Improved Crisp and Fuzzy Clustering Techniques for Categorical Data*

Indrajit Saha † and Anirban Mukhopadhyay ‡

Abstract—**Clustering is a widely used technique in data mining application for discovering patterns in underlying data. Most traditional clustering algorithms are limited in handling datasets that contain categorical attributes. However, datasets with categorical types of attributes are common in real life data mining problem. For these data sets, no inherent distance measure, like the Euclidean distance, would work to compute the distance between two categorical objects. In this article, we have described two algorithms based on genetic algorithm and simulated annealing in the field of crisp and fuzzy domain. The performance of the proposed algorithms has been compared with that of different well known categorical data clustering algorithms in crisp and fuzzy domain and demonstrated for a variety of artificial and real life categorical data sets. Also statistical significance tests have been performed to establish the superiority of the proposed algorithms.**

Keywords: Genetic Algorithm based Clustering, Simulated Annealing based Clustering, K-medoids Algorithm, Fuzzy C-Medoids Algorithm, Cluster Validity Indices, Statistical significance test.

1 Introduction

Genetic algorithms [1, 2, 3] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and have a large amount of implicit parallelism. GAs perform search in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem. The algorithm starts by initializing a population of potential solutions encoded into strings called chromosomes. Each solution has some fitness value based on which the fittest parents that would be used for reproduction are found (survival of the fittest). The new generation is created by applying genetic operators like crossover (exchange of information among parents) and mutation (sudden small change in a parent) on selected parents. Thus the quality of population is improved as the number of generations increases. The process continues until some specific criterion is met or the solution converges to some optimized value.

Simulated Annealing (SA) [4], a popular search algorithm, utilizes the principles of statistical mechanics regarding the behaviour of a large number of atom at low temperature, for finding minimal cost solutions to large optimization problems by minimizing the associated energy. In statistical mechanics, investigating the ground states or low-energy states of matter is of fundamental importance. These states are achieved at very low temperatures. However, it is not sufficient to lower the temperature alone since this results in unstable states. In the annealing process, the temperature is first raised, then decreased gradually to a very low value (T_{min}), while ensuring that one spends sufficient time at each temperature value. This process yields stable low-energy states. Geman and Geman [5] provided a proof that SA, if annealed sufficiently slow, converges to the global optimum. Being based on strong theory, SA has been applied in diverse areas by optimizing a single criterion.

Clustering [6, 7, 8, 9] is a useful unsupervised data mining technique which partitions the input space into K regions depending on some similarity/dissimilarity metric where the value of K may or may not be known a priori. The main objective of any clustering technique is to produce a $K \times n$ partition matrix $U(X)$ of the given data set X, consisting of n patterns, $X = x_1, x_2, \ldots, x_n$. The partition matrix may be represented as $U = [u_{k,j}]$, $k = 1,\ldots, K$ and $j = 1,\ldots, n$, where $u_{k,j}$ is the membership of pattern x_j to the kth cluster. For fuzzy clustering of the data, $0 < u_{k,j} < 1$, i.e., $u_{k,j}$ denotes the degree of belongingness of pattern x_j to the kth cluster. The objective of the Fuzzy C-Means algorithm [10] is to maximize the global compactness of the clusters. Fuzzy C-Means clustering algorithm cannot be applied for clustering categorical data sets, where there is no natural ordering among the elements of an attribute domain. Thus no inherent distance measures, such as Euclidean distance, can be used to compute the distance between two feature vectors [11, 12, 13]. Hence it is not feasible to compute the numerical average of a set of feature vectors. To handle such categorical data sets, well known relational clustering algorithm is PAM (Partitioning Around Medoids) due to Kaufman and Rousseeuw [14]. This algorithm is based on finding K

*Date of the manuscript submission: 13th April 2008

†Academy of Technology, Department of Information Technology. Adisaptagram-712121, West Bengal, India. *Email* : *indra_raju@yahoo.co.in*

‡University of Kalyani, Department of Computer Science and Engineering. Kalyani-741235, West Bengal, India. *Email* : *anirbanbuba@yahoo.com*

representative objects (also known as medoids [15]) from the data set in such a way that the sum of the within cluster dissimilarities is minimized. A modified version of PAM called CLARA (Clustering LARge Applications) to handle large data sets was also proposed by Kaufman and Rousseeuw [14]. Ng and Han [16] proposed another variation of CLARA called CLARANS. This algorithm tries to make the search for the representative objects (medoids) more efficient by considering candidate sets of mediods in the neighborhood of the current set of medoids. However, CLARANS is not designed for relational data. Finally, it is also interesting to note that Fu [17] suggested a technique very similar to the medoid technique in the context of clustering string patterns generated by grammars in syntactic pattern recognition. Some of the more recent algorithms for relational data clustering include [18, 19, 20, 21].

All the above algorithms, including SAHN [22] generate crisp clusters. When the clusters are not well defined (i.e., when they overlap) we may desire fuzzy clusters. As Krishnapuram describe an algorithm named Fuzzy C-Medoids (FCMdd) [23] which is effective in web document application. We have applied this algorithm to categorical dataset and the algorithm optimize a single objective function. Moreover motivated by this fact, here we have used global optimization tools like genetic algorithm and simulated annealing to optimize the FCMdd objective function (J_m). The superiority of the proposed methods over FCMdd clustering algorithm has been demonstrated on different synthetic and real life data sets.

2 Categorical Data Clustering Algorithms

This section describes some Hierarchical and Partitional clustering algorithms used for categorical data.

2.1 Complete-Linkage Clustering

The complete-linkage (CL) hierarchical clustering Algorithm is also called the maximum method or the farthest neighbor method [24]. It is obtained by defining the distance between two clusters to be the largest distance between a sample in one cluster and a sample in the other cluster. If C_i and C_j are clusters, we define

$$D_{CL}(C_i, C_j) = \max_{a \in C_i, b \in C_j} d(a, b) \tag{1}$$

2.2 Average-Linkage Clustering

The hierarchical average-linkage (AL) clustering algorithm, also known as the unweighted pair-group method using arithmetic averages (UPGMA) [24], is one of the most widely used hierarchical clustering algorithms. The average-linkage algorithm is obtained by defining the distance between two cluster to be the average distance between a pont in one cluster and a point in the other cluster. Formally, if C_i is a cluster with n_i members and C_j is a cluster with n_j members, the distance between the clusters is

$$D_{AL}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{a \in C_i, b \in C_j} d(a, b). \tag{2}$$

2.3 K-medoids Clustering

Partitioning around medoids (PAM), also called K-medoids clustering [25], is a variation of K-means with the objective to minimize the within cluster variance W(K).

$$W(K) = \sum_{i=1}^{K} \sum_{x \in C_i} D(x, m_i) \tag{3}$$

Here m_i is the medoid of cluster C_i and $D(x, m_i)$ denotes the distance between the point x and m_i. K denotes the number of clusters. The resulting clustering of the data set X is usually only a local minimum of $W(K)$. The idea of PAM is to select K representative points, or medoids, in X and assign the rest of the data points to the cluster identified by the nearest medoid. Initial set of K medoids are selected randomly. Subsequently, all the points in X are assigned to the nearest medoid. In each iteration, a new medoid is determined for each cluster by finding the data point with minimum total distance to all other points of the cluster. After that, all the points in X are reassigned to their clusters in accordance with the new set of medoids. The algorithm iterates until $W(K)$ does not change any more.

