, 1984). In fact, all possible clusterings of M are to be considered to get the optimal $C_1, C_2, \ldots, C_k$. So obtaining the exact solution of the problem is theoretically possible, yet not feasible in practice due to limitations of computer storage and time. If exhaustive enumeration is used to solve the problem, then one requires the evaluation of $S(m, k)$ partitions (Anderberg, 1973; Spath, 1980), where

$$S(m,k) = \frac{1}{k!} \sum_{j=1}^{k} (-1)^{k-j} \binom{k}{j} j^m.$$

This clearly indicates that exhaustive enumeration cannot lead to the required solution for most practical problems in reasonable computation time. For example, for the crude-oil data (Johnson and Wichern, 1982) used in this paper, the exact solution requires the examination of $S(56, 3)$ partitions. (Note that $S(56, 3) > 10^{18}$.)

Thus, approximate heuristic techniques seeking a compromise or looking for a local minimum solution which is not necessarily global have usually been adopted. In this paper we have applied a GA in an attempt to get the optimal value of the function f for a given clustering problem. The next section describes the method in detail.

3. Clustering using Genetic Algorithms

Genetic Algorithms (GAs) are stochastic search methods based on the principle of natural genetic systems (Goldberg, 1989; Michalewicz, 1992). They perform a multi-dimensional search in order to provide an optimal value of an evaluation (fitness) function in an optimization problem. Unlike conventional search methods, GAs deal with multiple solutions simultaneously and compute the fitness function values for these solutions. GAs are theoretically and empirically found to provide global near-optimal solutions for various complex optimization problems in the field of operation research, VLSI design, Pattern Recognition, Image Processing, Machine Learning, etc. (Ankerbrandt et al., 1990; Belew and Booker, 1991; Bornholdt and Graudenz, 1992).

While solving an optimization problem using GAs, each solution is usually coded as a binary string (called chromosome) of finite length. Each string or chromosome is considered as an individual. A collection of P such individuals is called a population. GAs start with a randomly generated population of size P. In each iteration, a new population of the same size is generated from the current population using two basic operations on the individuals. These operators are Selection and Reproduction. Reproduction consists of crossover and mutation operations.

In GAs, the best string obtained so far is preserved in a separate location outside the population so that the algorithm may report the best value found, among all possible solutions inspected during the whole process. In the present work, we have used the elitist model (EGA) of selection of De Jong (1992), where the best string obtained in the previous iteration is copied into the current population.

The remaining part of this section describes in detail the genetic algorithm that we propose for clustering. First, the string representation and the initial population for the problem under consideration are discussed. Then the genetic operators and the way they are used are stated. The last part of this section deals with the stopping criteria for the GA.

3.1. String representation and initial population

String representation. To solve partitioning problems with GAs, one must encode partitions in a way that allows manipulation by genetic operators. We consider an encoding method where a partition is encoded as a string of length m (where m is the number of data points in M). The ith element of the string denotes the group number assigned to point x_i. For example the partition $\{x_1, x_4\}\ \{x_3, x_6\}\ \{x_2, x_5\}\ \{x_7\}$ is represented by the string (1 3 2 1 3 2 4). We have adopted this method, since it allows the use of the standard single-point crossover operation. The value of the ith element of a string denotes the cluster membership of the ith data point in M. Thus, each string represents a possible cluster configuration and the fitness function for each string is the sum of the squared Euclidean distances between the patterns and their respective cluster centers. So, here the fitness function is the objective function f described in Section 2.

Initial population. An initial population of size P for a genetic algorithm is usually chosen at random. In the present implementation, several strings of length m are generated randomly where the value of each element of the string is allowed to lie between 1 and k. Only valid strings (that have at least one data point in each cluster) are considered to be included in the initial population to avoid wastage of processing time on invalid strings.

There exist no guidelines for choosing the 'appropriate' value of the size (P) of the initial population. In this work, we have taken $P = 6$ and this value of P is kept fixed throughout the experiment. Note that it has been shown in (Bhandari et al., 1996) that as the number of iterations goes to infinity the elitist model of GAs will provide the optimal string for any population size P.

