International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 27, No. 2 (2013) 1350005 (22 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218001413500055

OPTIMIZING FUZZY CLUSTER ENSEMBLE IN STRING REPRESENTATION

HOSEIN ALIZADEH*, BEHROUZ MINAEI-BIDGOLI† and HAMID PARVIN‡

Computer Engineering Department
Iran University of Science and Technology
Tehran, Iran
**halizadeh@iust.ac.ir*
†b_minaei@iust.ac.ir
‡parvin@iust.ac.ir

Received 3 September 2012
Accepted 31 January 2013
Published 10 April 2013

In this paper, we present a novel optimization-based method for the combination of cluster ensembles. The information among the ensemble is formulated in 0-1 bit strings. The suggested model defines a constrained nonlinear objective function, called fuzzy string objective function (FSOF), which maximizes the agreement between the ensemble members and minimizes the disagreement simultaneously. Despite the crisp primary partitions, the suggested model employs fuzzy logic in the mentioned objective function. Each row in a candidate solution of the model includes membership degrees indicating how much data point belongs to each cluster. The defined nonlinear model can be solved by every nonlinear optimizer; however; we used genetic algorithm to solve it. Accordingly, three suitable crossover and mutation operators satisfying the constraints of the problem are devised. The proposed crossover operators exchange information between two clusters. They use a novel relabeling method to find corresponding clusters between two partitions. The algorithm is applied on multiple standard datasets. The obtained results show that the modified genetic algorithm operators are desirable in exploration and exploitation of the big search space.

Keywords: Fuzzy cluster ensemble; nonlinear objective function; genetic algorithm; optimization.

1. Introduction

Data clustering is an essential and also difficult non-polynomial-hard (NP-hard) problem. The objective of clustering is to group a set of unlabeled objects into homogeneous groups or clusters (Jain *et al.*, 1999).[1,16] Each clustering algorithm optimizes its internal objective function which causes to find clusters with specific

*Corresponding author.

shapes. For example, k-means attempts to minimize sum of square errors (distances) between the data points and their corresponding cluster centers. Because minimizing the sum of square errors will result in globular clusters, k-means is a proper algorithm for datasets with spherical shapes. Generally, it will be suitable to apply each algorithm only on a special kind of dataset in which data distribution is fitted to its objective function. In other words, because there is no prior knowledge about cluster shapes, it is not easy to choose a specific clustering method for each dataset.[21] Knowing this result, one can immediately think about a combination of various clustering algorithms with different objectives instead of using a single one, hence, the appearance in a cluster ensemble context.

Cluster ensemble methods attempt to find more accurate, stable, and robust clustering solutions by fusing information from several primary data partitionings. Because different clustering algorithms look into different points of view over data, they can generate diverse partitionings of the same data. By combining the partitionings obtained from different base algorithms and by taking the strengths of each, it is possible to reach an efficient consensus partitioning, even when the clusters are not dense and well separated.[11]

Generally, there are two main steps in cluster ensemble.[1,4] In the first step, a number of base clustering algorithm are employed to provide a set of weak partitionings, which is called an ensemble. Every primary partitioning reveals an aspect about the data. Therefore, the primary results in the ensemble should be as diverse as possible to extract more information about the underlying patterns in the data. The information accumulated in the ensemble is combined in the next step of the cluster ensemble. In this paper, we are interested in the combination of spurious clustering results in order to produce a better consensus partition. The objective function used in the paper is inspired from the immature formulation introduced by Singh *et al.*[24] It maximizes the agreement between the ensemble members and also minimizes the disagreement simultaneously. Improving their nonlinear binary goal function, we propose a constrained nonlinear fuzzy objective function. The good exploration power of genetic algorithm in big search spaces persuaded us to use it for solving the model. However, without losing the generality, one can use any other optimizer to solve the model. There are two main schools of thought dealing with genetic algorithm solver. To maximize the advantages of the genetic algorithm, the suitable crossover and mutation functions should be defined.

As a summary, the main contributions of this work are the following:

(1) The proposed algorithm uses a state-of-the-art encoding for the cluster ensemble problem, i.e. the string representation which will be introduced in Sec. 3.
(2) The algorithm introduces a fuzzy objective function which yields to effectively optimizing the final partition variables.
(3) The algorithm employs a modified mutation function which strengthens the exploitation power of the genetic algorithm, especially on the last generations of genetic algorithm when the chromosomes converge structurally.

(4) Furthermore, the effect of using each proposed crossover operator on the speed of algorithm convergence is empirically studied.

Usually, the objective functions defined in the other optimization formulation of the both single and ensemble problems try to optimize only one aspect of the original clustering definition. Some of them increase the inter-cluster distances, whereas others attempt to decrease the intra-cluster variances. For example, the formulations to optimize the minimum sum-of-squares clustering (MSSC)[27] or structural-entropy-minimization-based clustering[12] only minimize their objective values. In the case of ensemble formulation, the EXAMCE procedure also tries to optimize the MSSC criterion during its two-level optimization process.[6] Despite the previous problem formulations and objective functions, the first above-mentioned point follows that string representation gives the ability of working on both maximizing assents out of ensemble members and minimizing dissents out of them, simultaneously.

We empirically compare the accuracy of the consensus partitioning with the accuracies of the best and the mean of the individual members in the ensemble. The accuracy is computed based on the true ground labels. Furthermore, we present the accuracy of the consensus partitioning obtained via evidence accumulation clustering (EAC).[9] The accuracies of the partitionings obtained via some basic clustering algorithms, namely, fuzzy c-means clustering (FCM) algorithm and single linkage hierarchical clustering algorithm, are also presented to support our evaluation process more. To show how the quality of a partitioning is related to the fitness function defined for the optimization problem, the paper also provides the readers with a fitness function corresponding to any reported accuracy. The admissible experimental results confirm our idea about employing fuzziness and choosing the genetic algorithm as an optimizer. This will be discussed in more detail in the next sections.

The rest of the paper is organized as follows: Section 2 reviews the recent related works in cluster ensemble problem. Section 3 presents the problem definitions. Section 4 introduces fuzzy string cluster ensemble optimized by genetic algorithm (FSCEOGA) as the problem solver. The modified operators of genetic algorithm are also introduced in Sec. 3 for FSCEOGA. In Sec. 5, we present a large number of experimental results performed on many diverse datasets and compare them directly with previously well-known proposed methods. Finally, we conclude the paper in Sec. 6 with a list of directions for future research.

2. Literature Review

In this section, we review some of the state-of-the-art studies in the fields of cluster ensemble approaches and genetic-algorithm-based cluster ensemble.

2.1. *New approaches in cluster ensemble*

There are two new trends in cluster ensemble approaches: cluster ensemble selection and cluster ensemble optimization.