2.4 Fuzzy C-Medoids

Fuzzy C-Medoids (FCMdd) [23] is a widely used technique that uses the principles of fuzzy sets to evolve a partition matrix $U(X)$ while minimizing the measure

$$J_m = \sum_{j=1}^{n} \sum_{k=1}^{K} u_{k,j}^m D(z_k, x_j), \quad 1 \leq m \leq \infty \tag{4}$$

where n is the number of data objects, K represents number of clusters, u is the fuzzy membership matrix (partition matrix) and m denotes the fuzzy exponent. Here x_j is the jth data point and z_k is the center of kth cluster, and $D(z_k, x_j)$ denotes the distance of point x_j from the center of the kth cluster. In this article, the new norm (describe in Section 3) is taken as a measure of the distance between two points.

FCMdd algorithm starts with random initial K cluster centers, and then at every iteration it finds the fuzzy membership of each data points to every cluster using

the following equation [23]

$$u_{i,k} = \frac{(\frac{1}{D(z_i,x_k)})^{\frac{1}{m-1}}}{\sum_{j=1}^{K}(\frac{1}{D(z_j,x_k)})^{\frac{1}{m-1}}},\ for\ 1 \le i \le K,\ 1 \le k \le n \quad (5)$$

for $1 \le i \le K$; $1 \le k \le n$, where $D(z_i, x_k)$ and $D(z_j, x_k)$ are the distances between x_k and z_i, and x_k and z_j respectively. m is the weighting coefficient. (Note that while computing $u_{i,k}$ using Eqn. 5, if $D(z_j, x_k)$ is equal to zero for some j, then $u_{i,k}$ is set to zero for all $i = 1, \ldots, K$, $i \neq j$, while $u_{i,k}$ is set equal to one.) Based on the membership values, the cluster centers are recomputed using the following equation

$$q_i = argmin_{1 \le j \le n} \sum_{k=1}^{n} u_{i,k}^m D(x_j, x_k), 1 \le i \le K \quad (6)$$

and

$$z_i = q_i, 1 \le i \le K \quad (7)$$

The algorithm terminates when there is no further change in the cluster centers. Finally, each data point is assigned to the cluster to which it has maximum membership.

3 Distance Metric

As discussed earlier, absence of any natural ordering among the elements of a categorical attribute domain prevents us to apply any inherent distance measure like Euclidean distance, to compute the distance between two categorical objects [26]. In this article following distance measure has been adopted for all the algorithms considered. Let $x_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]$, and $x_j = [x_{j1}, x_{j2}, \ldots, x_{jp}]$ be two categorical objects described by p categorical attributes. The distance measure between x_i and x_j, $D(x_i, x_j)$, can be defined by the total number of mismatches of the corresponding attribute categories of the two objects. Formally,

$$D(x_i, x_j) = \sum_{k=1}^{p} \delta(x_{ik}, x_{jk}) \quad (8)$$

where

$$\delta(x_{ik}, x_{jk}) = \begin{cases} 0 & \text{if } x_{ik} = x_{jk} \\ 1 & \text{if } x_{ik} \neq x_{jk} \end{cases} \quad (9)$$

Note that $D(x_i, x_j)$ gives equal importance to all the categories of an attribute. However, in most of the categorical data sets, the distance between two data vectors depends on the nature of the data sets. Thus, if a distance matrix is precomputed for a given data set, the algorithms can adopt this for computing the distances.

4 Genetic Algorithm based Clustering: GAC

4.1 Basic Principle

The searching capability of GAs has been used in this article for the purpose of appropriately determining a fixed number K of cluster centers in $\Re^n$; thereby suitably clustering the set of n unlabelled points. The clustering metric that has been adopted is the sum of the distances of the points from their respective cluster centers. Mathematically, the clustering metric ζ for the K clusters $C_1, C_2, ..., C_K$ is given by

$$\zeta(C_1, C_2, \ldots, C_K) = \sum_{i=1}^{K} \sum_{x_j \in C_i} D(x_j, z_i), \quad (10)$$

where D is the distance metric. The task of the GA is to search for the appropriate cluster centers $z_1, z_2, \ldots, z_K$ such that the clustering metric ζ is minimized. The basic steps of GAs, which are also followed in the GA-clustering (GAC) algorithm. These are now described in detail.

4.2 Chromosome representation

Each chromosome has K genes and each gene of the chromosome has an allele value chosen randomly from the set $\{1, 2, \ldots, n\}$, where K is the number of clusters and n is the number of points. Hence a chromosome is represented as a vector of indices of the points in the data set. Each point index in a chromosome implies that the corresponding point is a cluster medoid.

Example 1. Let K = 6, i.e., Then the chromosome

51 72 18 15 29 32

represents the indices of six points qualified for cluster medoids. A chromosome is valid if no point index occurs more than once in the chromosome.

4.3 Population initialization

The K cluster medoids encoded in each chromosome are initialized to K randomly chosen points from the data set. This process is repeated for each of the P chromosomes in the population, where P is the size of the population.

4.4 Fitness computation

The fitness computation process consists of two phases. In the first phase, the clusters are formed according to the centers encoded in the chromosome under consideration. This is done by assigning each point $x_i, i = 1, 2, \ldots, n$ to one of the clusters C_j with center z_j such that

$$D(x_i, z_j) < D(x_i, z_p),\ p = 1, 2, \ldots, K,\ and\ p \neq j, \quad (11)$$

where D is the distance metric. All ties are resolved arbitrarily. After the clustering is done, the cluster medoids encoded in the chromosome are replaced by the points having minimum total distance to the points of the respective clusters. In other words, for cluster C_i, the new medoid is point x_t where,

$$x_t = argmin_{x_j \in C_i} \sum_{x_k \in C_i} D(x_j, x_k). \quad (12)$$

Hence the ith gene in the chromosome is replaced by t. Subsequently, the clustering metric ζ computed as follows:

$$\zeta = \sum_{i=1}^{K} \zeta_i, \quad (13)$$

where,

$$\zeta_i = \sum_{x_j \in C_i} D(x_j, z_i). \quad (14)$$

The fitness function is defined as $f = \zeta$, so that minimization of the fitness function leads to minimization of ζ, indicating highly compact clusters.