3.2. Genetic operators

Selection. The 'selection' operator mimics the 'survival of the fittest' concept of natural genetic systems. Here strings are selected from a population to create a mating pool. The probability of selection of a particular string is directly or inversely proportional to the fitness value depending on whether the problem is that of maximization or minimization. The present problem is a minimization problem and thus the probability of selecting a particular string in the population is inversely proportional to the fitness value. The size of the mating pool is taken to be same as that of population.

Crossover. Crossover exchanges information between two parent strings and generates two children for the next population. A pair of chromosomes

$$\beta = (\beta_m \beta_{m-1} \cdots \beta_2 \beta_1),$$
$$\gamma = (\gamma_m \gamma_{m-1} \cdots \gamma_2 \gamma_1)$$

is selected randomly from the mating pool. Then the crossover is performed with probability p (crossover probability) in the following way.

Generate randomly an integer position pos from the range of $[1, m-1]$. Then two chromosomes β and γ are replaced by a pair α and δ, where

$$\alpha = (\beta_m \beta_{m-1} \cdots \beta_{pos} \gamma_{pos-1} \cdots \gamma_2 \gamma_1),$$
$$\delta = (\gamma_m \gamma_{m-1} \cdots \gamma_{pos} \beta_{pos+1} \cdots \beta_2 \beta_1).$$

Crossover operation on the mating pool of size P (P is even) is performed in the following way:

- Select $P/2$ pairs of strings randomly from the mating pool so that every string in the mating pool belongs to exactly one pair of strings.
- For each pair of strings, generate a random number rnd from $[0, 1]$. If $rnd \leqslant p$ then perform crossover; otherwise no crossover is performed.

Usually in GAs, p is chosen to have a value in the interval $[0.25, 1]$. In the present work p is taken to be 0.8 and the population size P is taken to be 6 for all generations. The crossover operation between two strings, as stated above, is performed at one position. This is referred to as *single-point crossover* (Michalewicz, 1992).

The single-point crossover operator may create one problem. The child may have fewer groups than the parents. For example, if we cross strings (1 2 2 3 2 1) and (1 3 3 2 2 1) after the third position with single-point crossover, then the two children are (1 2 2 2 2 1) and (1 3 3 3 2 1). Note that the first child (1 2 2 2 2 1) has only two groups instead of three. To avoid this problem, we use sampling with replacement. In other words, we repeat crossover until we get a child with k groups or until a limit on the number of attempted crossovers is reached. (We have used a limit of 100.) If the limit is reached without finding a child with k groups, the child is set to one of the parents chosen at random. Note that this strategy was adopted in a previous work (Jones and Beltramo, 1991) also. (Another way of performing the validity check of the string is to use the strategy of sampling without replacement. In such a case the maximum number of attempted crossovers would be $m - 1$, but a list of invalid break points has to be maintained.) It may also be noted that invalid strings having fewer groups will have a larger value of the evaluation function and thus they will be rejected eventually. But such strings should be excluded from the population to eliminate wastage of processing time and to make room for valid strings to produce a better valid offspring.

Mutation. Mutation is an occasional random alteration of a character. Every character β_i, $i = 1, 2, \ldots, m$, in each chromosome (generated after crossover) has equal chance to undergo mutation. Note that any string can be generated from any given string by mutation operation.

Note that the mutation operation can, theoretically, produce invalid offspring. Thus a similar procedure (as performed after the crossover operation) for checking the validity of the offspring may be incorporated after mutation too, if the need arises. In all of our experiments, mutation did not produce invalid offsprings. Hence no validity check for strings has been incorporated for the mutation operation.

The mutation introduces some extra variability into the population. Though it is usually performed with very low probability q, it has an important role in the generation process (Michalewicz, 1992). The mutation probability q is usually taken in the interval $[0, 0.5]$. The value of q is usually taken to be fixed. Sometimes it is varied with the number of iterations. For details, the reader is referred to (Qi and Palmieri, 1994). We have considered varying the mutation probability for reasons explained in the next subsection.