In the first approach, the idea is to select a subset of base clusterings so that the consensus partition derived from the subset is better than the full ensemble. In the most previous studies, all partitionings and their clusters in the ensemble have equal weight. This means that every ensemble member has the same value in the final decision.[9,25] As a general principle, it seems that weighing the better ideas effectuates final decisions of the ensembles. Therefore, Fern and Lin[8] have utilized the normalized mutual information (NMI) criterion, first defined by Strehl and Ghosh[25] and further completed by Fred and Jain,[9] in order to evaluate the primary partitionings. Then, they have shown by comprehensive experimental results that selecting a subset of partitionings can yield better results than that of whole partitionings in the full ensemble. Moreover, Azimi and Fern[5] have shown that choosing better primary results based on NMI will not always yield better final results. They have also suggested an adaptive approach to choose a special subset of base results for each kind of datasets. Furthermore, showing the drawbacks of NMI, Alizadeh *et al.*[2] have introduced a new benchmark to evaluate the individual clusters called the Alizadeh−Parvin−Moshki−Minaei (APMM) criterion. They have extended the idea of cluster ensemble selection from the level of partitions to individual clusters. Considering the other point of view of the cluster ensemble selection, Parvin *et al.*[19] have proposed a new method for clustering data so as to assign a weight vector to the feature space of the data. In this method, calculating the data variance through every feature, the feature in which variance is higher participates in combination with greater weight. They have also proved the convergence of their suggested algorithm.

In the second approach, the consensus partition is obtained by the solution of an optimization problem. The goal of the optimization problem is finding the optimal partition (by optimizing an objective function) with respect to the cluster ensemble. A common feature in most of the previous approaches is to rely on modeling an instance of the cluster ensemble problem as a graph comprising n (where n is the number of dataset) nodes and some edges. An edge indicates some measure of similarity calculated from the ensemble between two nodes. The graph representation of an ensemble, regardless of the sophistication of the algorithm to work on, will likely cause sub-optimal results. More recent researches in the cluster ensemble field show a tendency to formulate the problem as an optimization task and then solving it using mathematical solvers (or even intelligent optimization solvers).[6,10,20,24] A brief review over some of these methods is available in Ref. 26.

Christou[6] has proposed an optimization-based formulation for the combination of cluster ensembles for the class of problems with intra-cluster criteria, such as MSSC. He modified the set partitioning formulation of the original clustering problem[7] to reach a simple and efficient cluster ensemble algorithm. He has also confirmed that under general assumptions and relaxations of the original formulation, it is guaranteed to find better solutions than the ones in the ensemble. Singh *et al.*[24] have provided another optimization formulation for the formation of the final clusters so as to maximize the agreement and minimize the disagreement of the consensus result with respect to the ensemble members simultaneously. They have also proposed a new encoding for ensemble members named A-string representation. In the next step, they relaxed their initial formulation of the nonlinear binary program to a 0-1 semidefinite programming (SDP) problem. This problem is then further relaxed to give a final SDP. After that, they have used a rounding scheme based on a winner-take-all approach to produce a final feasible clustering. Their results show that their suggested idea performed better than the base clustering solutions used in terms of classifying error for most of the test cases. Despite most cluster ensemble techniques using a large set of weak primary results, their experimental results employed only a few, but accurate, base clusterings.

2.2. *Genetic algorithm-based cluster ensemble*

The genetic algorithm has shown its versatility in solving any optimization problem. Wan *et al.*[28] have argued the transformation of speed construction and hypocenters position in the Beijing−Tianjin−Tangshan−Zhangjiakou area into a genetic algorithm optimization problem. They then used a genetic algorithm to solve the problem. A camera vision system concentrating on 3D reconstruction has been introduced by Zhang *et al.*,[33] who have modified genetic algorithm to approximate the system parameters. A genetic algorithm model is developed and used for optimizing their objective function. Louati *et al.*[13] have used genetic algorithm as the optimizer for the water allocation to demand centers and the salinity level of the water supply to end users. They combined the two objective functions into one and solved it using a genetic algorithm. Xu and Cai[30] have used genetic algorithm to solve the problem of how to determine expert weights in multiple attribute group decision making. They have proposed a general nonlinear optimization model based on deviation function. Then they have employed a genetic algorithm so as to optimize their nonlinear optimization model and discover the best weights.

The genetic-algorithm-based cluster ensemble methods use its search capability to obtain a robust consensus clustering. Generally, the initial population is generated with the partitions in the cluster ensemble. Moreover, a fitness function is defined to determine which chromosomes (partitions) are better. Among these methods, we should advert to Yoon *et al.*[31,32] They have employed genetic algorithm as a consensus function for the cluster ensemble problem. Each partition in their method is encoded by a chromosome. With each pair of partitions obtained from the objects, an ordered pair is created. In this algorithm, the fitness function compares the amount of overlaps between the partitions in each chromosome. Another cluster ensemble method based on genetic algorithms is the method proposed by Luo *et al.*[14] This method uses genetic algorithm to minimize an information theoretical criterion. It also uses the Hungarian method to solve the label correspondence problem. Furthermore, Analoui and Sadighian[3] have proposed a probabilistic model by using a finite mixture of multinomial distributions. The consensus partition is found as a solution to the corresponding maximum likelihood problem using a genetic algorithm.

3. Problem Definition

String representation, first introduced by Singh *et al.*,[24] is one of the recent approaches to accumulate information from an ensemble. Each data point is figured as a 3D matrix determining the base algorithms' point of view.

Definition 1 (Cluster Ensemble Problem). Given a dataset $D = (x_1, x_2, \ldots, x_n)$, where x_i is the ith data point in a d-dimensional feature space, a set of clustering solutions $E = (C_1, C_2, \ldots, C_m)$ obtained from m different clustering algorithms or only one algorithm by perturbing the input dataset or modifying the algorithm parameters, is called an *ensemble*. Each solution, $Cj = (C_{1j}, C_{2j}, \ldots, C_{kj})$, is the partitioning of the data into k clusters where C_{ij} denotes the cluster i from the jth partitioning. The *Cluster Ensemble* problem finds the optimum partitioning which partitions D into k clusters that maximize the shared information in E.

Definition 2 (String Representation of the Ensemble). Given an ensemble $E = (C_1, C_2, \ldots, C_m)$, the string representation of the ensemble is a 3D space $A(1 \ldots n \times 1 \ldots k \times 1 \ldots m)$, where each element $A(l, i, j)$ denotes the assignment of x_l to C_{ij} in E. In other words, we can define the $A = [A(l, i, j)]$ as the following:

$$A(l,i,j) = \begin{cases} 1 & \text{if } x_l \text{ is assigned to } C_{ij} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

From Eq. (1), it can immediately be understood that each 2D matrix A_l stands for data sample x$_l$. In fact, the feature vector is here changed to a matrix. Put it differently, the representation of a solitary data point changes from an original Euclidean 1D space in D to another 1D ensemble integer space of primary results in E and then to a 2D binary matrix in A.

Definition 3 (Final Fuzzy Partition). The final partition (cluster ensemble solution) is defined to be a fuzzy partition comprising a set of fuzzy clusters $X = (C_1^*, C_2^*, \ldots, C_k^*)$. This fuzzy variable is a 2D matrix which determines the membership amounts of data points to the clusters. Put differently, we can define $X(1 \ldots n \times 1 \ldots k)$ as the following:

$$X = [X(l,p)], \quad \text{where } X(l,p) \text{ is the membershhip of } x_l \text{ to } C_p^*. \tag{2}$$

The aim of our optimization process is to find X optimally based on the similarity measure which is discussed in Definition 4. According to the definition of the matrix X, each row determines the membership values of a data point to the clusters. Therefore, we define a constraint to ensure that sum of memberships of each sample to all clusters is equal to one. It means the constraint: $\sum_{p=1}^{k} X(l,p) = 1,\ \forall\, l \in \{1, \ldots, n\}$. Furthermore, to guarantee that no cluster will remain empty, another constraint is required: $\sum_{l=1}^{n} X(l,p) \geq 1\ \forall\, p \in \{1, \ldots, k\}$.

