Research Journal of Information Technology

ISSN 1815-7432

Academic Journals Inc.

www.academicjournals.com

Research Journal of Information Technology 6 (2): 81-100, 2014
ISSN 1815-7432 / DOI: 10.3923/rjit.2014.81.100

Effective Evolution of Clusters: A Genetic Clustering Approach

[1]Singh Vijendra and [2]Sahoo Laxman
[1]Faculty of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh, Rajasthan, 332311, India
[2]School of Computer Engineering, KIIT University, Bhubaneswar, India

Corresponding Author: Singh Vijendra, Faculty of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh, Rajasthan, 332311, India

ABSTRACT

This study presents a new clustering algorithm called Robust Genetic Algorithm with Chromosome Reorganization (Robust GACR) based on genetic methodology. The discussion details key aspects of the proposed methodology, including a chromosome reorganization method and a new crossover operator that exploits a measure of similarity between chromosomes. Adaptive probabilities of crossover and mutation are employed to prevent the convergence of the GA to a local optimum. The performance of the Robust GACR algorithm, GCA, KGA, GA clustering and K-means algorithm are compared through the experiments based on several artificial data sets and real data sets. The K-means is unable to provide the correct clustering. However, the KGA and GCA are correctly clustered only some data sets, but they are unable to detect the correct clusters in all data sets. The Robust GACR is able to detect the clusters reasonably well in all data sets.

Key words: Clustering, genetic algorithm, Kd tree

INTRODUCTION

The organization of data using cluster analysis employs some dissimilarity measure among the set of patterns. The dissimilarity measure is defined based on the data under analysis and the purpose of the analysis. Various types of clustering algorithms have been proposed to suit different requirements. Traditional clustering algorithms primarily distinguish between hierarchical (Everitt *et al*., 2001) and partition (Xu and Wunsch, 2005). Based on the clustering criterion adopted by the algorithms (Handl *et al*., 2005), we can categorize existing clustering algorithms into the following types: The first type employs a local concept of clustering based on the idea that neighboring data points should share the same cluster (Sokal and Sneath, 1963; Defays, 1977; Lu and Fu, 1978). These methods are well suited to detect clusters of arbitrary shapes; however, they are not Robust when there is little spatial separation between the clusters. The second type methods are generally implemented by keeping intra cluster variation (i.e., variation between same-cluster data points or between data points and cluster representatives) small (MacQueen, 1967; McLachlan and Krishnan, 1997; Fraley and Raftery, 2006). These methods tend to be very effective for spherical and well-separated clusters, but they may fail for more complicated cluster structures. The third type performs simultaneous row-column clustering (Mitra and Banka, 2006; Madeira and Oliveira, 2004). The clusters identified by these algorithms are not mutually exclusive or exhaustive. The fourth type optimizes several validity measures that can capture the different characteristics of the data set (Hong *et al*., 2008; Strehl and Ghosh, 2002; Topchy *et al*., 2005).

Clustering ensembles operate with homogenous objective functions. Therefore, good clustering results may become diluted by weak results in an ensemble. For the multi objective approaches, the construction of the promising solutions set is a difficult conceptual problem, since clustering algorithms often are not accompanied by a measure of the goodness of the detected clusters. Each clustering result should be judged not only by the clustering algorithm that generated it, but also by external assessment criteria. The K-means algorithm is one of the most widely used algorithms. However, it is well known that K-means algorithm is sensitive to the initial cluster centers (MacQueen, 1967) and easy to get stuck at the local optimal solutions (Selim and Ismail, 1984). Moreover, when the number of data points is large, it takes enormous time to find the global optimal solution (Spath, 1989). In order to improve the performance of the K-means algorithm, a variety of methods have been proposed (Kanungo *et al*., 2002). In order to overcome the limitations of traditional clustering algorithms, some attempts have been made to use genetic algorithms for clustering data sets. Tseng and Yang (2001) proposed a genetic algorithm based approach to the clustering problem. Their method consists of two stages, nearest neighbor clustering and genetic optimization. Lin *et al*. (2005) presented a genetic clustering algorithm based on a binary chromosome representation. The proposed method selects the cluster centers directly from the data set. With the aid of a look-up table, the distances between all pairs of objects are saved in advance and evaluated only once throughout the evolution process. Motivated by features of genetic algorithms, in this study we propose a genetic algorithm for clustering data sets.

Kd TREE BASED NEAREST NEIGHBOR SEARCH

A K-dimensional tree or Kd-tree (Freidman *et al*., 1977) is a space-partitioning data structure for organizing points in a K-dimensional space. Kd trees are binary trees that partition the point space into hyper rectangular regions by hierarchically splitting it using axis-aligned hyper planes. Each node of a Kd tree is associated with a hyper rectangular region of the point space that it represents, with the root node being associated with a hyper rectangle comprising the entire point space. Internal nodes, in addition to the region they represent, also store information on the splitting hyper plane that splits their region and also each of the two sub-regions as children that result from the split. Leaf nodes of the tree comprise of unsplit rectangular regions of the point space called buckets. The Kd-tree data structure is based on a recursive subdivision of space into disjoint hyper rectangular regions called cells. Each node of the tree is associated with such region B, called a box and is associated with a set of data points that lie within this box. The root node of the tree is associated with a bounding box that contains all the data points. Consider an arbitrary node in the tree. As long as the number of data points associated with this node is greater than a small quantity, called the bucket size, the box is split into two boxes by an axis-orthogonal hyper plane that intersects this box. There are a number of different splitting rules which determine how this hyper plane is selected. Consider a set of n points, $(x_{1,}.., x_n)$ occupying a d dimensional space. Each point x_i has associated with it d co-ordinates $(x_{i1}, x_{i2,}..., x_{id})$. There exists a bounding box, or bucket which contains all data points and whose extrema are defined by the maximum and minimum co-ordinate values of the data points in each dimension. The data is then partitioned into two sub-buckets by splitting the data along the longest dimension of the parent bucket which we denote dm. The value at which the split is made can be determined in various ways. One method is to split the data along the median value of the co-ordinates in that dimension, median $(x_{1dm}, x_{2dm},..., x_{ndm})$, so that the number of points in each bucket remains roughly the same. Or, we may simply split at the mean in the direction of dm which requires less computation than the median.

This partitioning processes may then be recursively repeated on each sub-bucket until a leaf bucket (denoted LB) is created, at which point no further partitioning will be performed on that bucket. LB is a bucket which fulfills a certain requirement, such as, it only contains 1 data point or contains less than 10 data points. Eventually, after enough splitting of buckets, all buckets will be leaf buckets and the partitioning process will terminate. An interesting property of the Kd-tree is that each bucket will contain roughly the same number of points. However, if the data in a bucket is more densely packed than some other bucket we would generally expect the volume of that densely packed bucket to be smaller. For using Kd tree data structure for computing distance of the Eq. 3 in proposed algorithm Genetic Algorithm with Chromosome Reorganization (Robust GACR), ANN (Mount and Arya, 2010) is used. ANN is a library written in C^{++} which supports data structures and algorithms for both exact and ANN is searching in arbitrarily high dimensions. The distance computation of the Eq. 3 is time consuming that can be speeded up by using the Kd-tree based nearest neighbor search.

GENETIC ALGORITHM WITH CHROMOSOME REORGANIZATION

Here, a new clustering algorithm based on the proposed chromosome reorganization method and GA is described in details. It includes the determination of the number of clusters as the appropriate clustering of the data set. This genetic clustering technique is referred as Genetic Algorithm with Chromosome Reorganization (Robust GACR). The basic steps of proposed algorithm Robust GACR is presented in Fig. 1.