Elitist strategy. The aim of the elitist strategy is to carry the best string from the previous iteration into the next. We have implemented this strategy in the following way:

(a) Copy the best string (say s_0) of the initial population in a separate location.
(b) Perform selection, crossover and mutation operations to obtain a new population (say Q_1).
(c) Compare the worst string in Q_1 (say s_1) with s_0 in terms of their fitness values. If s_1 is found to be worse than s_0, then replace s_1 by s_0.
(d) Find the best string in Q_1 (say s_2) and replace s_0 by s_2.

Note. Steps (b), (c) and (d) constitute one iteration of the proposed GA based method. These steps are repeated till the stopping criterion is satisfied. Observe that a string s_1 is said to be better than another string s_2, if the fitness value of s_1 is less than that of s_2, since the problem under consideration is a minimization problem.

3.3. Stopping criterion

There exists no stopping criterion in the literature (Davis and Principe, 1991; Goldberg, 1989; Michalewicz, 1992) which ensures the convergence of GAs to an optimal solution. Usually, two stopping criteria are used in genetic algorithms. In the first, the process is executed for a fixed number of iterations and the best string obtained is taken to be the optimal one. In the other, the algorithm is terminated if no further improvement in the fitness value of the best string is observed for a fixed number of iterations, and the best string obtained is taken to be the optimal one. We have used the first method in the experiment. Note that the population size P is taken to be 6 in all the experiments. For a higher value of P, one may probably consider fewer iterations for stopping the GA for the same search space. For the same P and for different sizes of the search spaces,

the stopping times (maximum number of iterations of the GA-based method) are taken to be different (see Section 4). Some theoretical aspects relating to stopping times for the elitist model of GA are discussed in (Murthy et al., 1996).

In order to obtain the optimal string, one needs to maintain the population diversity. This means that the mutation probability needs to be high. On the other hand, as the optimal string is being approached, fewer changes in the present strings are necessary to move in the desired direction. This implies that the mutation probability needs to be reduced as the number of iterations increases. Sometimes, at any stage of the algorithm, many changes in the present best string are required to get the optimal string. Thus, to have an efficient search process with GAs, the variation of the mutation probability with the number of iterations may be made variable. Some of the ways in which the mutation probability can be varied are shown in Fig. 1(a-d). We have followed the function shown in Fig. 1(a) for the variation of the mutation probability. In fact, we have started with a mutation probability value of $q = 0.5$. The q value is then varied as a step function of the number of iterations until it reaches a value of $1/m$. The minimum value of the mutation probability is taken to be $1/m$.

4. Experimental results and analysis

Experiments have been carried out both on synthetic and on real-life data sets to judge the validity of the proposed method. The experiments and their results are described below.

Experiment 1. The objective of this experiment is to check whether the proposed GA-based clustering method provides the optimal clustering without searching all possible partitions. For this purpose, we have considered three data sets each having 10 randomly generated points in $\mathbb{R}^2$. We have then computed the optimal value of the objective function f for $k = 2$ by Exhaustive Search (ES) for each of these three data sets. Note that it requires searching 511 ($S(10,2) = 511$) partitions to get the optimal value of f. Then the above-mentioned clustering method with GA has been applied to all three data sets. The maximum number of iterations is taken to be 50. For each data set we have conducted the experiment five times with randomly generated initial populations. Table 1 shows that the proposed method has reached the optimal value of f in less than 20 iterations in most of the cases with a maximum of 26 iterations. Note that, since the population size P is 6, six partitions are examined at each iteration. So, the method required the examination of a maximum of 156 partitions to reach the optimal value.