4.5 Selection

The selection process selects chromosomes from the mating pool directed by the survival of the fittest concept of natural genetic systems. In the proportional selection strategy adopted in this article, a chromosome is assigned a number of copies, which is proportional to its fitness in the population, that go into the mating pool for further genetic operations. Tournament selection is one common technique that implements the proportional selection strategy.

4.6 Crossover

Crossover is a probabilistic process that exchanges information between two parent chromosomes for generating two child chromosomes. In this article single point crossover with a fixed crossover probability of μ_c is used. For chromosomes of length l, a random integer, called the crossover point, is generated in the range [1, l-1]. The portions of the chromosomes lying to the right of the crossover point are exchanged to produce two offspring chromosomes.

4.7 Mutation

Each chromosome undergoes mutation with a fixed probability μ_m. The mutation operation has been defined as following: From the string to be mutated, a random element is chosen and it is replaced by a different index of point in the range $\{1, \ldots, n\}$ such that no element is duplicated in the string.

4.8 Termination criterion

In this article the processes of fitness computation, selection, crossover, and mutation are executed for a fixed number of iterations. The best string seen upto the last generation provides the solution to the clustering problem. We have implemented elitism at each generation by preserving the best string seen upto that generation in a location outside the population. Thus on termination, this location contains the centers of the final clusters. Fig. 1 shows the steps of GAC algorithm.

```
1) Encoding
2) Initial population creation.
 Generation=100
 while( i<Generation)
   3) Fitness value calculation.
   4) Selection.
   5) Crossover with probability = 0.8.
   6) Mutation with probability = 0.1.
 i=i+1;
 End while
```

Figure 1: GAC Algorithm

5 Simulated Annealing based Clustering: SAC

Simulated annealing (SA) [4] is an optimization tool which has successful applications in a wide range of combinatorial optimization problems. This fact has motivated researchers to use SA in simulation optimization. However SA still needs to evaluate the objective function values accurately, and there have been few theoretical studies for the SA algorithm when the objective function is estimated through simulation. There are some applications of SA in clustering [27, 28]. In this article, we have used SA for designing a categorical data clustering method. The algorithm is named as simulated annealing clustering (SAC). This algorithm is described below.

5.1 String representation

In this article, a configuration (string) is represented in similar way a chromosome is represented in GAC, i.e., the string has length K and Each element of the string is chosen randomly from the set $\{1, 2, \ldots, n\}$, where K is the number of clusters and n is the number of points. Hence a string is represented as a vector of indices of the points in the data set. Each point index in a string indicates that the corresponding point is a cluster medoid. A string is valid if no point index occurs more than once in it.

5.2 Fitness computation

The fitness of a string is computed similarly as in GAC, i.e., first the encoded medoids are used for cluster assignments and the string is updated using new medoids. Thereafter the fitness (K-medoid error function) is computed as per Eqn. 10.

5.3 Perturbation

The current string undergoes perturbation as follows: the position of perturbation is chosen randomly and the value of that position is replaced by some other value chosen randomly from the set $\{1, 2, \ldots, n\}$. This way, perturbation of a string yields a new string. The steps of SAC algorithm is shown Fig. 2.

```
1) q = Random initial string.
 T = T_{max}.
  2) E(q,T) = Fitness of q.
  while( T >= T_{min} )
    for i = 1 to k
     3) s = Perturb ( q ).
     4) E(s,T) = Fitness of s.
       if (E(s,T) - E(q,T) < 0 )
        5) Set q = s and E(q,T) = E(s,T) .
      else
        6) Set q = s and E(q,T) = E(s,T) with
         probability exp-( E(s,T) - E(q,T) )/T
     End for
     T= T*r. /* 0 < r < 1 */
   End while
```

Figure 2: SAC Algorithm

6 The Proposed Genetic Algorithm based Fuzzy Clustering: GAFC

In this section, we describe the use of GA for evolving a fuzzy partition matrices. FCMdd [23] measure are considered as the objective functions that must be minimized. The technique of Chromosome representation (Section 4.2), Population initialization (Section 4.3), Selection (Section 4.5), Crossover (Section 4.6), Mutation (Section 4.7), Termination criterion (Section 4.8) are same only the Fitness computation described below in detail.

6.1 Fitness computation

In this article the FCMdd measure are taken as the objective that need to be optimized. For computing the measures, the centers encoded in a chromosome are first extracted. Let these be denoted as $v_1, v_2, \ldots, v_K$. The membership values $u_{i,k}$, $i = 1, 2, \ldots, K$ and $k = 1, 2, \ldots, n$ are computed as follows

$$u_{i,k} = \frac{(\frac{1}{D(v_i,x_k)})^{\frac{1}{m-1}}}{\sum_{j=1}^{K}(\frac{1}{D(v_j,x_k)})^{\frac{1}{m-1}}},\ for\ 1 \le i \le K,\ 1 \le k \le n \tag{15}$$

where $D(v_i, x_k)$ and $D(v_j, x_k)$ are as described earlier. m is the weighting coefficient. (Note that while computing $u_{i,k}$ using Eqn. 15, if $D(v_j, x_k)$ is equal to zero for some j, then $u_{i,k}$ is set to zero for all $i = 1, ..., K$, $i \neq j$, while $u_{j,k}$ is set equal to one.) Subsequently, the centers encoded in a chromosome are updated using the following equations

$$q_i = argmin_{1 \le j \le n} \sum_{k=1}^{n} u_{i,k}^m D(x_j, x_k),\ 1 \le i \le K \tag{16}$$

and

$$v_i = q_i, 1 \le i \le K \tag{17}$$

and the cluster membership values are recomputed. The FCMdd measure J_m is defined as follows:

$$J_m = \sum_{j=1}^{n}\sum_{k=1}^{K} u_{k,j}^m D(z_k, x_j),\ 1 \le m \le \infty \tag{18}$$

where m is the fuzzy exponent.

7 The Proposed Simulated Annealing based Fuzzy Clustering: SAFC

Here we are described the technique of Fitness Computation of SAFC algorithm, rest of the part of this algorithm is same with the Section 5.1, Section 5.3.

7.1 Fitness computation

The fitness of a string is computed similarly as in GAFC, i.e., first the encoded medoids are used for membership value calculation and the string is updated using new medoids. Thereafter the fitness (FCMdd error function) is computed as per Eqn. 18.

8 Experimental Results

The performance of the proposed algorithms has been evaluated on synthetic data sets (Cat01 and Cat02) and real life data sets (Soybean, Zoo, Votes, Balance-Scale, Tic-tac-toe and Car). The proposed clustering schemes have been compared with different algorithms, viz., Complete-linkage, Average-linkage, K-medoids and FCMdd. Each algorithm has been run for 20 times. The average of different indices (described later) has been reported.