Chromosome representation: Real-coded genetic algorithms use floating-point or other high cardinality encoding in their chromosomes (Hruschka *et al.*, 2009). Real-coded (non-binary)

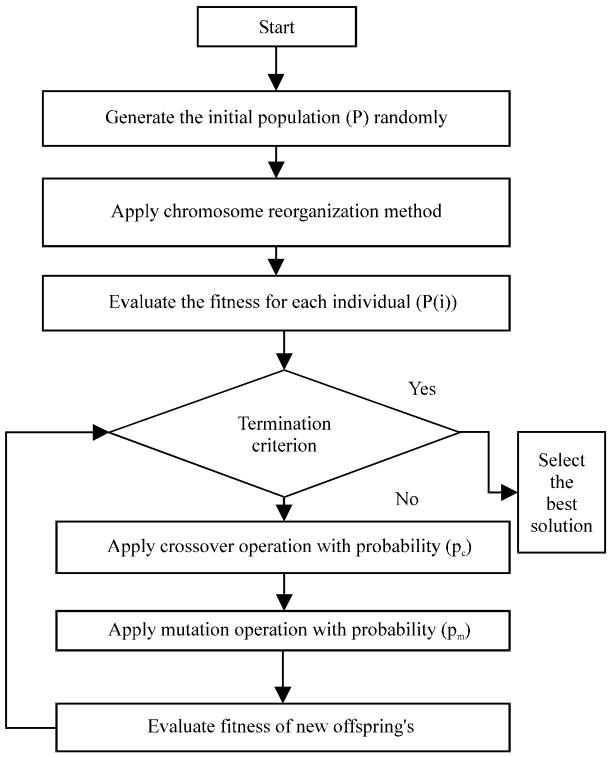


Fig. 1: Steps of proposed algorithm Robust GACR

representations are more natural for the specific problems. In a numerical optimization on continuous domain a chromosome is represented as a vector of floating point numbers in which each number corresponds to a variable in the problem. Real-coded GAs usually adopts mutation operators that perturb the current solution around the current value and are well suited for hill climbing in the decision space under the consideration. In these kinds of situations, binary coded genes can easily become stuck on Hamming cliffs. Under the assumptions of a fixed population size, a fixed number of search alternatives and serial processing of individual loci, it can be shown theoretically and empirically that higher cardinality alphabets converge to a solution more quickly than those coded over a smaller alphabet (Goldberg, 1989). Genetic representations for clustering or grouping problems are based on underlying scheme. The scheme represents the objects with gene values and the position of these genes signifies how the objects are divided amongst the clusters. The use of a simple encoding scheme causes problems of redundant codification and context insensitivity (Hruschka *et al*., 2004). This has led researchers to devise complicated representations and specialized operators for clustering problems. The cluster label based on n bit encoding is simple compared to parameterization of prototype location. In such a representation many genotypes translate to a unique phenotype. The notion of cluster labels built into the representation makes little intuitive sense. Such representations have spawned off a set of pre treatment methodologies to make the representations suitable for genetic operators. To consider encoding scheme of Robust GACR, let us consider a dataset formed by N objects. Then, a chromosome is an integer vector of (N) positions. Each position corresponds to an object, i.e., the i_{th} position (gene) represents i-th object. For example a dataset is composed of 19 objects, as given in Table 1.

The objects have two attributes each denoted by Attribute 1 and 2. Such objects have been arbitrarily grouped in four clusters. These clusters are depicted in Fig. 2 and a chromosome is represented as given below.

Table 1: Sample data set1

Objects (x_i)	Attribute 1	Attribute 2	Cluster No.
x_1	4.0	13.0	1
x_2	6.0	11.0	1
x_3	4.0	16.0	1
x_4	14.0	18.0	1
x_5	7.0	13.0	1
x_6	8.0	18.0	1
x_7	12.0	19.0	1
x_8	14.0	17.0	1
x_9	25.0	6.0	2
x_{10}	26.0	14.5	2
x_{11}	27.0	16.5	3
x_{12}	26.0	17.0	3
x_{13}	30.0	4.0	4
x_{14}	27.0	4.0	4
x_{15}	29.0	7.0	2
x_{16}	29.5	12.0	2
x_{17}	30.0	16.5	3
x_{18}	31.0	21.0	3
x_{19}	31.5	6.0	4

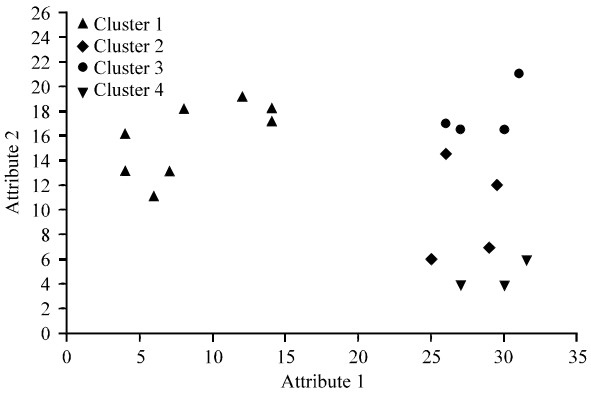


Fig. 2: Chromosome representation

A possible chromosome is:

Chromosome: 1111111122334422334

In this example, eight objects {1-8} form the cluster whose label is 1. The cluster whose label is 2 has 4 objects {9, 10, 15 and 16}, cluster label 3 has 4 objects {11, 12, 17 and 18} and 3 objects {13, 14 and 19} form the cluster 4.

Standard genetic operators are usually not suitable for clustering problems for several reasons. First, the encoding scheme presented above is naturally redundant, i.e., the encoding is one-to-many. In fact, there is k! (k represents number of clusters) different chromosomes that represent the same solution (Hruschka *et al.*, 2004). For example, there are 3! different chromosomes which represent same clustering solutions as (1 1 2 2 3 3), (2 2 3 3 1 1), (3 3 2 2 1 1), (2 2 1 1 3 3), (3 3 1 1 2 2), (1 1 3 3 2 2). Thus, the size of the search space in the genetic algorithm is much larger than the original space of solutions. This augmented space may reduce the efficiency of the genetic algorithm. In addition, the redundant encoding also causes the undesirable effect of casting context-dependent information out of context under the standard crossover, i.e., equal parents can originate different offspring. For example, if the simple one-point crossover operator is executed on the chromosome (1 1 2 2 3 3) and the chromosome (3 1 1 3 2 2), both new clustering solutions (1 1 2 2 2 2) and (3 1 1 3 3 3) have only two clusters and both of them are invalid. These problems are solved by the Chromosome Reorganization (CR) method. The pseudo code of CR method is given in Fig. 3.

Fitness function: The fitness function f of the chromosome is computed as the inverse of SSE (sum of squared error):

$$f = \frac{1}{SSE} \tag{1}$$

The fitness function will be maximized during the evolutionary process of Robust-GACR and lead to minimization of the SSE. The SSE is defined as:

```
Let C1 = {1, 2, ... , K} and C2 = ϕ (empty)
// chromosome P1 is consist of K clusters
Set i = 1
for j= 1 to N // N- number of genes in a chromosome
if C2(j) = 0 then
Set m = P1(j)
for n = 1 to N
if (C1(n) = m and C2(n) = 0)
C1(n) = i and C2(n) = 1
end_if
end_for
i = i+1
end_if
end_for
```

Fig. 3: Pseudo code of the chromosome reorganization method

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} (x - m_i)^T (x - m_i) \tag{2}$$

where, m_i is the center of the ith cluster. This measure computes the cumulative distance of each data point from its cluster center of each cluster individually and then sums of those measures overall clusters:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} \| x - m_i \|^2 \tag{3}$$

where, $x \epsilon C_i$ is a data point assigned to that cluster and k denotes number of clusters. If this measure is small, then the distances from data points to cluster centers are all small and the clustering would be regarded favorably. The sum of squared error has a theoretical minimum of zero which corresponding to all clusters containing only a single data point.