Table 1
Results using three data sets of size 10

Data set	f found by ES	Initial population	f found by GA	No. iterations required
1	2.3974	1	2.3974	7
		2	2.3974	13
		3	2.3974	10
		4	2.3974	8
		5	2.3974	3
2	2.7196	1	2.7196	11
		2	2.7196	4
		3	2.7196	19
		4	2.7196	6
		5	2.7196	8
3	2.5238	1	2.5238	6
		2	2.5238	21
		3	2.5238	8
		4	2.5238	26
		5	2.5238	11

Table 2
Results using a data set of size 50

Initial population	f found by GA	No. iterations required
1	15.8243	3485
2	15.8243	6521
3	15.8243	7386
4	15.8243	5263
5	15.8243	4208

Experiment 2. In this experiment, 50 points in $\mathbb{R}^2$ are generated randomly from four classes. The classes are taken in such a way that the distance of any point from its class mean is less than the distance of that point from any other class mean. The data points and the corresponding four clusters are shown in Fig. 2. The objective of this experiment is to inspect whether the proposed GA-based method can provide the same clusters as shown in Fig. 2. Note that the exhaustive search for this data set is computationally expensive ($S(50,4) > 10^{24}$).

Table 2 shows the results of this experiment. Five

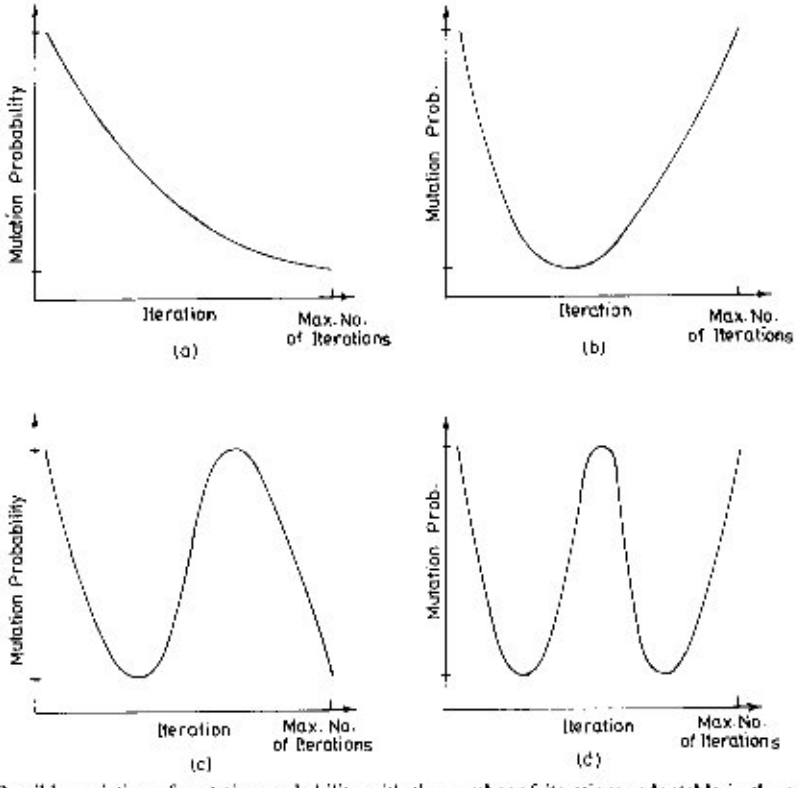

Fig. 1. Possible variation of mutation probability with the number of iterations, adoptable in the experiment.

Table 3
Results showing improvements of K-means final output using the proposed GA-based method on a data set of size 50

Initial config.	f found by K-means	f found by GA	No. iterations reqd. for GA
1	15.7456	15.7456	2000
2	17.4714	15.7456	1026
3	15.7456	15.7456	2000
4	15.7456	15.7456	2000
5	16.4901	15.7456	317

sets of initial populations are also used for this experiment. The maximum number of iterations is taken to be 10000. The objective function value obtained by the proposed method and the number of iterations required are depicted in Table 2. In all these five cases, the proposed GA-based method provided the expected results. Note also that the maximum number of possible partitions examined by the method is drastically less than that required in the exhaustive enumeration process.