Definition 4 (Fuzzy Cluster Centers). Given the A-strings and the membership matrix X, a 3D variable $S(1 \ldots k \times 1 \ldots m \times 1 \ldots k)$ is defined, which holds the similarity of primary clusters to the clusters in final partition X. More precisely, each entry of the matrix S is defined: $S(i,j,p) = \text{similarity } (C_{ij} \text{ to } C_p^*)$. We define the *similarity* function as the following equation:

$$S(i,j,p) = \frac{((\sum_{i=1}^{n} d(i,j,p)) - d(i,j,p))^{0.5+I/2}}{\sum_{p=1}^{k} ((\sum_{i=1}^{n} d(i,j,p)) - d(i,j,p))^{0.5+I/2}}, \quad \text{so that } d(i,j,p) \text{ is the } I\text{th maximum } \forall\, i, \tag{3}$$

where $d(i,j,p)$ is the distance between clusters C_{ij} and C_p^*; formally $d(i,j,p) = \|C_{ij} - C_p{}^*\|$. It is computed by Eq. (4):

$$d(i,j,p) = \sum_{z} |A(z,i,j) - X(z,p)|. \tag{4}$$

By the way of explanation, $d(i,j,p)$ is the distance between the ith cluster of jth partitioning and pth cluster of the final partition. The 3D matrix S is a kind of fuzzy cluster center for two reasons: First, it has the same dimension as the A strings. Second, the number of cluster centers which is the first dimension of the matrix S is equal to k.

To calculate the similarity between clusters C_{ij} and C_p^*, first the distance is converted to a kind of similarity by subtracting each $d(i,j,p)$ from sum of distances over all clusters in the jth partition (as it is shown in Eq. (3)). Then, the obtained similarity is normalized by dividing the sum of similarities over all clusters. This simple normalization way causes the values to be close together. For the sake of increasing contrast between values, we power the similarities to different exponents. The lower the similarity value, the smaller the power.

To guarantee that the sum of similarities between whole clusters existed in the jth partition and C_p^* is equal to one, it is normalized by the denominator, that is,

$$\sum_{i=1}^{k} S(i,j,p) = 1 \quad \forall\, j \in \{1,\ldots,m\}, \quad \forall\, p \in \{1,\ldots,k\}.$$

Definition 5 (Fuzzy String Objective Function (FSOF)). Given the A-strings, the membership matrix X, and the fuzzy cluster centers S, variable FSOF contains the absolute distance between the fuzzy cluster centers S and the average of A-strings belonging to the cluster C_p^*. In other words, the value returned from the absolute operator of FSOF equation (Eq. (5)) indicates the percentage of memberships of data points in C_p^* that disagree with the greater part of memberships in the cluster (with respect to the clustering solution provided by C_j). Put differently, the FSOF optimization maximizes the agreement and minimizes the disagreements between data points belonging to a cluster, simultaneously. The following constraints, as discussed previously, enforce the solution to remain feasible.

$$\text{FSOF} = \sum_{p}^{k} \sum_{i}^{k} \sum_{j}^{m} \left| S(p,i,j) - \frac{\sum_{l=1}^{n} A(l,i,j) X(l,p)}{\sum_{l=1}^{n} X(l,p)} \right|$$

s.t.

$$S(i,j,p) = \frac{((\sum_{i=1}^{n} d(i,j,p)) - d(i,j,p))^{0.5+I/2}}{\sum_{p=1}^{k}((\sum_{i=1}^{n} d(i,j,p)) - d(i,j,p))^{0.5+I/2}},$$

$$\text{so that } d(i,j,p) \text{ is the } I\text{th maximum } \forall\, i$$

$$d(i,j,p) = \sum_{l=1}^{n} |A(l,i,j) - X(l,p)|$$

$$\sum_{i=1}^{k} S(i,j,p) = 1 \quad \forall\, j \in \{1,\ldots,m\}, \quad \forall\, p \in \{1,\ldots,k\}$$

$$\sum_{p=1}^{k} X(l,p) = 1 \quad \forall\, l \in \{1,\ldots,n\}$$

$$\sum_{l=1}^{n} X(l,p) \geq 1 \quad \forall\, p \in \{1,\ldots,k\}$$

$$0 \leq X(l,p) \leq 1 \quad \forall\, l \in \{1,\ldots,n\}, \quad \forall\, p \in \{1,\ldots,k\}. \tag{5}$$

4. Problem Solver

The proposed solver named the FSCEOGA is introduced in this section. At the cost of repetition, the cluster ensemble is formulated as an optimization problem. The goal of the optimization problem is to minimize the fuzzy string objective function that implicitly yields to a partitioning in which the agreements among base partitionings are maximized while the disagreements among them are minimized simultaneously. To solve the proposed model, any nonlinear optimization solver can be employed. The easy-to-understand as well as effective-in-exploration/optimizer named genetic algorithm is used as problem solver in this paper.

Genetic algorithms have been used in many optimization problems as a general search method. The algorithm is able to search the complex search spaces and usually obtain an optimal or a near-optimal solution. The ability of the genetic algorithm has been shown in many different issues. Among them, we can point out the utilizing genetic algorithms in classification problems, neural network training and speech recognition systems training.[15] In all these issues with an appropriate definition of genetic algorithm, it has been able to achieve the proper results in admissible time. There are also vast researches in employing genetic algorithm for solving the clustering problems. Genetic algorithm has been used in clustering and classification problems successfully. Sheng and Liu[22] have proposed a modified genetic algorithm for data clustering problem. Some heuristic operators are considered and used with the genetic algorithm to better solve the problem. Wang *et al.*[29] have proposed three algorithms based on genetic algorithm to create a classifier system. Partitional clustering has been dealt with by Sheng *et al.*[23] considering the reliability and efficiency of high-quality solutions using a niching genetic k-means algorithm. They have used the sum of squared errors as their main objective function.

The first step in solving a problem by genetic algorithm is to code it in a way that it can be applied to genetic algorithm. In the proposed algorithm, fuzzy encoding of matrix X (as it is defined in Eq. (2)) is used as the chromosomes of genetic algorithm. It means that each chromosome in FSCEOGA is a matrix of $n \times k$. In our implementation, *init_pop*() function generates a set of random feasible solutions as the initial population and feeds it to the genetic algorithm. It includes a function named *feasible*() to check the feasibility of the randomly made potential solutions. The function *feasible*() checks the feasibility of its input chromosome to be sure about satisfying the two following constraints:

$$\sum_{p=1}^{k} X(l,p) = 1 \quad \forall\, l \in \{1,\ldots,n\},$$

$$\sum_{l=1}^{n} X(l,p) \geq 1 \quad \forall\, p \in \{1,\ldots,k\}.$$

The three most important elements of a genetic algorithm which should be discussed further than the others are fitness, crossover, and mutation functions.