Genetic operators

Selection: The selection operator supports the choice of the individuals from a population that will be allowed to mate in order to create a new generation of individuals (Kaufman and Rousseeuw, 1990). GA methods attempt to develop fitness methods and elitism rules that find a set of Pareto optimal values quickly and reliably. But, ultimately it is the selection method that is responsible for the choice of a new generation of parents. Thus, the selection process is responsible for making the mapping from the fitness values of the input population to a probability of selection and thus the probability of an individual's genetic material being passed to the next generation. Therefore, to fully understand the probability of selection for fitness values, in this study we adapt roulette selection method (Goldberg and Deb, 1991).

Crossover: Crossover is a probabilistic process that exchanges information between two parent chromosomes for generating two child chromosomes (Holland, 1975). The classical crossover operator can not perform well enough due to the problems of clustering invalidity and context

insensitivity. It may be necessary to check that offspring produced by a certain operator are valid and reject any invalid chromosomes. In this algorithm a new relation based crossover is presented. The relation based crossover works by building a relation between two parent's chromosomes. The representation of chromosome 4 and 5 is shown in Fig. 4 and 5, respectively. It works in the following way. First, two chromosomes are selected:

Chromosome 4:11223344

Chromosome 5:11232334

Then, assuming that cluster 2 and 3 of chromosome 4 are randomly selected by Robust GACR. These clusters are copied into corresponding genes of chromosome 5:

Chromosome 5:11223334

The clusters of unchanged genes of chromosome 5 are placed based on relation between selected clusters of chromosome 4 and clusters of corresponding genes of chromosome 5 by applying this procedure, we get new child chromosome 6 (Fig. 6):

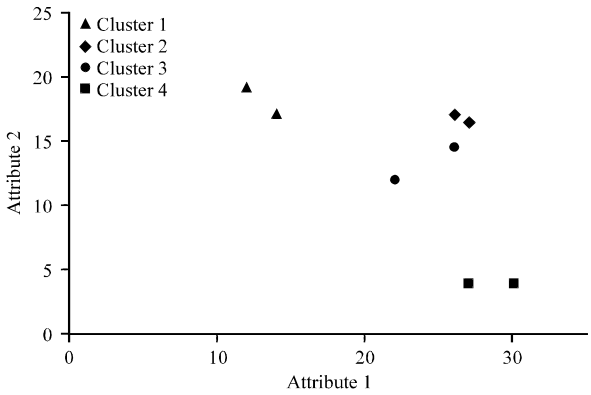


Fig. 4: Representation of chromosome 4

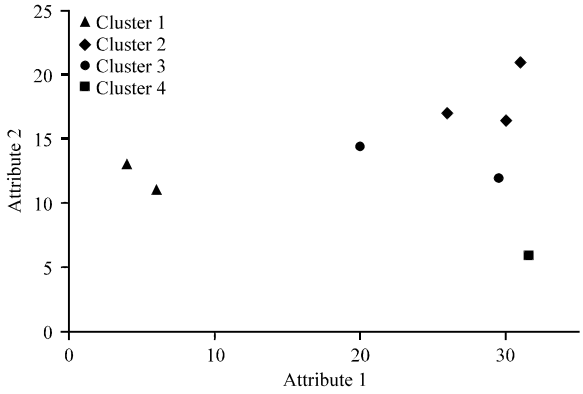


Fig. 5: Representation of chromosome 5

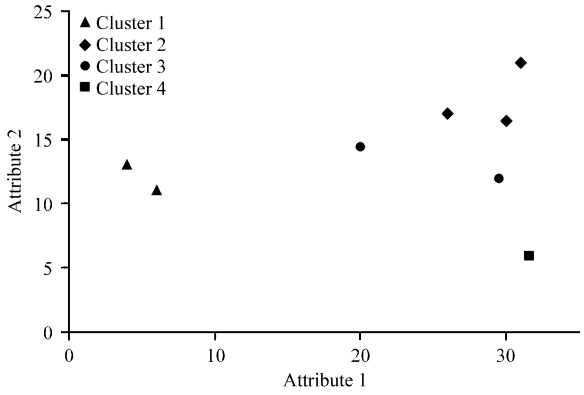


Fig. 6: Representation of chromosome 6

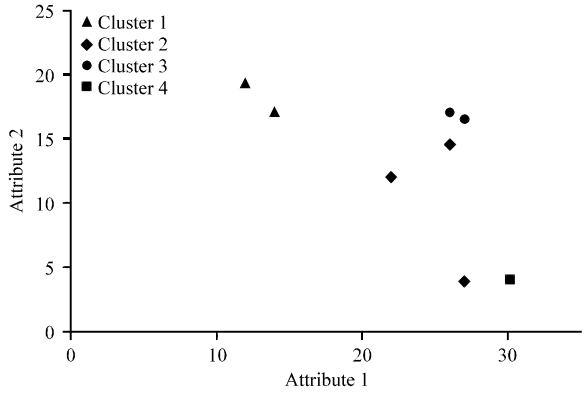


Fig. 7: Representation of chromosome 7

2→3, 3→2

Chromosome 6:11323224

The same procedure is applied to get child chromosome 7 (Fig. 7), but now considering that the changed clusters of chromosome 5 are copied into chromosome 4.

Chromosome 7:11332234

Crossover probability is selected adaptively as in (Srinivas and Patnaik, 1994). The expressions for crossover probabilities are computed as follows. Let f_{max} be the maximum fitness value of the current population, $\bar{f}$ be the average fitness value of the population and f' be the larger of the fitness values of the solutions to be crossed. Then the probability of crossover, p_c, is calculated as:

$$p_c = k_1 \times \frac{f_{max} - f'}{f_{max} - \bar{f}}, \text{ if } f' > \bar{f} \tag{4}$$

$$p_c = k_3, \text{ if } f' \leq \bar{f} \tag{5}$$

Here, as in Srinivas and Patnaik (1994), the values of k_1 and k_3 are kept equal to 1.0. Note that, when $f_{max} = \bar{f}$, then $f' = f_{max}$ and p_c will be equal to k_3. The aim behind this adaptation is to achieve a trade-off between exploration and exploitation in a different manner. The value of pc is increased when the better of the two chromosomes to be crossed is itself quite poor. In contrast when it is a good solution, p_c is low so as to reduce the likelihood of disrupting a good solution by crossover.

Mutation: Mutation introduces new genetic material into the population. In a clustering context this corresponds to moving an object from one cluster to another. Two operators for mutation are used. First mutation operator eliminates a selected cluster by placing its objects into the nearest clusters. Let us consider chromosome C1 which encodes 4 clusters in a data set composed of nineteen objects as shown in Fig. 8:

C1: 1111111122334422334

Suppose cluster 2 has been selected by mutation operator. Two objects of cluster 2 are closer to cluster 3, where two bottom objects of cluster 2 are closer to cluster 4, as shown in Fig. 9. The mutated chromosome is C2 as shown in Fig. 10:

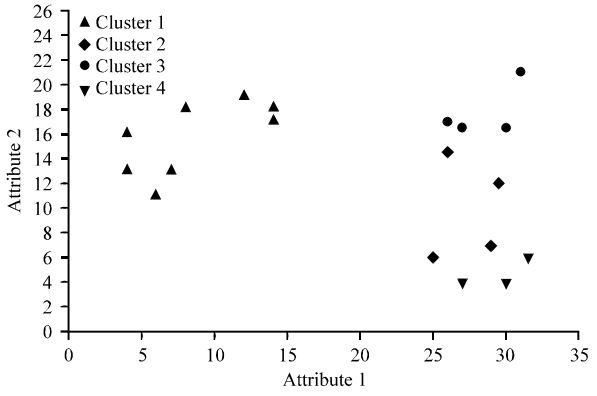


Fig. 8: Clustering encoded by chromosome C1

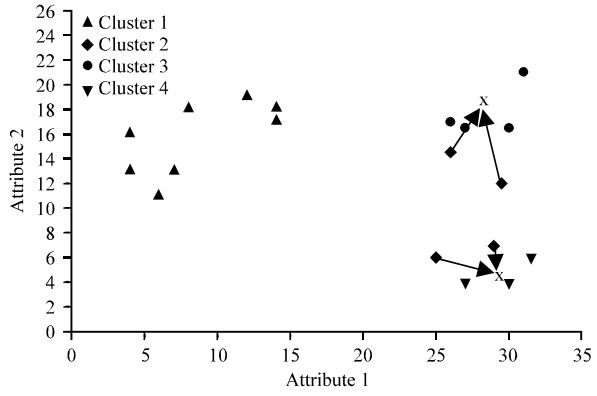


Fig. 9: Mutation operation on chromosome C1

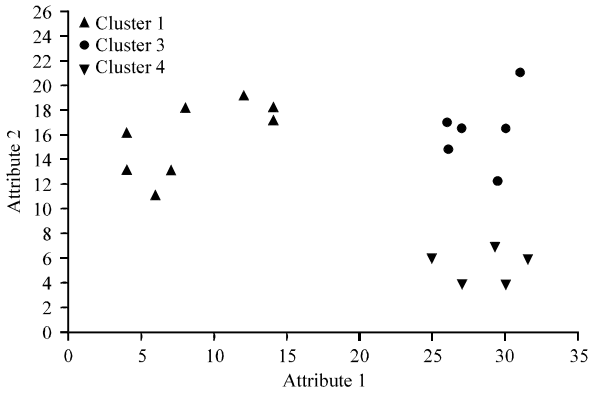


Fig. 10: Clustering solutions after mutation by first mutation operator

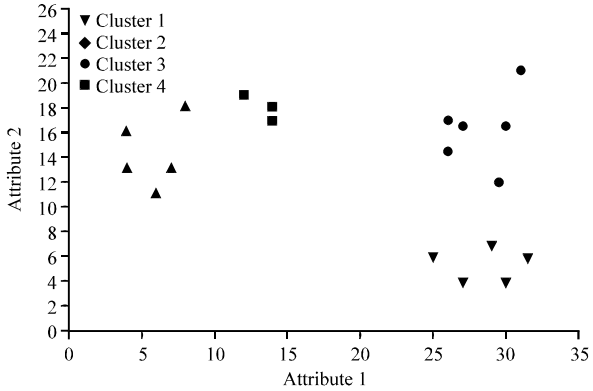


Fig. 11: Clustering result after second mutation operator

C2: 111111133334433444

The second mutation operator splits a selected cluster into two new ones. The first cluster is created by objects nearest to the centroid. The second cluster is created by those objects nearest to the farthest object from the centroid. Let us suppose that cluster1 is selected by the mutation operator two. Cluster1 is then split into two clusters as shown in Fig. 11. The resulting chromosome is given below:

C3: 1111122233334433444

The first operator works only on chromosomes that encode more than two clusters. It eliminates a randomly chosen cluster, placing its objects into the nearest remaining clusters (according to their centroids). The second operator divides a randomly selected cluster into two new ones. The first cluster is formed by the objects closer to the original centroid, whereas the other cluster is formed by those objects closer to the farthest object from the centroid.

Each chromosome undergoes mutation with a probability p_m. The mutation probability is also selected adaptively for each chromosome as in Srinivas and Patnaik (1994). The expression for mutation probability, p_m is given below:

$$p_m = k_2 \times \frac{f_{max} - f}{f_{max} - \bar{f}}, \text{ if } f > \bar{f}$$
$$p_m = k_4, \text{ if } f \leq \bar{f} \quad (6)$$

Here, the values of k_2 and k_4 are kept equal to 0.5. This adaptive mutation helps GA to come out of local optimum. When GA converges to a local optimum, i.e., the value of $f_{max} - \bar{f}$ decreases, p_c and p_m both will be increased. As a result GA will come out of local optimum. It will also happen for the global optimum and may result in disruption of the near-optimal solutions. As a result GA will never converge to the global optimum. But as p_c and p_m will get lower values for high fitness solutions and get higher values for low fitness solutions while the high fitness solutions aid in the convergence of the GA, the low fitness solutions prevent the GA from getting stuck at a local optimum. The use of elitism will also keep the best solution intact. For a solution with the maximum fitness value, p_c and p_m are both zero. The best solution in a population is transferred undisrupted into the next generation. Together with the selection mechanism, this may lead to an exponential growth of the solution in the population and may cause premature convergence. To overcome the above stated problem, a default mutation rate (of 0.01) is kept for every solution in the Robust GACR. We have used the mutation operation similar to that used in GA based clustering (Goldberg, 1989). In Robust GACR, the best string seen up to the last generation provides the solution to the clustering problem. Elitism has been implemented at each generation by preserving the best string seen up in that generation in a location outside the population (Thierens and Goldberg, 1994). Thus on termination, this location contains the centers of the final clusters.

Termination criterion: We have executed Robust GACR algorithm for a fixed number of generations. The best string of the last generation provides the solution to the clustering problem. Moreover, the elitist model of GAs has been used, where the best string seen so far is stored in a location within the population.

EXPERIMENTAL EVALUATION

The experimental results comparing the performance of K-means, GA clustering (Murthy and Chowdhury, 1996), KGA (Bandyopadhyay and Maulik, 2002), GCA and Robust GACR are provided for the two artificial and three real-life data sets. In order to show the efficacy of the proposed GCA (Vijendra *et al.*, 2010) and Robust GACR clustering algorithms over existing single objective clustering techniques, two developed genetic clustering algorithms, GA clustering (Murthy and Chowdhury, 1996) and KGA clustering (Bandyopadhyay and Maulik, 2002), are also executed on the two artificial and three real-life data sets. These single objective genetic clustering techniques provide a single solution after their execution. KGA clustering technique optimizes a Euclidean distance based cluster objective function by using the search capability of genetic algorithms to determine the appropriate partitioning from data sets. The parameters of the GA clustering and KGA clustering techniques are as follows: Population size = 40, No. of generations = 200. All experiments were run on a PC with a 2.0 GHz processor and 2 GB RAM. In the experiments, the population size of proposed GCA and Robust GACR is taken as 40 and number of generations = 200. The crossover and mutation probabilities for Robust GACR clustering is $p_c = 0.8$ and $p_m = 0.001$, respectively.