Experiment 3. It is known in the literature that the K-means algorithm may not provide the optimal clustering. It may converge to a local minimum (Selim and Ismail, 1984). The objective of this experiment is to find whether the GA-based method can provide a better output if the result of the K-means algorithm (after convergence is achieved) is taken as one of the strings in the initial population. For this purpose, we have considered another synthetic data set of size 50 in $\mathbb{R}^2$. The data points are generated randomly from four classes which are placed very close to each other. Five different initial cluster configurations were taken randomly and the K-means algorithm was run on this data set till convergence was achieved for each of these five initial configurations. Each one of these five outputs is taken as a string in the initial population for GA and the other string in the initial population is generated randomly. The maximum number of iterations for the GA-based method is taken to be 2000. The number of iterations taken by the proposed GA-based method and the final output are as shown in Table 3. It can

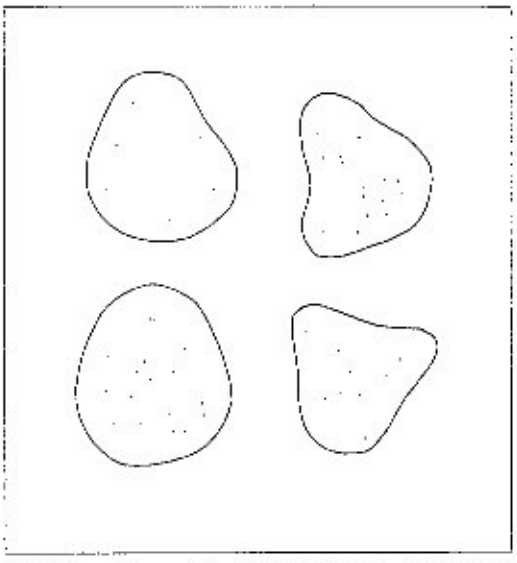

Fig. 2. Data points and cluster configuration obtained by the proposed method in Experiment 2.

Table 4
Results of K-means on crude-oil data

Initial config.	f found by K-means
1	283.7432
2	279.2709
3	296.4848
4	279.2709
5	279.2432

Table 5
Results of the proposed GA-based method on crude-oil data

Initial population	f found by GA	No. iterations required
1	278.9651	8847
2	278.9651	9712
3	278.9651	7382
4	278.9651	8273
5	278.9651	8054

be seen that the proposed method improves the final output of the K-means by giving a lower value of the objective function in a maximum of 1026 iterations where such an improvement is possible. In this experiment, the number of iterations taken by the K-means algorithm to converge is less than or equal to 16.

Experiment 4. The objective of this experiment is to compare the results of the K-means algorithm and the proposed GA-based method on a real-life data set. Crude-oil data (Johnson and Wichern, 1982), having 56 data points, 5 features and 3 classes is chosen for this experiment. The K-means algorithm was run (till convergence was achieved) on this data set for 50 randomly generated initial cluster configurations. The proposed GA-based method was also run on this data set for 50 randomly generated initial populations. The maximum number of iterations for the GA-based method is taken to be 10000. It is found that the GA-based method provided an objective function value of 278.9651 in all 50 cases. The lowest value of the objective function provided by the K-means algorithm in all 50 cases is 279.2432. The K-means algorithm achieved this value in 39 out of 50 cases. The highest value of the objective function provided by the K-means algorithm is 296.4848. Experimental results of K-means with five different initial cluster configurations are reported in Table 4. Table 5 shows the results of the proposed GA-based method with five different initial populations. The number of iterations taken by the K-means algorithm to converge in all 50 cases is less than or equal to 23.

Note that different initial populations are considered in each experiment described above. The number of iterations taken by the GA-based method to converge (i.e. the fitness value is found to be the same after that iteration) has been stated in the tables (Tables 1–5) for the above experiments. It may be noted that the number of iterations required to converge is smaller than the stopping time value for each experiment. But premature convergence (i.e. all or most of the strings in the population are identical after that iteration) of the process has not been observed in the above experiments.

5. Conclusions and discussion

The aim of this work is to observe whether the proposed GA-based method can find the optimal clustering without searching all possible partitions. Experiment 1 shows that the proposed method indeed finds the optimal clustering without searching all possible partitions. In Experiment 2, the data points are such that the optimal clustering was known. The GA-based method provided the expected results. Experiment 3 shows that the output of K-means may be further improved by the proposed method. In Experiment 4,

where both the K-means algorithm and the GA-based method were run independently on a real-life data set, we find that in all 50 runs K-means provided higher value of the objective function.