4.1. *Fitness function*

The fitness function of FSCEOGA evaluates the FSOF defined in Eq. (5). Given the A-strings and the input chromosome, first, the fitness function calculates the fuzzy cluster centers from Eq. (3). Then, it computes the absolute difference between the fuzzy cluster centers and the average A-strings which belong to the corresponding clusters as Eq. (5).

4.2. *Crossover*

The crossover function combines two input solutions in order to create an offspring which is inherited from the two input solutions also named its two parents. We proposed two different crossover functions named *Cross_Twop*() and *Cross_Clust*(). The first one, which is a modification of the original two-point crossover of MATLAB, selects two random integer values *point* 1 and *point* 2 between 1 and n where n is the number of genomes. Then, it splits each parent into three parts at the points of *point* 1 and *point* 2. After that, it makes the offspring by copying the first and the last parts from the first parent and the middle part from the other. The pseudo code of the first defined crossover function is depicted in Fig. 1.

To better understand how the *Cross_Twop*() function works, see Example 1.

```
Function child = Cross_Twop(p1,p2)    {
        Generate two random points as point1 and point2 subject to 1≤point1≤point2 ≤ n.
        child = p1;
        child (point1:point2) = p2(point1:point2);
        if (~feasible(child))    {
                child = p2;
                child (point1:point2) = p1(point1:point2);
                if (~feasible(child))    {
                        child = p1;
                }
        }
}
```

Fig. 1. *Cross_Twop*() function.

Example 1. Suppose that $p1$ and $p2$ are the parents and the crossover points are 2 and 4. The function returns the following child:

$$p1 = \begin{bmatrix} 0.6 & 0.1 & 0.3 \\ 0.7 & 0.2 & 0.1 \\ 0.1 & 0.5 & 0.4 \\ 0.9 & 0 & 0.1 \\ 0.1 & 0.2 & 0.7 \end{bmatrix} \quad p2 = \begin{bmatrix} a & b & b \\ d & e & f \\ g & h & i \\ j & k & l \\ m & n & o \end{bmatrix}$$

$$\begin{matrix} \text{point } 1 = 2 \\ \text{point } 2 = 4 \end{matrix} \Rightarrow \text{child} = \begin{bmatrix} 0.6 & 0.1 & 0.3 \\ 0.7 & 0.2 & 0.1 \\ g & h & i \\ j & k & l \\ 0.1 & 0.2 & 0.7 \end{bmatrix}$$

The second crossover function *Cross_Clust*() takes two parents and then produces one child. The child inherits one cluster from the second parent while it inherits the rest clusters from the first one. Figure 2 shows the procedure of the second suggested crossover function.

```
Function child = Cross_Clust (p1,p2)        {
        ReLabeled_p2=Fuzzy_relabling(p1,p2);
        Generate a random cluster number, clus_num, subject to 1≤ clus_num ≤ k.
        Index=Find the data points belonging to cluster clus_num in p1;
        child = p1;
        child = Copy and replace  ReLabeled_p2(Index) into child;
        if (~feasible(child))       {
                ReLabeled_p1=Fuzzy_relabling(p2,p1);
                Generate a random cluster number, clus_num, subject to 1≤ clus_num ≤ k.
                Index=Find the index of cluster clus_num in p2;
                child = p2;
                child = Copy and replace  ReLabeled_p1(Index) into child;
                if (~feasible(child))       {
                        child = p1;
                }
        }
}
```

Fig. 2. *Cross_Clust*() function.

To make the *Cross_Clust*() function clear, see Example 2.

Example 2. Suppose that we have two parents $p1$ and $p2$ and the $clus\_num = 2$. The *Cross_Clust*()crossover function is illustrated as follows:

$$\left.\begin{array}{r} p1=\begin{bmatrix}0.8 & 0.1 & 0.1\\ 0.9 & 0.1 & 0\\ 0 & 1 & 0\\ 0.9 & 0 & 0.1\\ 0.1 & 0 & 0.9\\ 0 & 0.8 & 0.2\end{bmatrix} \\ p2=\begin{bmatrix}0.1 & 0.8 & 0.1\\ 0 & 0.9 & 0.1\\ 0.1 & 0.9 & 0\\ 1 & 0 & 0\\ 0 & 0.1 & 0.9\\ 0.7 & 0.1 & 0.2\end{bmatrix}\xrightarrow{\text{Relabling P2}}\text{Relabeled\_}p2=\begin{bmatrix}0.8 & 0.1 & 0.1\\ 0.9 & 0 & 0.1\\ 0.9 & 0.1 & 0\\ 0 & 1 & 0\\ 0.1 & 0 & 0.9\\ 0.1 & 0.7 & 0.2\end{bmatrix}\end{array}\right\}\begin{array}{c}\text{clus\_num}=2\\ \Rightarrow\end{array}\quad \text{child}=\begin{bmatrix}0.8 & 0.1 & 0.1\\ 0.9 & 0.1 & 0\\ \hline 0.9 & 0.1 & 0\\ \hline 0.9 & 0 & 0.1\\ 0.1 & 0 & 0.9\\ \hline 0.1 & 0.7 & 0.2\end{bmatrix}$$

In Example 2, choosing $clus\_num = 2$ causes replacing the rows *in* $p1$ dedicated to the cluster $clus\_num = 2$ with the corresponding rows in $ReLabeled\_p2$. In this example, data points in the rows indexed by 3 and 6 in *p1* belong to the second cluster. Therefore, the procedure replaces those rows (rows three and six) from $ReLabeled\_p2$ in child which was first created as a copy of $p1$. If the procedure does not lead to a feasible solution, the function will be repeated by exchanging $p1$ and $p2$. If the second attempt does not result in a feasible answer, *Cross_Clust* will return $p1$.

The function $Fuzzy\_relabling(p1, p2)$ relabels its second fuzzy input $p2$ based on its first fuzzy input $p1$. The first option to relabel a fuzzy partition is transforming it to a crisp partition. Then, we apply one of the common relabeling methods on the achieved crisp partition. One downside of this way is that we miss some information among the fuzzy memberships because the converted crisp partition deals only with the maximum membership. Maintaining the information, we propose a new method for fuzzy relabeling of two fuzzy partitions.

On the other hand, the problem is finding matching between fuzzy partitions. This problem is immediately reduced to calculate the similarity between two fuzzy clusters and, consequently, to find a match of their components. Therefore, the first step of our fuzzy relabeling method is calculating similarity between two membership values belonging to a data point from two different clusters. At first glance, the term $1 - |p1(c,i) - p2(c,j)|$, where $p1(c,i)$ is the fuzzy membership of the cth data point to ith cluster in partitioning $p1$, seems to be a measure for calculating similarity (matching). We name it "*simple matching*." In our opinion, however, the amount of matching should be dependent on the certainty (membership) of them. For example, the simple matching of two components with memberships 0.1, 0.2 is $1 - |0.1 - 0.2| = 0.9$. In a similar way, the simple matching has the same value for inputs 0.8 and 0.9, that is, $1 - |0.8 - 0.9| = 0.9$. We believe that membership degrees denote the certainty level of classifying a sample to a cluster. Furthermore, low degrees only mean that the sample probably belongs to the other clusters. Moreover, we will calculate the calculations for all other clusters. As a result, it seems to be more rational that simple matching be associated with membership degrees. We suggest Eq. (6) to calculate matching between two fuzzy clusters.

$$M(i,j) = \sum_{c=1}^{n} \min(p1(c,i), p2(c,j)) \times (1 - |p1(c,i) - p2(c,j)|), \tag{6}$$

where $M(i,j)$ aims to count the similarities between membership distributions of cluster i from $p1$ and cluster j from $p2$. Figure 3 demonstrates our relabeling procedure. The Hungarian algorithm[17] calculates the best matching between the clusters in M.