8.1 Synthetic Categorical Data Sets

Cat01: The *'Cat01'* is a synthetic data set which consists of 20 instances with 5 features. The data set has 2

clusters.

Cat02: The *'Cat02'* data is also a synthetic data set which consists of 132 instances with 5 features. This data set has 3 clusters. The synthetic data sets are generated using a web based data generation tool[1].

8.2 Real Life Categorical Data Sets

Soybean: The Soybean data set contains 47 data points on diseases in soybeans. Each data point has 35 categorical attributes and is classified as one of the four diseases, i.e., number of clusters in the data set is 4.

Zoo: The Zoo data consists of 101 instances of animals in a zoo with 17 features. The name of the animal constitutes the first attribute. This attribute is neglected. There are 15 boolean attributes corresponding to the presence of hair, feathers, eggs, milk, backbone, fins, tail; and whether airborne, aquatic, predator, toothed, breathes, venomous, domestic and catsize. The character attribute corresponds to the number of legs lying in the set 0, 2, 4, 5, 6, 8. The data set consists of 7 different classes of animals.

Congressional Votes: This data set is the United States Congressional voting records in 1984. Total number of records is 435. Each row corresponds to one Congress mans votes on 16 different issues (e.g., education spending, crime etc.). All attributes are boolean with Yes (that is, 1) and No (that is, 0) values. A classification label of Republican or Democrat is provided with each data record. The data set contains records for 168 Republicans and 267 Democrats.

Balance-Scale: This is a weight and distance Database. The Balance-Scale data set contains 625 data points. Each data point has 4 categorical attributes. Number of clusters in the data set is 3. The Information about the attribute of this data set are Left-Weight, Left-Distance, Right-Weight and Right-Distance. Attributes are given in numerical from such as 1 to 5.

Tic-tac-toe: The Tic-tac-toe data consists of 958 instances of legal tic-tac-toe endgame boards with 10 features where each corresponding to one tic-tac-toe square. Out of this 10 features last one is a class identifier. Others are corresponding to the top-left-square, top-middle-square, top-right-square, middle-left-square, middle-middle-square, middle-right-square, bottom-left-square, bottom-middle-square and bottom-right-square. Those squares are identified by x = player x has taken or o = player o has taken or b = blank.

Car: The Car data consists of 1728 instances. All instances completely cover the attribute space. Out of this 7 features last one is a class identifier. Others are corresponding to the Buying (vhigh, high, med, low.), Maint (vhigh, high, med, low.), Doors (2, 3, 4, 5 more.), Persons (2, 4, more.), Lug_boot (small, med, big.), Safety (low, med, high.). Class identifier has four distinct members, those are unacc, acc, good and vgood.

The real life data sets mentioned above were obtained from the UCI Machine Learning Repository[2].

8.3 Input Parameters

The GAC and GAFC algorithms is executed for 100 generations with population size=20. The crossover and mutation probabilities are taken to be 0.8 and 0.1, respectively. The parameters of the SAC and SAFC algorithm are as follows: T_{max}=100, T_{min}=0.01, r=0.9 and k=100. The FCMdd and K-medoids algorithms are run for 100 iterations unless they converge before that.

8.4 Performance Metric

For evaluating the performance of the clustering algorithms, Minkowski score [29], $\mathcal{I}$ index [30], Silhouette index [31] and Adjusted Rand Index [32] are used for both artificial and real life categorical data sets, respectively.

8.4.1 Minkowski score

Minkowski score (MS) is define as follows : A clustering solution for a set of n elements can be represented by an $n \times n$ matrix C, where $C_{i,j} = 1$ if point i and j are in the same cluster according to the solution, and $C_{i,j} = 0$ otherwise. The Minkowski score of a clustering result C with reference to T , the matrix corresponding to the true clustering, is defined as

$$MS(T, C) = \frac{\| T - C \|}{\| T \|} \tag{19}$$

where

$$\| T \| = \sqrt{\sum_i \sum_j T_{i,j}}$$

The Minkowski score is the normalized distance between the two matrices. Lower Minkowski score implies better clustering solution, and a perfect solution will have a score zero.

8.4.2 $\mathcal{I}$ index

A cluster validity index $\mathcal{I}$ [30], proposed recently as a measure of indicating the goodness/validity of cluster solution, is defined as follows:

$$I(K) = (\frac{1}{K} \times \frac{E_1}{E_K} \times D_K)^p, \tag{20}$$

where K is the number of clusters. Here,

$$E_K = \sum_{i=1}^{K} \sum_{k=1}^{n} u_{i,k} \parallel z_i - x_k \parallel, \tag{21}$$

[1] http://www.datgen.com

[2] http://www.ics.uci.edu/∼ mlearn/MLRepository.html

Table 1: Average values of Indices for Crisp Algorithms.

DataSets	Mesure Indices	CL	AL	K-medoids	GAC	SAC
Cat01	MS	0.9953	0.9492	0.6617	0.0302	**0.0000**
	$\mathcal{I}$	1.5302	3.2201	4.2125	5.0263	**9.6319**
	s(C)	0.4632	0.5010	0.5519	0.8561	**1.0000**
	ARI	0.5563	0.6151	0.7751	0.8708	**1.0000**
Cat02	MS	1.3520	1.1384	0.9425	**0.4522**	0.4922
	$\mathcal{I}$	1.0042	2.2341	3.1682	**6.9508**	5.0234
	s(C)	0.4231	0.5003	0.5983	**0.7781**	0.7205
	ARI	0.4905	0.5518	0.7906	**0.9183**	0.8901
Soybean	MS	1.3954	1.2432	0.7851	**0.2522**	0.2982
	$\mathcal{I}$	4.6322	7.3256	10.6329	**15.9326**	13.0659
	s(C)	0.4829	0.5621	0.6212	**0.7851**	0.7535
	ARI	0.5162	0.5966	0.7174	**0.9063**	0.8832
Zoo	MS	1.1642	1.1262	0.6920	0.5332	**0.4840**
	$\mathcal{I}$	11.6310	16.7255	21.0632	25.5316	**27.5981**
	s(C)	0.4261	0.5516	0.6882	0.7219	**0.7433**
	ARI	0.5852	0.6355	0.6827	0.7441	**0.7939**
Votes	MS	1.1192	0.9747	0.7401	**0.6452**	0.6618
	$\mathcal{I}$	5.3216	7.5353	9.6363	**17.6291**	15.5271
	s(C)	0.4322	0.4891	0.6182	**0.7431**	0.7102
	ARI	0.5150	0.5726	0.7278	**0.8556**	0.8233
Balance-Scale	MS	1.2024	1.0871	0.7113	0.4315	**0.4207**
	$\mathcal{I}$	7.3241	9.0935	12.0631	16.5032	**18.0233**
	s(C)	0.4853	0.5233	0.6635	0.6855	**0.7361**
	ARI	0.6033	0.6462	0.7582	0.8287	**0.8592**
Tic-tac-toe	MS	1.4503	1.0520	0.6372	**0.4512**	0.5001
	$\mathcal{I}$	9.0302	12.4631	19.0638	**26.4242**	23.4762
	s(C)	0.4853	0.5263	0.5728	**0.6172**	0.6053
	ARI	0.5162	0.5321	0.6457	**0.7601**	0.7472
Car	MS	1.3215	1.1127	0.7218	0.4686	**0.4635**
	$\mathcal{I}$	15.5326	20.0673	26.5373	28.6372	**31.9251**
	s(C)	0.3882	0.4671	0.5051	0.7236	**0.7751**
	ARI	0.5781	0.6363	0.7071	0.8242	**0.8441**

and

$$D_K = max_{i \neq j} \| z_i - z_j \|, \tag{22}$$

The power p is used to control the contrast between the different cluster configurations. In this article, we have taken $p = 2$.