Artificial data sets: In order to evaluate the performance of the proposed Robust GACR with other clustering algorithms, GCA, KGA, GA clustering and K-means more objectively, some artificial data sets are generated and performed clustering by using the proposed algorithm:

- **Data set 1:** This is a 5 dimensional data set consists of three classes of 400 data points. This dataset is to be clustered into 4 clusters
- **Data set 2:** This is a 10-dimensional data set consists of seven clusters of 2000 data points. This dataset is to be clustered into seven clusters

Real data sets: The performance of the Robust GACR, GCA, KGA, GA clustering algorithm and K-means algorithm are compared through the experiments on three real data sets. The real data sets are obtained from UCI repository (http://archive.ics.uci.edu/ml/):

- **Iris:** Iris data set consists of 150 data points distributed over three clusters. Each cluster has 50 points. This data set represents different categories of irises characterized by four feature values. It has three classes Setosa, Versicolor and Virginica among which the last two classes have a large amount of overlap while the first class is linearly separable. The sepal and petal areas were used as attributes (variables). The sepal area is obtained by multiplying the sepal length by the sepal width and the petal area is calculated in an analogous way
- **Cancer:** Wisconsin Breast cancer data set consisting of 683 sample points. Each pattern has nine features corresponding to clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. There are two categories in the data: malignant and benign. The two classes are known to be linearly separable
- **Wine:** This is the Wine recognition data consisting of 178 objects having 13 features resulting from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines

Clustering results analysis: Cluster validation refers to quantitatively evaluate the quality of a clustering solution. Validating a clustering solution is important since bad clustering arise in many common situations: for instance, as a result of an inappropriate choice of parameter values, unsuccessful initialization of the algorithm, or working on a data set that does not contain cluster structure. The experimental clustering results of K-means, GA clustering and KGA, GCA and Robust GACR algorithms are evaluated using Rand Index (Rand, 1971). Performance results are shown in Table 2. The discussion on experimental results is given below:

- **Data set 1:** As can be seen from Table 2, both the proposed GCA and Robust GACR are able to detect the appropriate number of clusters and the proper partitioning from this data set. The corresponding partitioning is shown in Fig. 12d and e. K-means fails to detect the appropriate number of clusters and the appropriate partitioning. The corresponding partitioning is shown in Fig. 12a. GA clustering algorithm is also not able to detect the appropriate partitioning. The partitioning provided by GA and KGA are shown in Fig. 12b, c, respectively

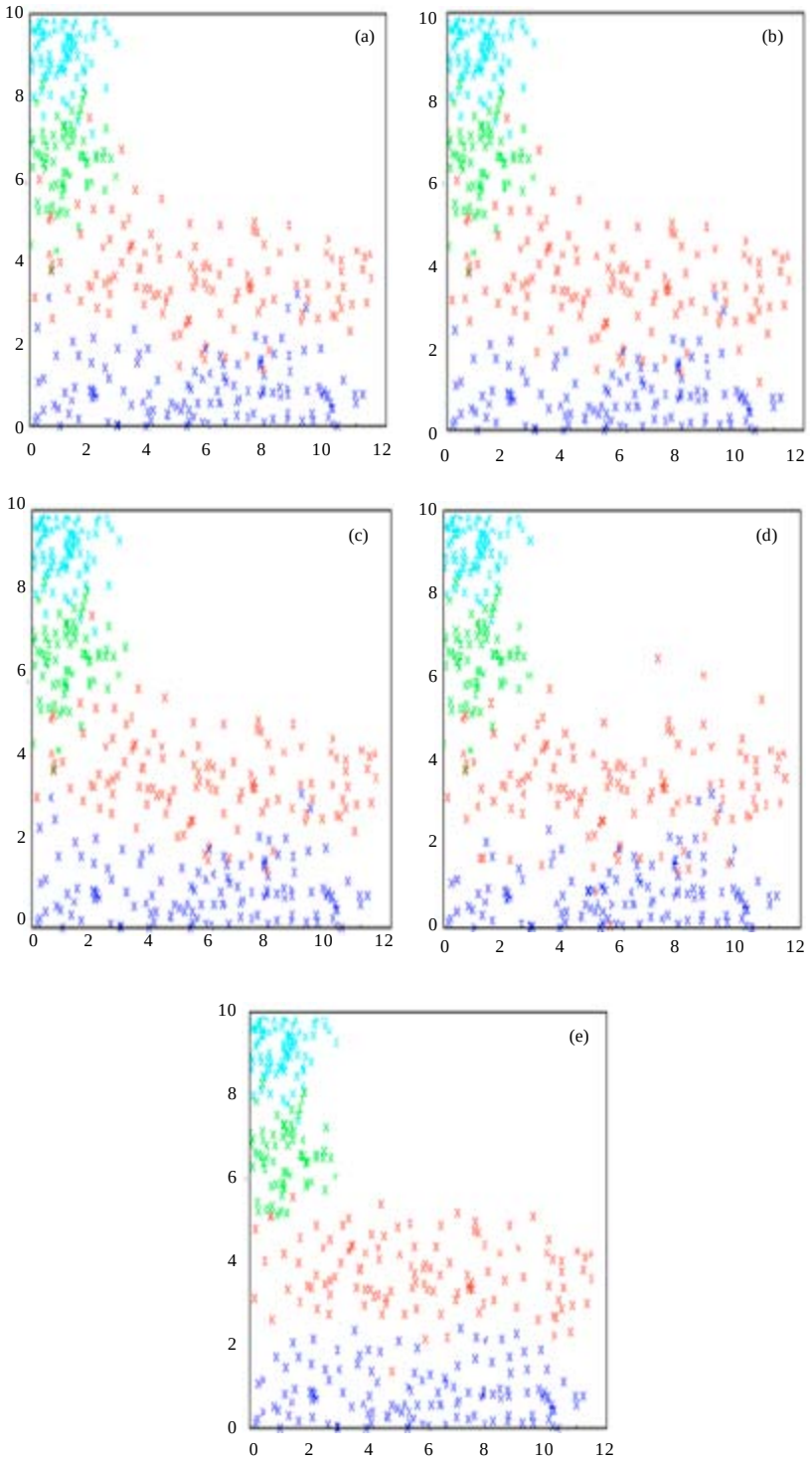


Fig. 12(a-e): Result of clustering of data set 1 by (a) K-means of data set1, (b) GA, (c) KGA, (d) GCA and (e) Robust GACR

Table 2: Accuracy of K-means, GA, KGA, GCA and robust GACR (rand index)

	Accuracy of clustering in percentage (rand index value)				
Data sets	K-means	GA	KGA	GCA	Robust GACR
Data set 1	78.2	90.5	95.5	95.7	96.6
Data set 2	68.0	76.7	80.5	81.0	95.0
Iris	87.4	92.6	97.2	97.0	99.0
Cancer	85.5	93.5	94.1	94.2	96.8
Wine	86.0	91.8	92.4	92.5	95.8

- **Data set 2:** As seen from Table 2, the genetic based clustering technique Robust GACR is able to detect the proper number of clusters and the proper partitioning from this data set. The corresponding partitioning is shown in Fig. 13e. K-means again merges the two overlapping clusters into one cluster and unable to provide the optimal number of clusters. The corresponding partitioning is shown in Fig. 13a. GA clustering technique identifies total K = 6 number of clusters from this data set. The corresponding partitioning is shown in Fig. 13b. The Rank Index values reported in Table 2 also show the poorer performance of both K-means and GA clustering. KGA and GCA clustering algorithms detect the overlapping clusters from this data set and the corresponding partitioning is very near to the actual partitioning of the data set (refer to Table 2). The corresponding partitioning is shown in Fig. 13c and d
- **Iris:** For this real-life data set, only Robust GACR clustering technique is able to determine the appropriate number of clusters. The corresponding Rank Index is also the higher value (Table 2). Other four clustering algorithms, proposed GCA, KGA, GA clustering and K-means, provide K = 3 as the optimal number of clusters which is also often obtained for many other methods for Iris data set. The partitioning provided by K-means, GA clustering, KGA, GCA and Robust GACR are shown in Fig. 14a-e, respectively
- **Cancer:** For this data set, Rank Index (34) of the partitioning obtained after application of all five algorithms. The Rank Index values are 0.85, 0.93, 0.94, 0.94 and 0.96 for K-means, GA clustering, KGA, GCA and Robust GACR, respectively. From Rank Index analysis it was observed that GCA and KGA algorithms perform equally for this indicating that the two clusters present in the Cancer data set are convex as well as highly symmetrical. They are able to find the best clustering among all the algorithms. As can be noted from Table 2, Rank Index indicates Robust GACR as the most appropriate choices. Rank Index is also successful in correctly identifying Robust GACR algorithms as the most appropriate clustering algorithms
- **Wine:** From Table 2, it is evident that Robust GACR performs the best (providing the highest Rank Index value) while K-means performs the worst. The improvement in performance obtained by Robust GACR as compared to each of the other four clustering techniques is significant. GCA and KGA are also found to provide improved performance over K-means and GA clustering and these improvements are also significant