To make our proposed relabeling function easier to understand, see Example 3.

```
Function ReLabeled_p2=Fuzzy_relabling(p1,p2)        {
    For i=1: number of clusters in p1        {
        For j=1: number of clusters in p2        {
            $M(i,j)=\sum_{c=1}^{n}\min(p1(c,i),p2(c,j))\times(1-|p1(c,i)-p2(c,j)|)$
        }
    }
    Index=Hungarian(M);
    ReLabeled_p2=[];
    for i=1: number of clusters in p2        {
        ReLabeled_p2 (:,i)=p2(:,Index(i));
    }
}
```

Fig. 3. *Fuzzy_relabling*() function.

Example 3. Suppose that the goal is relabeling fuzzy partitioning $p2$ with respect to $p1$ (ground truth). The *Fuzzy_relabling*$(p1, p2)$ procedure is illustrated as follows:

$$p1=\begin{bmatrix}0.8&\boxed{0.1}&0.1\\0.9&\boxed{0.1}&0\\0&\boxed{1}&0\\0.9&\boxed{0}&0.1\\0.1&\boxed{0}&0.9\\0&\boxed{0.8}&0.2\end{bmatrix}\quad p2=\begin{bmatrix}0.1&0.8&\boxed{0.1}\\0&0.9&\boxed{0.1}\\0.1&0.9&\boxed{0}\\1&0&\boxed{0}\\0&0.1&\boxed{0.9}\\0.7&0.1&\boxed{0.2}\end{bmatrix}\Rightarrow M=\begin{array}{cc} & p2\\ p1 & \begin{bmatrix}0.84&\mathbf{1.8}&0.07\\\mathbf{0.74}&0.89&\boxed{(0.28)}\\0.21&0.14&\mathbf{1.2}\end{bmatrix}\end{array}\xrightarrow{Hungarian(M)}[2\quad 1\quad 3]\Rightarrow \mathrm{Re}Labeled\_p2=\begin{bmatrix}0.8&0.1&0.1\\0.9&0&0.1\\0.9&0.1&0\\0&1&0\\0.1&0&0.9\\0.1&0.7&0.2\end{bmatrix}$$

Comparing cluster 2 of $p1$ and cluster 3 of $p2$:

$$M[2,3]=0.1\times(1-0)+0.1\times(1-0)+0+0+0+0.2\times(1-0.6)$$
$$=0.1+0.1+0.08$$
$$=0.28$$

In Example 3, we construct the matching matrix M from Eq. (6). Then, it will be served to the Hungarian algorithm to find the best possible matching.

4.3. *Mutation*

The mutation function causes the solution to jump a bit in the feasible solution. The proposed mutation function, which is shown in Fig. 4, randomly changes the memberships of a data point to the clusters. It seems that changing the memberships

```
Function new_child=mutation(child)    {
        Generate a random Index, subject to 1≤ Index ≤ n.
        Generate a random number Chance, subject to 0≤ Chance≤ 1.
        counter=1;     found=false;
        While (~found and counter<5)  {
                if (Chance<0.5)
                        new_Child=reproduce_crisp(child,Index);
                else
                        new_Child=reproduce_fuzzy(child,Index);
                }
                found=feasible(new_child);
                counter++;
        }
}
```

Fig. 4. *Mutation*() function.

in a random selected row is a rational way of mutation. Hence, our mutation function replaces the selected fuzzy genome to another random generated vector with the same length. Whereas a data point may strictly belong to a solitary cluster and this phenomenon often occurs, the suggested mutation function should generate crisp assignments for some data points.

In Fig. 4, *reproduce_crisp*() creates a vector of size k so that all elements are zero except one. On the other hand, *reproduce_fuzzy*() makes a vector of random variables so that each variable is extracted randomly from the interval [0,1]. The summation of the values of the mentioned variable vector should be equal to one.

5. Experimental Results

This section aims to evaluate the performance of FSCEOGA experimentally. The proposed method has been examined on seven different datasets which are varying in number of clusters, features and examples. The results of some other clustering algorithms are also reported on the same datasets. Six of the used datasets are from UCI repository of machine learning.[18] The last dataset is the well known and hard-for-clustering HalfRing. Due to the high number of datasets, the experimental results can be reliable and general. Table 1 gives information about the used standard datasets.

For all of the mentioned benchmarks, the number of clusters and real labels are already known. The results are reported in terms of the accuracy. To calculate the accuracy for each obtained partition the two following steps are done. First, the

Table 1. Characteristics of used datasets.

	Class	Features	Samples
Halfrings	2	2	400
Iris	3	4	150
Wine	3	13	178
Ionosphere	2	34	351
SAHeart	2	9	462
Glass	6	9	214
Breast	2	9	683

Hungarian algorithm is employed for matching the partition with the true labels of the dataset.[17] After finding the proper matching, the accuracy is obtained by computing the percentage of samples which are correctly classified. The other way to evaluate a partition is through its FSOF value. The FSOF for a partition is calculated based on Eq. (5).

The summary of the results obtained by employing different clustering algorithms is presented in terms of accuracy in Table 2. All methods have been implemented in MATLAB 2012. To reach any result throughout the paper, each algorithm is run with 10 different initializations, and then the averaged accuracy is reported. Like many other cluster ensemble studies, this work employed the well known k-means algorithm with random initialization to generate the ensemble. Moreover, sampling 70% of data is used to enforce more diversity to the ensemble. Furthermore, the maximum number of iterations for running k-means is limited to gain weak as well as diverse partitions. The ensemble was generated with the known k extracted from the ground truth.

In Table 2, FSCEOGA1 stands for FSCEOGA when it uses *Cross_Twop*() as the crossover operator. Consequently, FSCEOGA2 stands for FSCEOGA when it uses *Cross_Clust*() as the crossover operator. The proposed methods are compared with two well known single clustering methods, Single Linkage and Fuzzy C-means. To extend the experimental evaluations, the famous EAC algorithm[9] and renowned

Table 2. The summary of the accuracies obtained by employing different clustering algorithms. The best results obtained by different methods over each data set are highlighted in bold.

			Ens. Members		Single		Ensemble Methods				
Accuracy	FSCEOGA1	FSCEOGA2	Best	Average	SL	FCM	EAC	HGPA	MCLA	CSPA	SDP
Iris	93.27	93.34	94.13	84.08	68.00	**96.00**	93.13	70.50	89.00	92.33	89.67
Wine	68.99	68.43	71.29	67.36	42.70	68.54	66.12	60.22	**71.12**	69.89	69.55
Glass	**50.37**	46.26	51.06	47.99	36.45	50.00	46.45	40.84	46.45	39.16	47.90
Halfrings	74.65	74.92	75.13	74.39	**75.75**	74.50	73.67	50.00	74.00	74.38	—
Ionosphere	69.63	70.04	69.69	67.16	64.39	70.09	64.87	53.99	**70.14**	68.23	—
SAHeart	**65.65**	65.63	65.74	64.72	65.15	64.94	65.28	52.16	64.23	65.31	—
Breast	95.14	**95.21**	95.79	95.14	65.15	70.13	**95.21**	50.37	94.05	83.02	—
Average of all datasets	**73.96**	73.40	74.69	71.55	59.66	70.60	72.10	54.01	72.71	70.33	—

Table 3. The summary of the FSOFs obtained by employing different clustering algorithms. The best results obtained by different methods over each data set are highlighted in bold.