8.4.3 Silhouette index

Silhouette index [31] is a cluster validity index that is used to judge the quality of any clustering solution C. Suppose a represents the average distance of a point from the other points of the cluster to which the point is assigned, and b represents the minimum of the average distances of the point from the points of the other clusters. Now the silhouette width s of the point is defined as:

$$s = \frac{b - a}{max\{a, b\}} \tag{23}$$

Silhouette index $s(C)$ is the average Silhouette width of all the data points and it reflects the compactness and separation of clusters. The value of Silhouette index varies from -1 to 1 and higher value indicates better clustering result.

8.4.4 Adjusted Rand Index

Suppose T is the true clustering of a data set based on domain knowledge and C a clustering result given by some clustering algorithm. Let a, b, c and d respectively denote the number of pairs belonging to the same cluster in both T and C, the number of pairs belonging to the same cluster in T but to different clusters in C, the number of pairs belonging to different clusters in T but to the same cluster in C and the number of pairs belonging to different clusters in both T and C. The adjusted Rand index

Table 2: Average values of Indices for Fuzzy Algorithms.

DataSets	Mesure Indices	CL	AL	FCMdd	GAFC	SAFC
Cat01	MS	0.9953	0.9492	0.2393	0.0041	**0.0032**
	$\mathcal{I}$	1.5302	3.2201	4.3277	5.0765	**8.4341**
	s(C)	0.4632	0.5010	0.6212	0.8769	**0.9061**
	ARI	0.5563	0.6151	0.8761	0.9091	**0.9362**
Cat02	MS	1.2520	1.1384	0.7205	**0.4161**	0.4284
	$\mathcal{I}$	1.0042	2.2341	3.2012	**7.8184**	6.0232
	s(C)	0.4231	0.5003	0.6631	**0.8513**	0.7805
	ARI	0.4905	0.5518	0.8607	**0.9447**	0.9225
Soybean	MS	1.3954	1.2432	0.7821	0.1203	**0.1130**
	$\mathcal{I}$	4.6322	7.3256	12.3246	16.6602	**19.6051**
	s(C)	0.4829	0.5621	0.6322	0.8321	**0.8605**
	ARI	0.5162	0.5966	0.7345	0.9237	**0.9522**
Zoo	MS	1.1642	1.1262	0.5549	**0.4312**	0.4762
	$\mathcal{I}$	11.6310	16.7255	24.5632	**32.4646**	28.4921
	s(C)	0.4261	0.5516	0.7351	**0.7808**	0.7533
	ARI	0.5852	0.6355	0.7562	**0.8708**	0.8066
Votes	MS	1.1192	0.9747	0.7329	0.6344	**0.6261**
	$\mathcal{I}$	5.3216	7.5353	10.3219	18.5030	**20.6304**
	s(C)	0.4322	0.4891	0.6388	0.7632	**0.7982**
	ARI	0.5150	0.5726	0.7678	0.8653	**0.8861**
Balance-Scale	MS	1.2024	1.0871	0.6809	**0.3872**	0.4125
	$\mathcal{I}$	7.3241	9.0935	14.9131	**21.0391**	19.0908
	s(C)	0.4853	0.5233	0.6952	**0.7682**	0.7452
	ARI	0.6033	0.6462	0.7805	**0.9222**	0.8703
Tic-tac-toe	MS	1.4503	1.0520	0.5353	**0.4273**	0.4502
	$\mathcal{I}$	9.0302	12.4631	21.3092	**28.4504**	26.0359
	s(C)	0.4853	0.5263	0.6032	**0.6645**	0.6346
	ARI	0.5162	0.5321	0.6955	**0.8073**	0.7651
Car	MS	1.3215	1.1127	0.7106	**0.4302**	0.4581
	$\mathcal{I}$	15.5326	20.0673	26.4343	**35.5347**	32.6233
	s(C)	0.3882	0.4671	0.6235	**0.8053**	0.7869
	ARI	0.5781	0.6363	0.7526	**0.8815**	0.8545

$ARI(T,\ C)$ is then defined as follows:

$$ARI(T,\ C) = \frac{2(ad - bc)}{(a+b)(b+d) + (a+c)(c+d)} \qquad (24)$$

The value of $ARI(T,\ C)$ lies between 0 and 1 and higher value indicates that C is more similar to T. Also, $ARI(T,\ T) = 1$.

8.5 Results

Each algorithm is executed for 20 times. Average values obtained for different indices in 20 consecutive runs of the algorithms are reported for different data sets in Tables 1 and 2. It is evident from the tables, as well as from the Figs. 3 and Fig. 4 that both the GAC, SAC in crisp domain and GAFC, SAFC in fuzzy domain consistently outperform the hierarchical clustering, K-medoids and FCMdd algorithms. The performances of GAC, SAC and GAFC, SAFC are comparable to each other.
It can be noticed from the Table 1 (for crisp domain) that for the Zoo data set, SAC algorithm provides best result for all the performance indices. The algorithm produces the average values of 0.4840, 27.5981, 0.7433 and 0.7939 for MS, $\mathcal{I}$, $s(C)$ and ARI, respectively. It is also evident from Table 1 that for Tic-tac-toe data set, GAC algorithm has outperformed other algorithms. Similarly, for example, GAFC algorithm of Table 2 (for fuzzy domain) provides the different average indices values of 0.4302, 35.5347, 0.8053 and 0.8815, respectively for the Car data set. It shows that the GAFC algorithm performs better than the other algorithms for this data set. Also for the data set like Votes, the SAFC algorithm works better compared to the other algorithms. Overall it is found that GAC and SAC in crisp domain, and GAFC and SAFC in the fuzzy domain consistently outperform the other algorithms for all the data sets. Another interesting observation is that GAFC and SAFC algorithms outperform GAC and SAC for all the data sets. This indicates the utility of incorporating fuzziness in the algorithms.