It can be seen from the above results that the proposed Robust GACR clustering algorithm is able to detect the appropriate partitioning and the appropriate number of clusters from most of the data sets used here for the experiments. It outperforms another KGA clustering technique, GA clustering and K-means clustering algorithm. The superiority of Robust GACR is also established on three real-life data sets which are of different characteristics with the number of dimensions

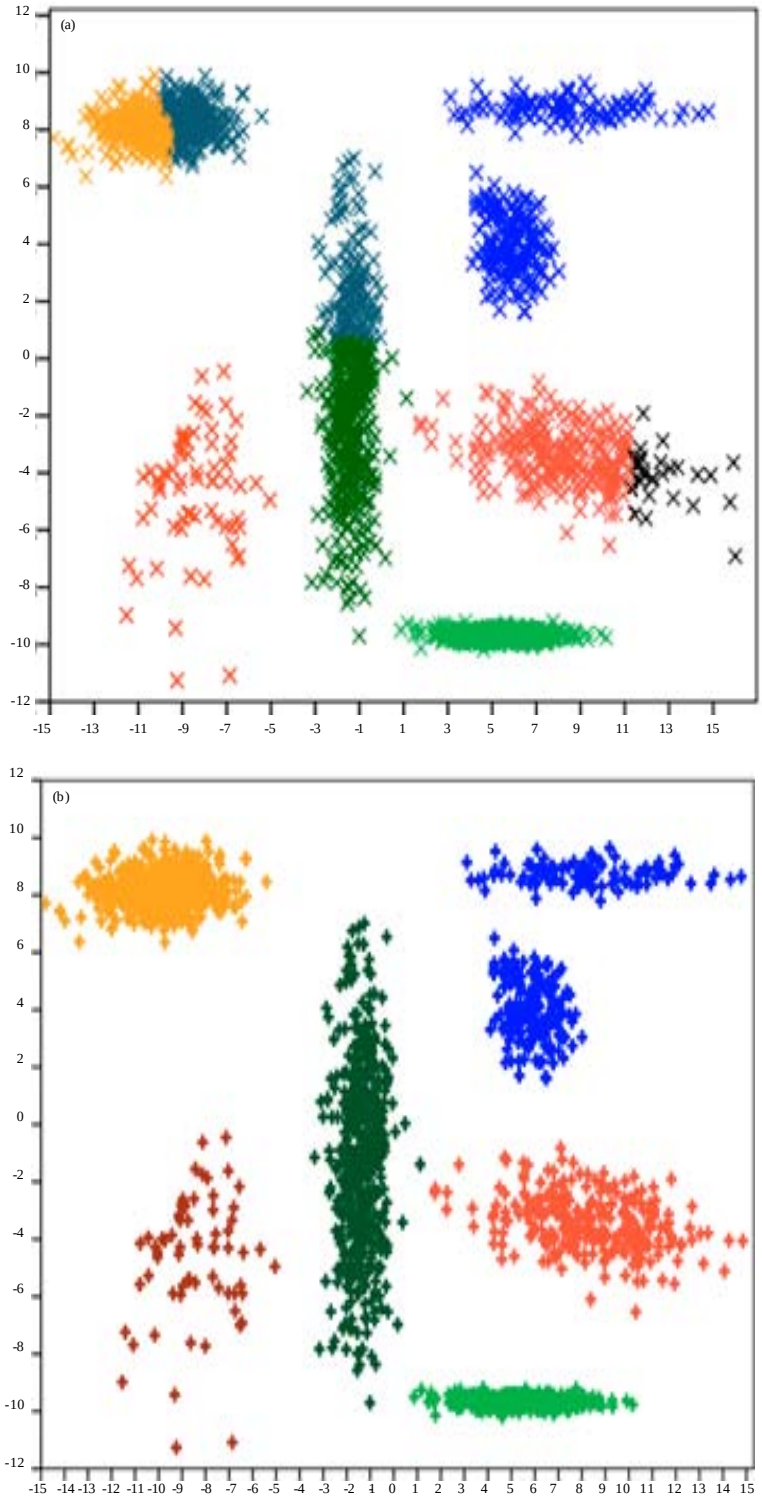
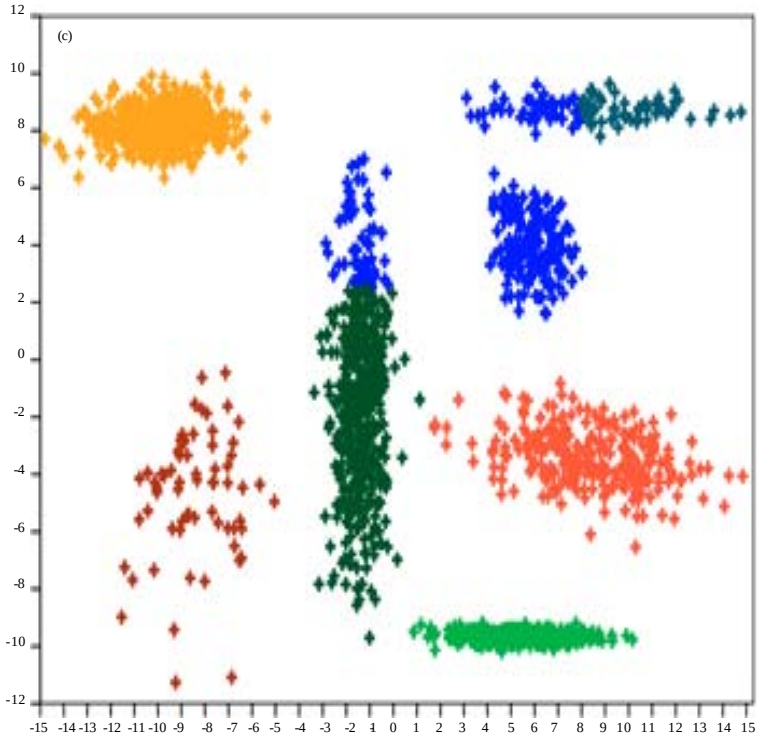