			Ens. Members		Single		Ensemble Methods				
FSOF	FSCEOGA1	FSCEOGA2	Best	Average	SL	FCM	EAC	HGPA	MCLA	CSPA	SDP
Iris	9.11	9.09	10.10	15.34	28.00	15.10	13.77	31.14	**8.59**	10.24	10.37
Wine	16.70	**1.58**	15.68	20.07	36.59	22.44	28.10	32.66	18.77	17.58	28.27
Glass	**68.35**	71.48	70.38	90.57	166.23	78.77	134.44	123.91	98.17	129.57	125.23
Halfrings	5.43	**5.31**	4.65	6.30	31.40	12.41	7.39	33.86	5.85	5.86	—
Ionosphere	10.40	10.36	10.34	14.21	22.45	24.64	32.44	33.33	**9.90**	11.40	—
SAHeart	**7.17**	**7.17**	7.24	9.84	33.27	27.78	40.68	33.51	7.43	6.77	—
Breast	1.58	1.59	1.58	2.41	26.62	24.58	1.66	35.75	**1.07**	13.21	—
Average of all datasets	16.96	**15.23**	17.14	22.68	49.22	29.39	36.93	46.31	21.40	27.80	—

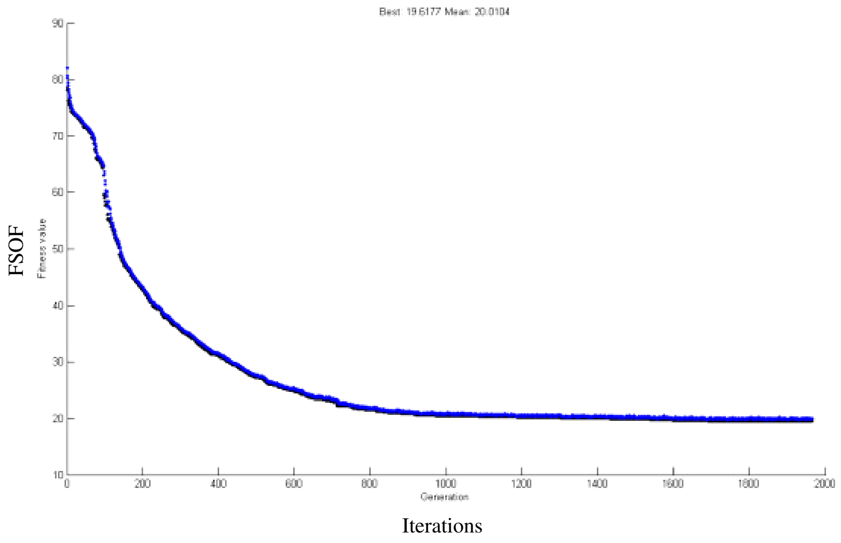
hyper graph based methods HGPA, MCLA and CSPA[25] are added to the basket of ensemble methods for comparison. We also used implementations of SDP method of Singh *et al.*[24] downloaded from "http://www.biostat.wisc.edu/∼vsingh/". It worked for Iris, Wine and Glass datasets; however, it had some errors when it was run for the other four test cases including Halfrings, Ionosphere, SAHeart and Breast. The reached accuracy and FSOF results are shown in Tables 2 and 3, respectively.

As has been inferred from Table 2, overall, the accuracy of FSCEOGA is the best among the accuracies of different clustering algorithms. However, the other methods may outperform the FSCEOGA in some of the datasets. For example, in the Breast dataset the EAC method is the best. Moreover, the hierarchical SL, a rather old clustering algorithm outperforms all other methods considering the Halfrings dataset. These cases may be observed because of some reasons. One of the most important reasons is that we have used only k-means as our base algorithm. On account of k-means finds only spherical clusters well and has serious problems with the non-globular shapes, the combinational results of k-means-generated ensemble may be affected with the non-globular-shaped datasets. On the other hand, hierarchical methods like SL which are excellent in solving the problems with continuous clusters, outperforms even the k-means-generated ensemble methods in such cases. Since the Halfrings benchmark has two separate continuous clusters, it is a digestible case for SL. This is the motivation that all studies in this field should compare the methods, by referring to a set of benchmarks instead of one or two datasets. Consequently, without loss of generality, the combinational methods like ours can compensate this downside by using a range of different base algorithms and other diversity enforcing methods. FSCEOGA outperforms all methods in terms of averaged accuracy among all case studies. However, FSCEOGA usually reaches better accuracy when it uses *Cross_Twop*() as the crossover operator instead of *Cross_Clust*().

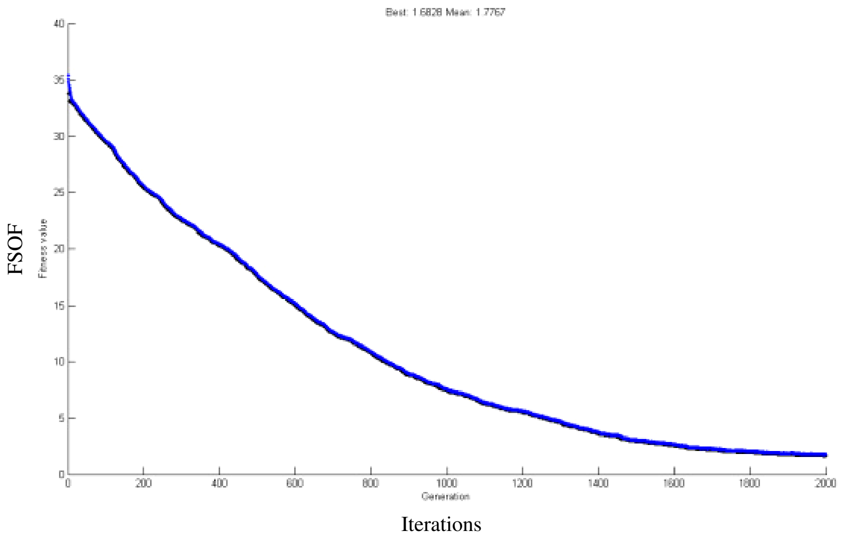
The summary of the results obtained by employing different clustering algorithms is presented in terms of FSOF in Table 3.

As concluded from Table 3, overall, FSOF of FSCEOGA is the best among the different clustering algorithms. The only closed competitor is MCLA method.