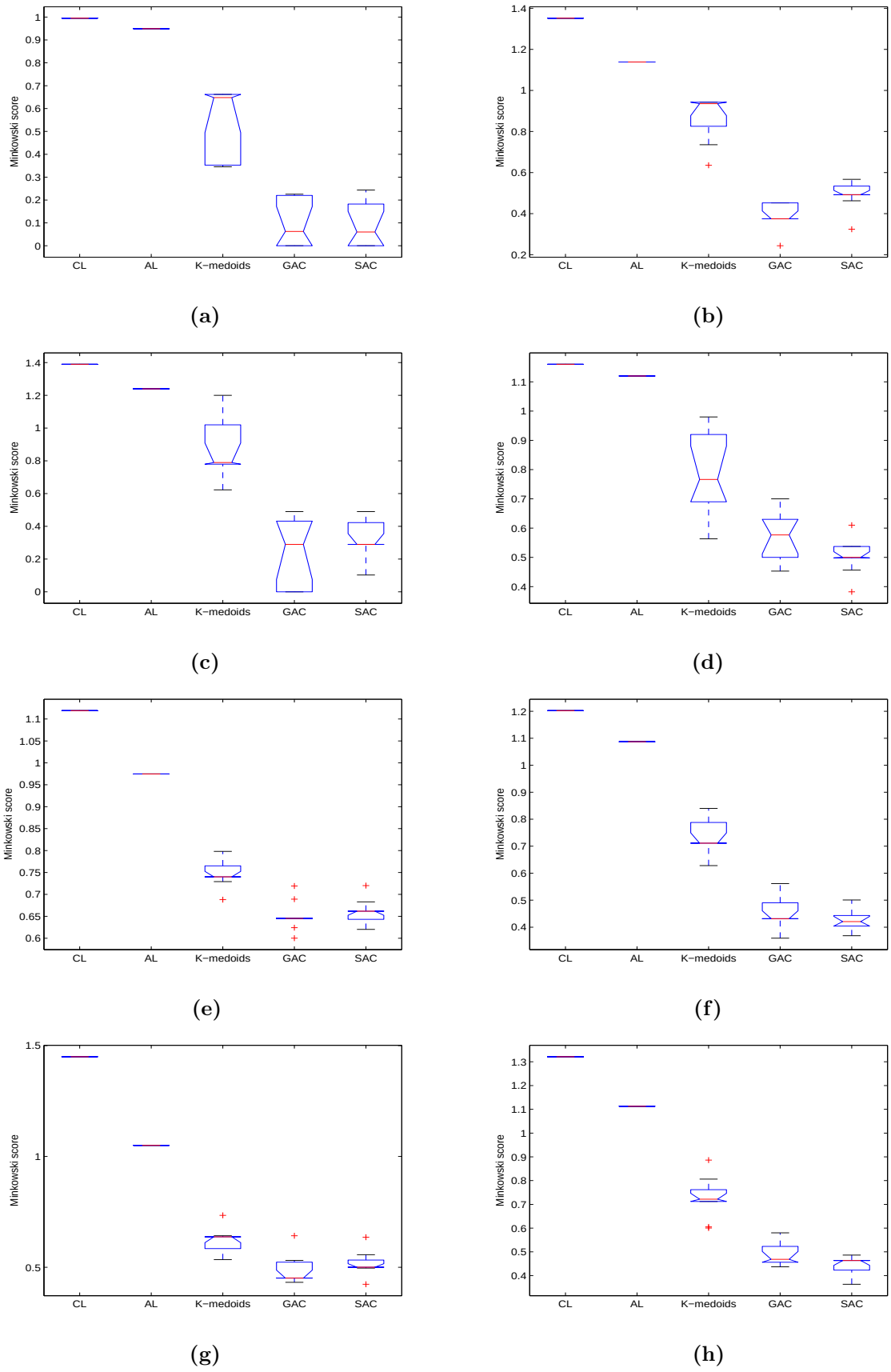

Figure 3: Boxplots of Minkowski scores for different crisp algorithms on (a) Cat01 (b) Cat02 (c) Soybean (d) Zoo (e) Votes (f) Balance-Scale (g) Tic-tac-toe (h) Car .