Fig. 13(a-e): Continue



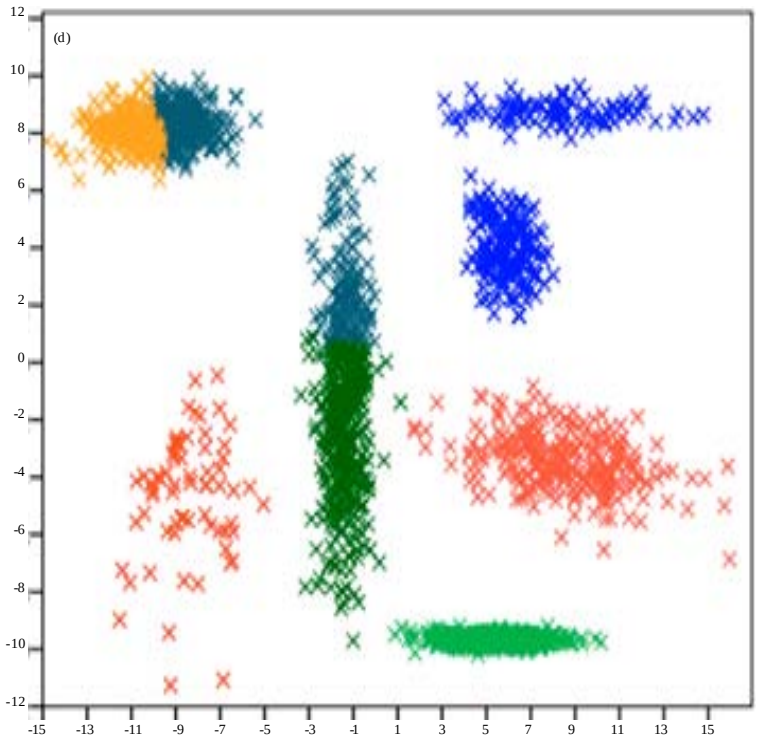


Fig. 13(a-e): Continue

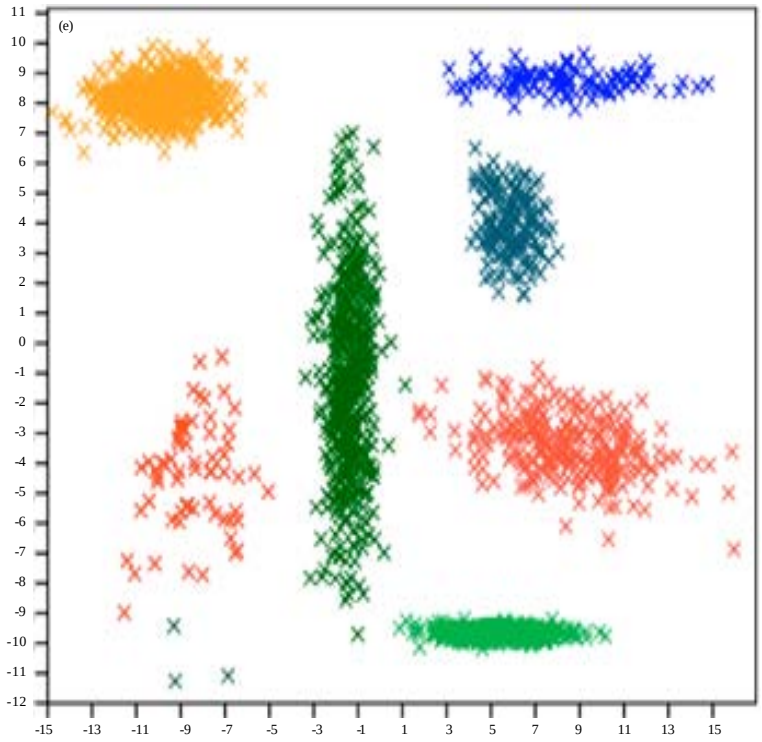


Fig. 13(a-e): Result of clustering of data set 2 by (a) K-means, (b) GA, (c) KGA, (d) GCA and (e) Robust GACR

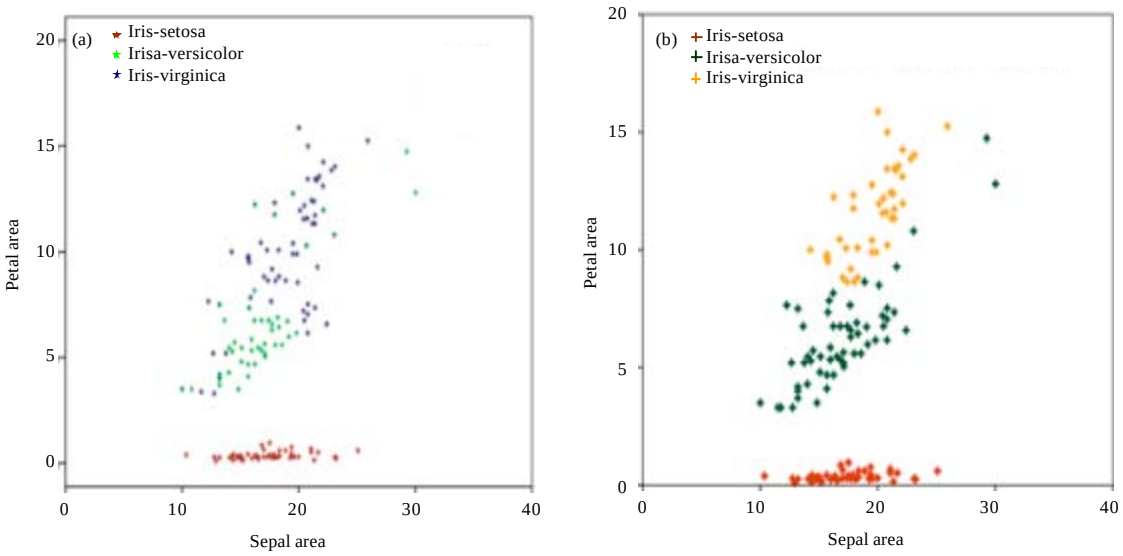


Fig. 14(a-e): Continue

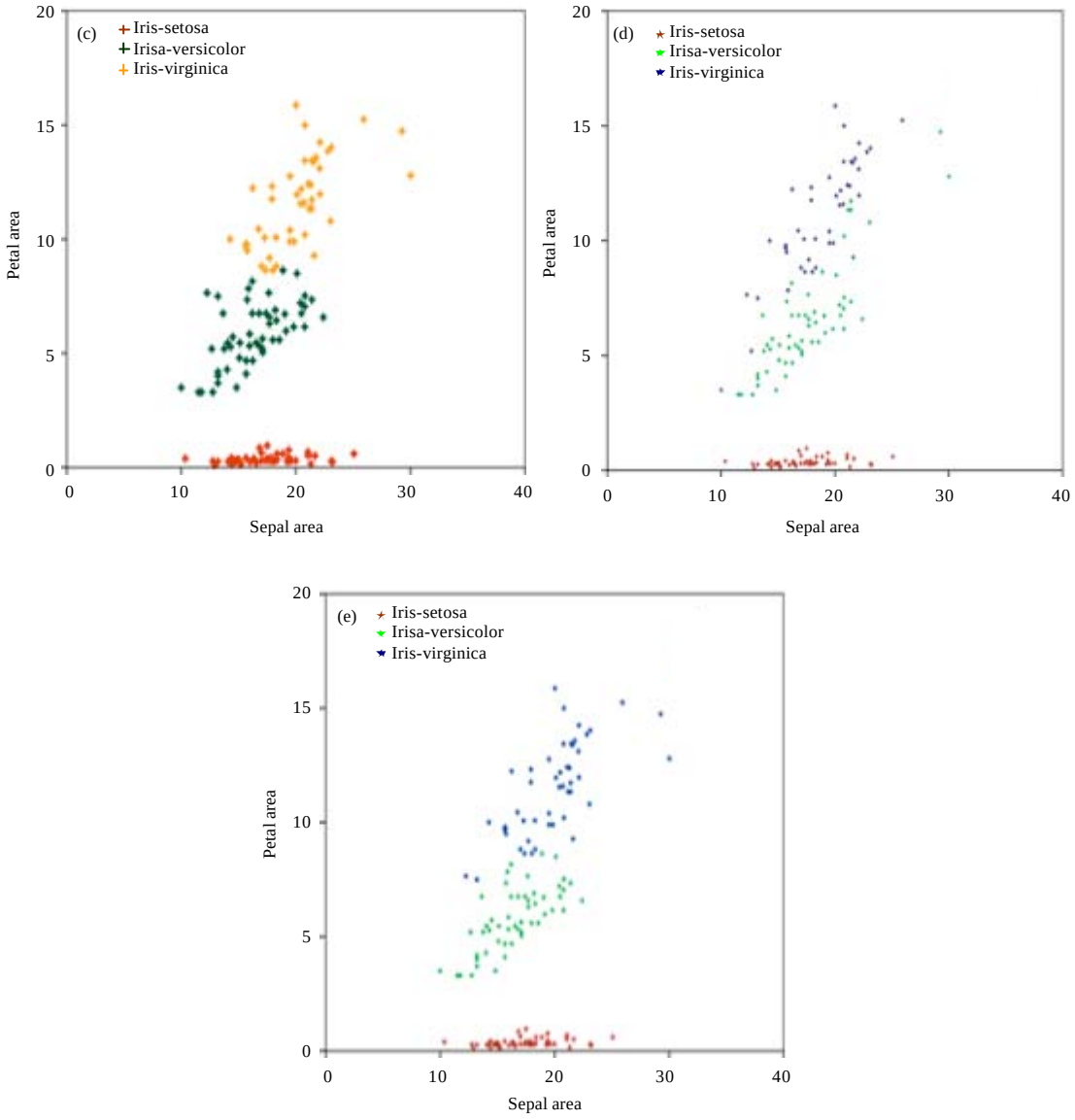


Fig. 14(a-e): Result of clustering of Iris data set by (a) K-means, (b) GA, (c) KGA, (d) GCA and (e) Robust GACR