It shows that MCLA consensus function has an objective that is conceptually closed to FSOF. Consequently, FSCEOGA outperforms the other methods in terms of the averaged FSOF over all datasets. It is worthy to mention that in Tables 2 and 3 we have ignored the column *Best* due to its unavailability, that is, it is impractical to

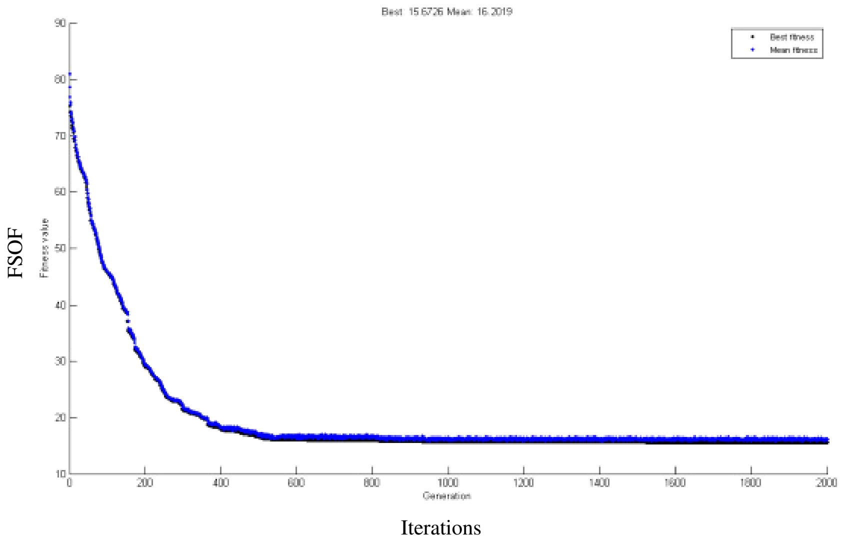


(a)

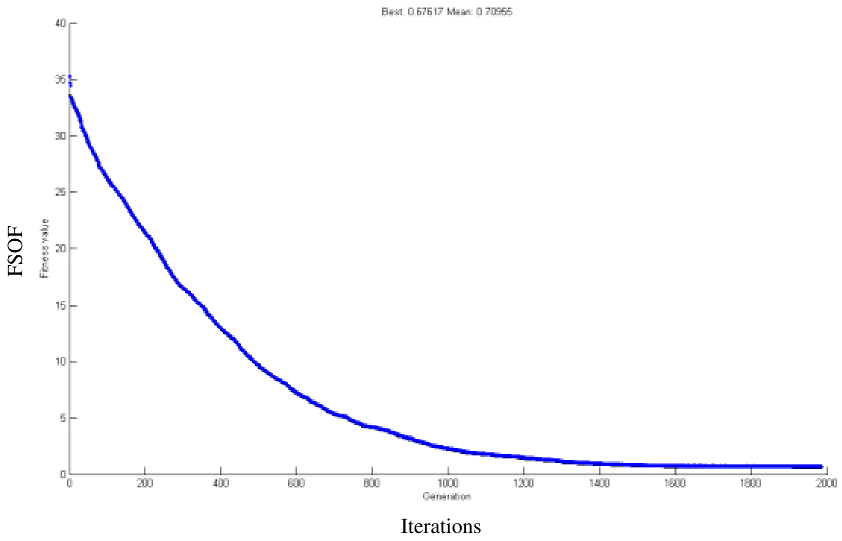


(b)

Fig. 5. The convergence of FSOF in the FSCEOGA when using (a) *Cross_Twop*() as the crossover operator over the Wine dataset; (b) over the Breast dataset; (c) *Cross_Clus*() as the crossover operator over the Wine dataset; (d) over the Breast dataset (color online).



(c)



(d)

Fig. 5. (*Continued*)

reach a method like *Best*. It is also worth mentioning that the *Best* column is reported only for comparing the best solution in the ensemble with the consensus solution.

Although FSCEOGA reaches better accuracy when it uses *Cross_Twop*() as the crossover operator (see Table 2), employing *Cross_Clust*() as the crossover function yields the FSCEOGA to touch FSOF in a lower degree (refer to Table 3). To show how deep the modified genetic algorithm operators affect the quality of the final solution, see Fig. 5. The method rapidly decreases the fitness function in the initial 500 generations to a great extent. This is a confirmation that the crossover operator is performing well. After dramatically reducing the fitness function in the initial 500 generations, the fitness function gradually stabilizes. Although it is still reduced in each successive generation, the amount of decrement is much lower than the first 500 generations. It is due to the convergence of the population in the genetic algorithm structurally. As you can observe, the fitness function reduces for a while after the convergence of the population. It means that the proposed mutation operator is capable of well exploiting the locality found by the population.

Therefore, the experimental results confirm the ability of modified genetic algorithm operators in handling the exploring/exploiting dilemma. While the crossover operator can help the modified genetic algorithm to explore the big search space overall and also to find the near-optimal localities, the mutation operator can help it find the best solution in any locality.

6. Conclusion and Future Work

In this paper, we have redefined the cluster ensemble problem and introduced an innovative fuzzy string representation of the cluster ensemble problem. In other words, the proposed formulation of the problem uses a string representation to encode information of the ensemble of primary results. The suggested formulation employs fuzzy logic to define a fuzzy objective function. Each candidate consensus partitioning (each candidate solution of the model) also uses a membership degree indicating how much data point belongs to each cluster. Finally, we have put the new formulation into a mathematical optimization model with some constraints. Although each nonlinear solver can be used to solve the model, the easy-to-understand as well as effective-in-exploration characteristics of genetic algorithm persuaded us to employ it as the optimizer. We have supported the genetic algorithm solver by well-suitable-to-the-problem crossover and mutation operators. The FSCEOGA has been examined on seven different datasets. The experimental results confirm the ability of modified genetic algorithm operators in handling the exploring/exploiting dilemma. While the crossover operator can help the modified genetic algorithm to explore the big search space overall and also to find the near optimal localities, the mutation operator can help the modified genetic algorithm find the best solution in any locality.

It is necessary to stress that we have opted for the genetic algorithm as the first choice for model solving only due to its great qualities; however, it is not a guarantee that it is the best one for this problem. In our future work we plan to investigate the use of the other solvers for our proposed model. This may include both mathematical and evolutionary solvers.

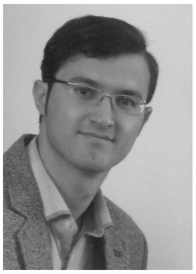
Acknowledgments

This work was supported in part by the Research Institute for ICT−ITRC grant program. We would like to appreciate the ITRC for this support.