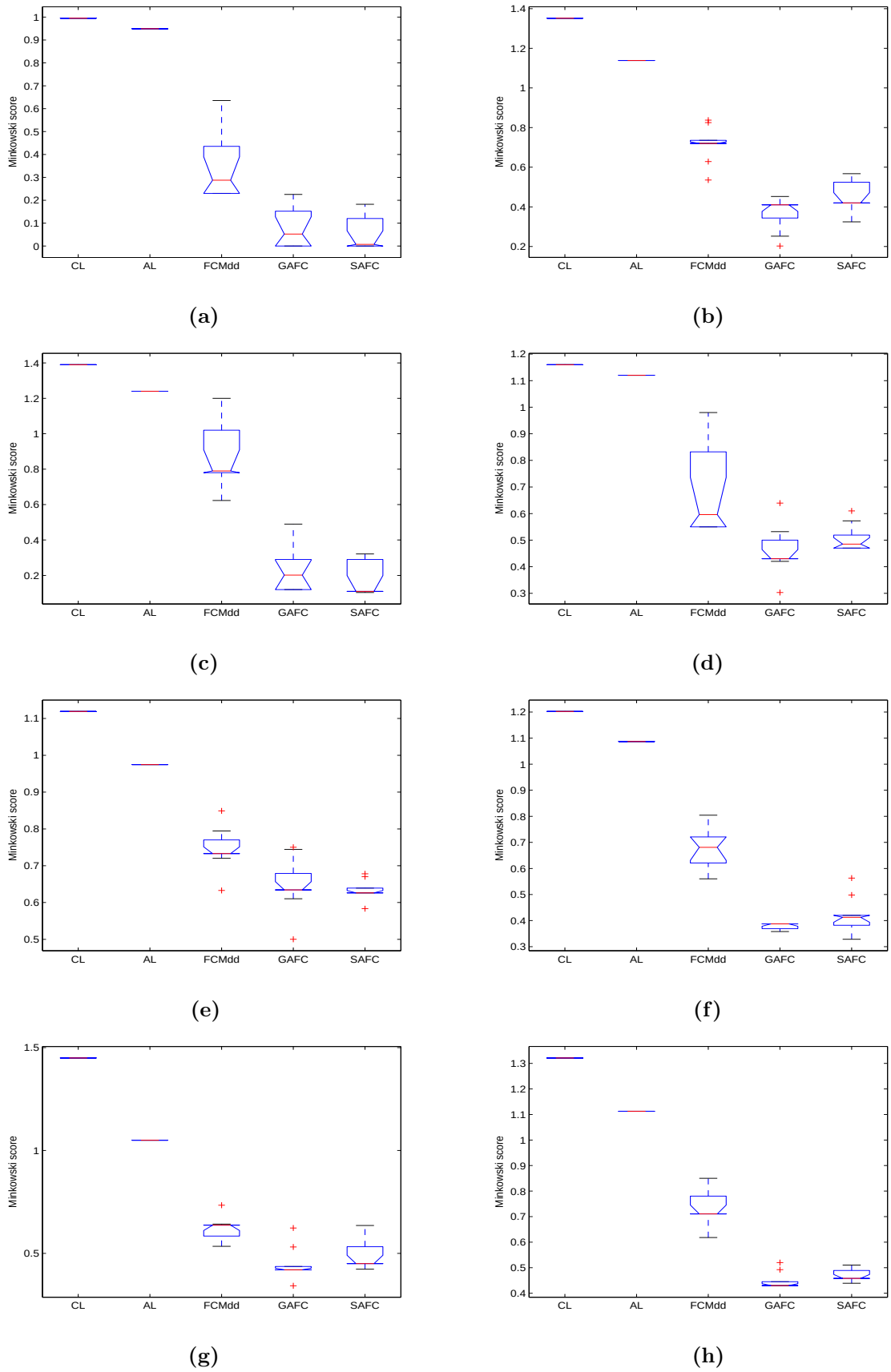

Figure 4: Boxplots of Minkowski scores for different fuzzy algorithms on (a) Cat01 (b) Cat02 (c) Soybean (d) Zoo (e) Votes (f) Balance-Scale (g) Tic-tac-toe (h) Car .

Table 3: Median values of the Minkowski Score for the Datasets over 50 consecutive runs of different crisp Algorithms.

Algorithm	Cat01	Cat02	Soybean	Zoo	Votes	Balance-Scale	Tic-tac-toe	Car
CL	0.9953	1.3520	1.3900	1.1600	1.1301	1.1842	1.4500	1.3209
AL	0.9492	1.1384	1.2400	1.1200	0.9481	1.1246	1.0500	1.1681
K-medoids	0.6480	0.9354	0.7900	0.7662	0.7102	0.7321	0.6372	0.7431
GAC	0.0625	0.4458	0.2900	0.5766	0.6483	0.4219	0.4512	0.4218
SAC	0.0602	0.4922	0.2900	0.5000	0.6742	0.4325	0.5001	0.4492

Table 4: Median values of the Minkowski Score for the Datasets over 50 consecutive runs of different fuzzy Algorithms.

Algorithm	Cat01	Cat02	Soybean	Zoo	Votes	Balance-Scale	Tic-tac-toe	Car
CL	0.9953	1.3520	1.3900	1.1600	1.1192	1.3643	1.4500	1.3215
AL	0.9492	1.1384	1.2400	1.1200	0.9747	1.0871	1.0500	1.1193
FCMdd	0.2577	0.7200	0.7900	0.5261	0.7329	0.6654	0.6372	0.7106
GAFC	0.0525	0.4100	0.2016	0.4300	0.6344	0.3872	0.4200	0.4428
SAFC	0.0073	0.4200	0.1100	0.4850	0.6261	0.4125	0.4500	0.4581

Table 5: $P-values$ produced by Wilcoxon's Rank Sum test comparing GAC with other Algorithms.

DataSet	*P-value*			
	CL	AL	K-medoids	SAC
Cat01	5.5111e-005	5.4332e-005	1.4118e-004	0.8717
Cat02	5.3477e-005	5.3067e-005	1.4462e-004	0.4182
Soybean	5.9802e-005	5.9002e-005	1.6211e-004	0.3459
Zoo	5.7633e-005	5.7033e-005	1.6091e-003	0.1062
Votes	5.2983e-005	5.0743e-005	1.7082e-004	0.2873
Balance-Scale	5.7427e-005	5.7104e-005	1.6871e-004	0.5291
Tic-tac-toe	5.5111e-005	5.5111e-005	1.6091e-003	0.1669
Car	5.8641e-005	5.5801e-005	1.6094e-004	0.6603

Table 6: $P-values$ produced by Wilcoxon's Rank Sum test comparing SAC with other Algorithms.

DataSet	*P-value*			
	CL	AL	K-medoids	GAC
Cat01	5.5111e-005	5.4211e-005	1.4418e-004	0.8717
Cat02	5.5121e-005	5.3911e-005	1.4504e-004	0.4182
Soybean	5.4699e-005	5.4099e-005	1.5116e-004	0.3459
Zoo	6.1582e-005	6.1082e-005	3.1472e-004	0.1062
Votes	5.5111e-005	5.6281e-005	1.3218e-004	0.2873
Balance-Scale	5.4517e-005	5.4352e-005	1.6529e-004	0.5291
Tic-tac-toe	5.5111e-005	5.4811e-005	6.3633e-004	0.1669
Car	5.5874e-005	5.5111e-005	1.4418e-004	0.6603

Table 7: $P-values$ produced by Wilcoxon's Rank Sum test comparing GAFC with other Algorithms.

DataSet	*P-value*			
	CL	AL	FCMdd	SAFC
Cat01	5.5111e-005	5.5111e-005	1.4504e-004	0.6280
Cat02	5.4477e-005	5.2067e-005	1.4762e-004	0.1562
Soybean	5.4699e-005	5.4699e-005	1.5116e-004	0.9844
Zoo	5.1633e-005	4.7033e-005	3.6128e-003	0.5008
Votes	5.4055e-005	5.7761e-005	1.7609e-004	0.7663
Balance-Scale	5.6544e-005	5.3221e-005	1.3371e-004	0.6354
Tic-tac-toe	5.5111e-005	4.6111e-005	3.5814e-004	0.1567
Car	5.9142e-005	5.4699e-005	1.7752e-004	0.4552

Table 8: $P-values$ produced by Wilcoxon's Rank Sum test comparing SAFC with other Algorithms.