Note that in Experiment 4, the K-means algorithm takes very few iterations to converge. But it converges to a local minimum. GAs do not face this problem, since here the process can theoretically move from any partition to any other partition with non-zero probability. The GA-based method suggested in this paper has been found to provide good results for all the data sets considered for experimentation.

Observe that the population size P is taken to be 6 for all the experiments, although the sizes of the search spaces associated with each problem are not the same. But we have used different stopping times (maximum number of iterations of the GA-based method) depending upon the size of the search space. There probably exists a relationship between the stopping time and the population size for a given search space. The theoretical results available on this aspect of GAs are very little (Murthy et al., 1996). For a higher value of P, probably, a smaller stopping time would provide similar results.

While solving an optimization problem using GAs, one always needs to make a compromise between two conflicting facets of GAs. One facet is the maintenance of population diversity such that the process searches for optimal strings in different regions of the search space. The other facet is that as the GA goes nearer to the optimal solution, fewer changes in the bits of the present best string are necessary to get the optimal string. This means that as the process approaches the optimal string, the search space needs to be confined to the strings in the vicinity of the present best string. If the population diversity is maintained, then it is difficult to make the process perform the other facet namely allowing only few changes in the strings. On the other hand, if the process performs the second facet alone, then it may lead to premature convergence. Both these facets have been advocated in our method in terms of varying the mutation probability as depicted in Fig. 1. The experiments carried out in this work utilize the functional form of Fig. 1(a) for varying the mutation probability. Premature convergence of the GA-based method has not been found in any of the experiments reported in this paper.

Acknowledgements

The authors acknowledge Mr. D. Bhandari for helpful discussions during the course of this work. The authors gratefully acknowledge the constructive comments of the anonymous reviewers.

References

Anderberg, M.R. (1973). *Cluster Analysis for Application*. Academic Press, New York.

Ankerbrandt, C.A., B.P. Buckles and F.E. Petry (1990). Scene recognition using genetic algorithms with semantic nets. *Pattern Recognition Lett.* 11, 285–293.

Belew, R. and L. Booker, Eds. (1991). *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, Los Altos, CA.

Bhandari, D., C.A. Murthy and S.K. Pal (1996). Genetic algorithm with elitist model and its convergence. *Internat. J. Pattern Recognition Artificial Intelligence*. Accepted.

Bornholdt, S. and D. Graudenz (1992). Genetic asymptotic and neural networks and structure design by genetic algorithms. *Neural Networks* 5, 327–334.

Davis, T.E. and C.J. Principe (1991). A simulated annealing like convergence theory for the simple genetic algorithm. In: (Belew and Booker, 1991), 174–181.

Devijver, P.A. and J. Kittler (1982). *Pattern Recognition: A Statistical Approach*. Prentice-Hall International, Hemel Hemstead, Hertfordshire, UK.

Goldberg, D.E. (1989). *Genetic Algorithms: Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Jain, A.K. and R.C. Dubes (1988). *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ.

Johnson, R.A. and D.W. Wichern (1982). *Applied Multivariate Statistical Analysis*. Prentice-Hall, Englewood Cliffs, NJ.

Jones, D.R. and M.A. Beltramo (1991). Solving partitioning problems with genetic algorithms. In: (Belew and Booker, 1991), 442–449.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structure = Evolution Programs*. Springer, Berlin.

Murthy, C.A., D. Bhandari and S.K. Pal (1996). Epsilon optimal stopping time for genetic algorithms with Elitist model. *IEEE Trans. Syst. Man Cybernet.*, Submitted.

Selim, S.Z. and M.A. Ismail (1984). K-means type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Trans. Pattern Anal. Mach. Intell.* 6 (1), 81–87.

Spath, H. (1980). *Cluster Analysis Algorithms*. Ellis Horwood, Chichester, UK.

Tou, T.J. and C.R. Gonzalez (1974). *Pattern Recognition Principles*. Addison-Wesley, Reading, MA.

Qi, Xiaofeng and F. Palmieri (1994). Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, Part I: Basic properties of selection and mutation. *IEEE Trans. Neural Networks* 5 (1), 102–119.