References

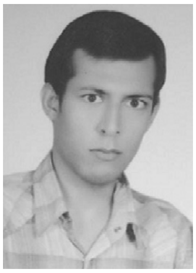
1. H. Alizadeh, H. Parvin, M. Moshki and B. Minaei-Bidgoli, A new clustering ensemble framework, *INCT 2011, CCIS* **241** (2011) 216−224.
2. H. Alizadeh, B. Minaei-Bidgoli and H. Parvin, Cluster ensemble selection based on a new cluster stability measure, *Intelligent Data Anal.* in press. **18**(3) (2014).
3. M. Analoui and N. Sadighian, Solving cluster ensemble problems by correlation's matrix & GA, *IFIP Int. Fed. Inform. Process.* **228** (2006) 227−231.
4. H. G. Ayad and M. S. Kamel, Cumulative voting consensus method for partitions with a variable number of clusters, *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(1) (2008) 160−173.
5. J. Azimi and X. Fern, Adaptive cluster ensemble selection, *Proc. Int. Joint Conf. Artificial Intellegence (IJCAI, 2009).*
6. I. T. Christou, Coordination of cluster ensembles via exact methods, *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(2) (2011) 279−293.
7. O. Du Merle, P. Hansen, B. Jaumard and N. Mladenovich, An interior point algorithm for minimum sum of squares clustering, *SIAM J. Sci. Comput.* **21**(4) (2000) 1484−1505.
8. X. Fern and W. Lin, Cluster ensemble selection, *Statis. Anal. Data Mining* **1**(3) (2008) 128−141.
9. A. Fred and A. K. Jain, Combining multiple clusterings using evidence accumulation, *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(6) (2005) 835−850.
10. A. Guénoche, Consensus of partitions: A constructive approach, *Adv. Data Anal. Classification* **5**(3) (2011) 215−229.
11. A. K. Jain, M. N. Murty and P. J. Flynn, Data clustering: A Review, ACM Computing Surveys (CSUR), **31**(3) (1999) 264−323.
12. A. K. Jain, Data clustering: 50 years beyond k-means, *Pattern Recog. Lett.* **31**(8) (2009) 651−666.
13. H. Li, K. Zhang and T. Jiang, Minimum entropy clustering and applications to gene expression analysis, *Proc. IEEE Conf. Computational Systems Bioinformatics* (2004), pp. 142−151.
14. M. H. Louati, S. Benabdallah, F. Lebdi and D. Milutin, Application of a genetic algorithm for the optimization of a complex reservoir system in Tunisia, *Water Resources Management, Earth Environ. Sci.* **25**(10) (2011) 2387−2404.
15. H. Luo, F. Jing and X. Xie, Combining multiple clusterings using information theory based genetic algorithm, *IEEE Int. Conf. Computational Intelligence and Security* (2006), pp. 84−89.
16. B. Minaei-Bidgoli, H. Parvin, H. Alinejad-Rokny, H. Alizadeh and W. F. Punch, Effects of resampling method and adaptation on clustering ensemble efficacy, *Artif. Intell. Rev.* (2011) 1−22. Available at http://link.springer.com/article/10.1007%2Fs10462-011-9295-x?LI=true.
17. P. Y. Mok, H. Q. Huang, Y. L. Kwok and J. S. Au, A robust adaptive clustering analysis method for automatic identification of clusters, *Pattern Recog.* **45**(8) (2012) 3017−3033.
18. J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Industrial Appl. Math.* **5**(1) (1957) 32−38.

19. C. B. D. J. Newman, S. Hettich and C. Merz, UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/MLSummary.html. (1998).
20. H. Parvin, B. Minaei-Bidgoli and H. Alizadeh, A new clustering algorithm with the convergence proof, *KES 2011*, *Part I, LNAI* **6881** (2011) 21–31.
21. P. R. Rao and J. P. P. Da Costa, A performance study of a consensus clustering algorithm and properties of partition graph, *Proc. of ICCIC 2010* (2010), pp 1–5.
22. V. Roth, T. Lange, M. Braun and J. Buhmann, A resampling approach to cluster validation, *Intl. Conf. Computational Statistics, COMPSTAT* (2002), pp. 123–128.
23. W. Sheng and X. Liu, A genetic k-medoids clustering algorithm, *J. Heuristics* **12** (2006) 447–466.
24. W. Sheng, A. Tucker and X. Liu, A niching genetic k-means algorithm and its applications to gene expression data, Soft Computing — A Fusion of Foundations, *Methodol. Appl.* **14**(1) (2010) 9–19.
25. V. Singh, L. Mukherjee, J. Peng and J. Xu, Ensemble clustering using semidefinite programming with applications, *Mach. Learn* **79** (2010) 177–200.
26. A. Strehl and J. Ghosh, Cluster ensembles — a knowledge reuse framework for combining multiple partitions, *J. Mach. Learn. Res.* **3** (2002) 583–617.
27. S. Vega-Ponz and J. Ruiz-Shulcloper, A survey of clusering ensemble algorithms, *Int. J. Pattern Recog. Artif. Intell.* **25**(3) (2011) 337–372.
28. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd edn. (Academic Press, CA, USA, 2006).
29. Y. G. Wan, R. F. Liu and H. J. Li, The inversion of 3-D crustal structure and hypocenter location in the Beijing-Tianjin-Tangshan-Zhangjiakou area by genetic algorithm, *Acta Seismologica Sinica* **10**(6) (1997) 769–781.
30. P. Wang T. Weise and R. Chiong, Novel evolutionary algorithms for supervised classification problems: An experimental study, *Evolutionary Intell.* **4**(1) (2011) 3–16.
31. Z. Xu and X. Cai, Minimizing group discordance optimization model for deriving expert weights, *Group Decision Negotiation* **21**(6) (2011) 863–875.
32. H. S. Yoon, S. Y. Ahn, S. H. Lee, S. B. Cho and J. H. Kim, Heterogeneous clustering ensemble method for combining different cluster results, *BioDM06*, *LNBI* **3916** (2006a), pp. 82–92.
33. H. S. Yoon, S. H. Lee, S. B. Cho and J. H. Kim, A novel framework for discovering robust cluster results, *DS06*, *LNAI* **4265** (2006b), pp. 373–377.
34. K. Zhang, B. Xu, L. Tang and H. Shi, Modeling of binocular vision system for 3D reconstruction with improved genetic algorithms, *Int. J. Adv. Manuf. Technol.* **29**(7) (2006) 722–728.



Hosein Alizadeh received his B.Sc. degree in computer engineering from Payam Nour University, Babol, in 2006. He then obtained his M.Sc. degree in Computer Engineering, Artificial Intelligence from Iran University of Science and Technology (IUST), Tehran, in 2009. Since 2009, he has been a Ph.D. student in Computer Engineering, Artificial Intelligence in IUST. In both M.Sc. and Ph.D., Hosein has worked under the supervision of Dr. Behrouz Minaei-Bidgoli. He has published several papers in various journals and book chapters. His research interests include cluster ensemble, community detection, classifier fusion, and optimization methods.

Behrouz Minaei-Bidgoli obtained his Ph.D. from the Computer Science & Engineering Department of Michigan State University, USA, in the field of Data Mining and Web-Based Educational Systems. Now, he is an Assistant Professor in the Department of Computer Engineering, Iran University of Science & Technology (IUST). He is also the leader of the Data and Text Mining Research Group in Noor Computer Research Center, Qom, Iran, developing large scale NLP and Text Mining projects for Persian/Arabic language.



Hamid Parvin was born in Nourabad Mamasani. He received his B.Sc. degree from Shahid Chamran University in Software Engineering in 2006. He received his M.Sc. degree in Artificial Intelligence from Iran University of Science and Technology, Tehran, Iran under the supervision of Behrouz Minaei-Bidgoli in 2008. He is a Ph.D. student of Artificial Intelligence in Iran University of Science and Technology, Tehran, Iran under supervision of Behrouz Minaei-Bidgoli. He has published several journal papers among which 5 of them are SCIE indexed. He has also published many papers in various book chapters. He is now a faculty member in the Islamic Azad University, Nourabad Mamasani Branch. His research interests are ensemble-based learning, evolutionary learning and data mining.