DataSet	*P-value*			
	CL	AL	FCMdd	GAFC
Cat01	5.5111e-005	5.4711e-005	1.4504e-004	0.6280
Cat02	5.5221e-005	5.0311e-005	2.6645e-004	0.1562
Soybean	5.5019e-005	5.2079e-005	1.5205e-004	0.9844
Zoo	5.3609e-005	4.7092e-005	2.446e-003	0.5008
Votes	5.3221e-005	5.7705e-005	1.7532e-004	0.7663
Balance-Scale	5.4463e-005	5.6542e-005	1.6554e-004	0.6354
Tic-tac-toe	5.5111e-005	5.1111e-005	6.3633e-004	0.1567
Car	5.7302e-005	5.4081e-005	1.4432e-004	0.4552

8.6 Statistical Significance

A non-parametric statistical significance test called Wilcoxons rank sum test for independent samples [33] has been conducted at the 5% significance level. Five groups each from crisp and fuzzy domain corresponding to the five algorithms, have been created for each data set. Each group consists of the Minkowski scores (MS) for the data sets produced by 50 consecutive runs of the corresponding algorithm. The median values of each group for all the data sets are shown in Table 3 and Table 4.
As a null hypothesis (H_0), it is assumed that there are no significant differences among the median Minkowski scores produced by all the algorithms.

$$H_0 : \forall i, j \ : \ i \neq j \Longrightarrow \mu_i = \mu_j \tag{25}$$

The alternative hypothesis (H_1) is that there are significant differences in median Minkowski scores for at least two methods.

$$H_1 : \exists i, j : i \neq j \Longrightarrow \mu_i \neq \mu_j \tag{26}$$

where μ_i denotes the median Minkowski score of the ith group.
It is evident from Table 3 and Table 4 that the median MS values for GAC, SAC in crisp domain and GAFC, SAFC in fuzzy domain are better than that for other algorithms. To establish that this goodness is statistically significant, Tables 5-8 reports the *P-values* produced by Wilcoxons rank sum test for comparison of two groups (group corresponding to one of GAC, SAC, GAFC and SAFC and a group corresponding to some other algorithm) at a time. All the *P-values* reported in the table are less than 0.05 (5% significance level). For example, the rank sum test between the algorithms GAC or SAC and K-Medoid for Zoo provides P-values of 0.0016 and 3.1472e-004, respectively, which are very small and it is also the case for fuzzy domain. This is strong evidence against the null hypothesis, indicating that the better median values of the performance metrics produced by GAC, SAC in crisp domain and GAFC, SAFC in fuzzy domain are statistically significant and have not occurred by chance. It is also evident from Tables 5-8 that GAC, SAC in crisp domain and GAFC, SAFC in fuzzy domain are both accepting the null hypothesis when they are compared with each other (as P-values are greater than 0.05). This indicates GAC and SAC as well as GAFC and SAFC provide comparable solutions. Similar results are obtained for all other data sets and for all other algorithms compared to GAC, SAC in crisp domain and GAFC, SAFC in fuzzy domain, establishing the significant superiority of the proposed technique.

9 Conclusions

In this article, genetic algorithm and simulated annealing based clustering algorithms for categorical data have been proposed in crisp and fuzzy domain. The proposed algorithms effectively optimize the K-medoids and FCMdd error function globally. The performance of the proposed algorithms have been demonstrated for different synthetic and real life data sets and also compared with that of other well-known clustering algorithms used for categorical data clustering. The results indicate that the proposed genetic algorithm and simulated annealing based algorithms can be efficiently used for clustering different categorical data sets.

References

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

[2] L. Davis, ed., *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[3] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. Reading: Addison-Wesley, 1974.

[4] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 22, pp. 671–680, 1983.

[5] S. Geman and D. Geman, "Stochastic relaxation, Gribbs distributions and the Bayesian restoration of image.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

[6] A. K. Jain and R. C. Dubes, "Data clustering: A review.," *ACM Computing Surveys*, vol. 31, 1999.

[7] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1650–1654, 2002.

[8] S. Bandyopadhyay, U. Maulik, and A. Mukhopadhyay, "Multiobjective genetic clustering for pixel classification in remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1506–1511, 2007.

[9] U. Maulik and S. Bandyopadhyay, "Genetic algorithm based clustering technique," *Pattern Recognition*, vol. 33, pp. 1455–1465, 2000.

[10] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[11] Z. Huang and M. K. Ng, "A fuzzy k-modes algorithm for clustering categorical data," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 4, 1999.

[12] P. Zhang, X. Wang, and P. X. Song, "Clustering categorical data based on distance vectors," *The Journal of the American Statistical Association*, vol. 101, no. 473, pp. 355–367, 2006.

[13] V. Ganti, J. Gehrke, and R. Ramakrishnan, "CACTUS-clustering categorical data using summeries," in *Proc. ACM SIGKDD*, 1999.

[14] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data, An Itroduction to Cluster Analysis*. Brussels, Belgium: JohnWiley and Sons, 1990.

[15] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids," *in Statistical Data Analysis Based on the L1 Norm*, pp. 405–416, 1987.

[16] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," *in Proceedings of the 20th VLDB Conference*, pp. 144–155, 1994.

[17] K. S. Fu, *Syntactic Pattern Recognition and Applications*. San Diego, CA: Academic Press, 1982.

[18] K. C. Gowda and E. Diday, "Symbolic clustering using a new similarity measure," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 368–377, 1992.

[19] G. D. Ramkumar and A. Swami, "Clustering data without distance functions," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 21, pp. 9–14, 1998.

[20] Y. E. Sonbaty and M. A. Ismail, "Fuzzy clustering for symbolic data," *IEEE Transactions on Fuzzy Systems*, vol. 6, pp. 195–204, 1998.

[21] P. Bajcsy and N. Ahuja, "Location and density-based hierarchical clustering using similarity analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1011–1015, 1998.

[22] P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy The Principles and Practice of Numerical Classification*. San Francisco: W. H. Freeman, 1973.

[23] R. Krishnapuram, A. Joshi, and L. Yi, "A fuzzy relative of the k-medoids algorithm with application toweb document and snippet clustering," *in Proceedings of IEEE Intl. Conf. Fuzzy Systems*, 1999.

[24] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. London: Prentice-Hall, 1982.

[25] L. Kaufman and P. Roussenw, *Finding Groups in Data: An Introduction to Cluster Analysis*. US: John Wiley and Sons, 1990.

[26] E.-G. Talbi, L. Vermeulen-Jourdan, and C. Dhaenens, "Clustering nominal and numerical data: a new distance concept for an hybrid genetic algorithm," in *Proc. 4th European Conf. Evolutionary Computation on Combinatorial Optimization (LNCS 3004)*, (Coimbra, Portugal), pp. 220–229, April 2004.

[27] S. Bandyopadhyay, "Simulated annealing using a reversible jump markov chain monte carlo algorithm for fuzzy clustering," *IEEE Transactions on Knowledge and data Engineering*, vol. 17, no. 4, pp. 479–490, 2005.

[28] S. Bandyopadhyay, U. maulik, and M. Pakhira, "Clustering using simulated annealing with probabilistic redistribution," *Int. J. Patt. Rec. Art. Int*, vol. 15, no. 2, pp. 269–285, 2001.

[29] U. Maulik, A. Mukhopadhyay, and S. Bandyopadhyay, "Effcient clustering with multi-class point identification," *Journal of 3D Images*, vol. 20, pp. 35–40, 2006.

[30] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1650–1654, 2002.

[31] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *J. Comp. App. Math*, vol. 20, pp. 53–65, 1987.

[32] K. Y. Yeung and W. L. Ruzzo, "An empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 17, pp. 763–774, 2001.

[33] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*. 2nd ed., 1999.