varying from 4-13. Results on the two artificial and three real-life data sets establish the fact that Robust GACR is well-suited to detect the number of clusters from data sets having clusters of widely varying characteristics. The performance of proposed GCA is also superior compare to GA clustering and K-means. We have also experimented with KGA and GA clustering algorithms on artificial and real-life data sets. We found that for data set1 and 2, GCA and KGA were able to detect the appropriate number of clusters. The performance results reported in Table 2, clearly demonstrate the clustering accuracy of K-means, GA clustering, KGA, GCA and Robust GACR for artificial and real data sets.

CONCLUSION

In this study, we presented a new clustering algorithm called Robust GACR (Robust-Genetic Algorithm with Chromosome Reorganization). To overcome the problems of clusters invalidity and context insensitivity, we presented a chromosome reorganization method which may effectively remove the degeneracy for the purpose of more efficient search. A new crossover operator that exploits a measure of similarity between chromosomes is also presented. The Kd-tree based nearest neighbor search is used to reduce the complexity of finding the closest points. Adaptive probabilities of crossover and mutation are employed to prevent the convergence of the genetic algorithm to a local optimum. The performance of the Robust GACR algorithm, GCA, KGA, GA clustering and K-means algorithm are compared through the experiments based on several artificial data sets and real data sets. The K-means is unable to provide the correct clustering. However, the KGA and GCA are correctly clustered only some data sets but they are unable to detect the correct clusters in all data sets. The Robust GACR is able to detect the clusters reasonably well in all data sets.

REFERENCES

Bandyopadhyay, S. and U. Maulik, 2002. An evolutionary technique based on K-means algorithm for optimal clustering in RN. Inform. Sci., 146: 221-237.

Defays, D., 1977. An efficient algorithm for a complete link method. Comput. J., 20: 364-366.

Everitt, B., S. Landau and M. Leese, 2001. Cluster Analysis. Arnold, London.

Fraley, C. and A.E. Raftery, 2006. MCLUST version 3 for R package for normal mixture modeling and model-based clustering. Department of Statistics, University of Washington, Seattle, USA., September 2006. http://www.stat.washington.edu/research/reports/2006/tr504.pdf.

Freidman, J.H., J.L. Bentley and R.A. Finkel, 1977. An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Software, 3: 209-226.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. 1st Edn., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA., ISBN-13: 978-0201157673.

Goldberg, D.E. and K. Deb, 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In: Foundations of Genetic Algorithms, Rawlins, G.J.E. (Ed.). Morgan Kaufmann Publishers, Inc., San Francisco, CA., USA., ISBN-13: 978-1558601703, pp: 69-93.

Handl, J., J. Knowles and D.B. Kell, 2005. Computational cluster validation in post-genomic data analysis. Bioinformatics, 21: 3201-3212.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. 1st Edn., University of Michigan Press, Ann Arbor, Michigan, ISBN: 0472084607.

Hong, Y., S. Kwong, Y. Chang and Q. Ren, 2008. Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. Pattern Recognit., 41: 2742-2756.

Hruschka, E.R., R.J.G.B. Campello and L.N. de Castro, 2004. Improving the Efficiency of A Clustering Genetic Algorithm. In: Advances in Artificial Intelligence, IBERAMIA 2004, Lemaitre, C., C.A. Reyes and J.A. Gonzalez (Eds.). Vol. 3315, Springer, Germany, pp: 861-870.

Hruschka, E.R., R.J.G.B. Campello, A.A. Freitas and A.P.L.F. de Carvalho, 2009. A survey of evolutionary algorithms for clustering. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev., 39: 133-155.

Kanungo, T., D.M. Mount, N.S. Netanyahu, C.D. Piatko, R.S. Angela and Y. Wu, 2002. An efficient k-means clustering algorithm: Analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell., 24: 881-892.

Kaufman, L. and P.J. Rousseeuw, 1990. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley and Sons, New York, ISBN: 0471878766, pp: 342.

Lin, H.J., F.W. Yang and Y.T. Kao, 2005. An efficient GA based clustering technique. Tamkang J. Sci. Eng., 8: 113-122.

Lu, S.Y. and K.S. Fu, 1978. A sentence-to-sentence clustering procedure for pattern analysis. IEEE Trans. Syst. Man Cybern., 8: 381-389.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1, January 17-20, 1967, University of California Press, USA., pp: 281-297.

Madeira, S.C. and A.L. Oliveira, 2004. Biclustering algorithms for biological data analysis: A survey. IEEE/ACM Trans. Comput. Biol. Bioinform., 1: 24-45.

McLachlan, G.J. and T. Krishnan, 1997. The EM Algorithm and Extensions. 1st Edn., Wiley, New York, ISBN: 9780471201700.

Mitra, S. and H. Banka, 2006. Multi-objective evolutionary biclustering of gene expression data. Pattern Recogn., 39: 2464-2477.

Mount, D.M. and S. Arya, 2010. ANN: A library for approximate nearest neighbor searching: Version 1.1.2. January 27, 2010. http://www.cs.umd.edu/~mount/ANN/.

Murthy, C.A. and N. Chowdhury, 1996. In search of optimal clusters using genetic algorithms. Pattern Recog. lett., 17: 825-832.

Rand, W.M., 1971. Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc., 66: 846-850.

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: 81-87.

Sokal, R.R. and P.H.A. Sneath, 1963. Principles of Numerical Taxonomy. W.H. Freeman, San Francisco, CA, USA., Pages: 359.

Spath, H., 1989. Cluster Analysis Algorithms. Ellis Horwood, Chichester, UK.

Srinivas, M. and L.M. Patnaik, 1994. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybernet., 24: 656-667.

Strehl, A. and J. Ghosh, 2002. Cluster ensembles: A knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res., 3: 583-617.

Thierens, D. and D. Goldberg, 1994. Elitist recombination: An integrated selection recombination GA. Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, June 27-29, 1994, Orlando, FL., pp: 508-512.

Topchy, A., A.K. Jain and W. Punch, 2005. Clustering ensembles: Models of consensus and weak partitions. IEEE Trans. Pattern Anal. Mach. Intell., 27: 1866-1881.

Tseng, L.Y. and S.B. Yang, 2001. A genetic approach to the automatic clustering problem. Pattern Recognit., 34: 415-424.

Vijendra, S., L. Sahoo and K. Ashwini, 2010. An effective clustering algorithm for data mining. Proceedings of the International Conference on Data Storage and Data Engineering, February 9-10, 2010, Bangalore, India, pp: 250-253.

Xu, R. and D. Wunsch, 2005. Survey of clustering algorithms. IEEE Trans. Neural Networks, 16: 645-678.