Expert Systems with Applications 39 (2012) 9695–9703

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A new grouping genetic algorithm for clustering problems

L.E. Agustín-Blas [a], S. Salcedo-Sanz [a,*], S. Jiménez-Fernández [a], L. Carro-Calvo [a], J. Del Ser [b], J.A. Portilla-Figueras [a]

[a] Department of Signal Theory and Communications, Universidad de Alcalá, Madrid, Spain
[b] Tecnalia Research & Innovation, Bizkaia, Spain

ARTICLE INFO

Keywords:
Grouping genetic algorithms
Clustering problems
Hybrid algorithms

ABSTRACT

In this paper we present a novel grouping genetic algorithm for clustering problems. Though there have been different approaches that have analyzed the performance of several genetic and evolutionary algorithms in clustering, the grouping-based approach has not been, to our knowledge, tested in this problem yet. In this paper we fully describe the grouping genetic algorithm for clustering, starting with the proposed encoding, different modifications of crossover and mutation operators, and also the description of a local search and an island model included in the algorithm, to improve the algorithm's performance in the problem. We test the proposed grouping genetic algorithm in several experiments in synthetic and real data from public repositories, and compare its results with that of classical clustering approaches, such as *K*-means and DBSCAN algorithms, obtaining excellent results that confirm the goodness of the proposed grouping-based methodology.

1. Introduction

Clustering is an important subgroup of unsupervised learning techniques consisting in grouping data objects into disjoint groups of *clusters* (Jain, Murty, & Flynn, 1999; Liao, 2005; Lingras & Huang, 2005; Xu & Wunsch, 2005). The classification into clusters is usually defined in such a way that objects in the same cluster are similar in terms of a given measure, and different from the objects in the other clusters, with respect to the same measure.

Clustering has been applied to a wide variety of problems in many different fields such as pattern recognition, bio-engineering, image quantization, renewable energy prediction, etc. (Gomez-Muñoz & Porta-Gándara, 2002; Mitra & Banka, 2006; Scheunders, 1997). In Chang, Zhang, and Zheng (2009), four major types of clustering algorithms are identified: first, algorithms based on the idea that neighbor data should share the same cluster. Classical clustering algorithms such as density-based approaches (Ester, Kriegel, & Sander, 1996) belong to this first group. As pointed out in Chang et al. (2009), this kind of algorithms are robust to detect clusters of any shape, but they fail to locate clusters when there is small spatial separation between clusters. The second set of clustering algorithms is formed by those approaches which consider intra-clusters variation (intra-clusters points or centroids) to form the final solution. This category of algorithms includes the well known *K*-means (Chang et al., 2009; Kanungo et al., 2002; Likas, Vlassis, & Verbeek, 2003), and other approaches such as model-based clustering (Mclachlan & Basford, 1988). Following (Chang et al., 2009), the third category includes a simultaneous row-column clustering known as bi-clustering algorithms (Madeira & Oliveira, 2004). Finally, the fourth group of clustering algorithms includes approaches that optimize different characteristics of the data set. This group includes the multi-objective clustering algorithms (Dehuri, Ghosh, & Mall, 2006; Mitra & Banka, 2006) and also clustering ensembles approaches (Hong, Kwong, Chang, & Ren, 2008).

In the last few years, evolutionary computing algorithms (EAs) have been widely applied to clustering problems, due to their capacity to be applied to very different problems with very few changes, and also because these algorithms are able to manage constraints in an efficient way. Some recent work in the direct application of EAs to clustering problems can be found in Hruschka and Ebecken (2003), where a genetic algorithm is applied to a clustering problem. In this work, a simple encoding scheme with constant-length individuals is used. The objective function maximizes both the homogeneity within each cluster and the heterogeneity among clusters, and this approach also finds the right number of clusters according to an external measure (the average silhouette width criterion). In Chang, Zhao, Zheng, and Zhang (2012) an evolutionary algorithm for clustering is described, where the objective function is constructed through a message-based similarity function, which simulates messaging between objects and optimal centroids of the clusters. The performance of the approach is shown in different synthetic examples and also in real datasets from UCI repository (Asuncion & Newman, 2007). In Liu, Wu, and Shen (2011) a real-encoding

* Corresponding author. Address: Department of Signal Theory and Communications, Universidad de Alcalá, 28871 Alcalá, Madrid, Spain. Tel.: +34 91 885 6731; fax: +34 91 885 6699.
E-mail address: sancho.salcedo@uah.es (S. Salcedo-Sanz).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.02.149

evolutionary algorithm with a special division-absorption mutation operator and a fitness function based on the Davis–Bouldin index (Davies & Bouldin, 1997) is presented. The authors show the good performance of the approach in different synthetic problems and in the well-known Breast Cancer dataset. In Deng, He, and Xu (2010) a genetic clustering algorithm is presented, which is based on a measure of mutual information between objects in the different clusters. EAs have also been applied to improve the *K*-means approach in the literature, for example in Krishna and Murty (1999), obtaining the *genetic K-means algorithm*, which is known to be more effective that the *K*-means in hard clustering problems. Other works dealing with evolutionary *K*-means algorithms are (Xiao, Yan, Zhang, & Tang, 2010; Zahraie & Roozbahani, 2011). Similar approaches have been used in different applications, such as color quantization (Scheunders, 1997) or bio-engineering (Chang et al., 2009). EAs have also been applied to other clustering problems (Murthy & Chowdhury, 1996; Tseng & Yang, 2001), and also recently to bi-clustering problems (Divina & Aguilar-Ruiz, 2006; Mitra & Banka, 2006). There are different types of evolutionary approaches that have been studied in clustering problems, such as evolutionary programing (Sarkar, Yegnanarayana, & Khemani, 1997), particle swarm optimization (Cura, 2012; Das, Abraham, & Konar, 2008; Yang, Sun, & Zhang, 2009), Ant Colony algorithms (Jiang, Yi, Li, Yang, & Hu, 2010) or Artificial Bee Colony algorithms (Zhang, Ouyang, & Ning, 2010).

In spite of this massive application of evolutionary techniques, to our knowledge, grouping genetic algorithms have not been tested in clustering problems yet. Intuitively, the grouping genetic algorithm should perform really well in clustering, since its structure is adaptive to manage groups of items. The problem is to find the appropriate evolution operators and fitness function to obtain a robust algorithm for clustering, which offers good performance in different scenarios. In this paper, we present a contribution to clarify the performance of the grouping genetic algorithm in clustering problem. In fact, we describe a new grouping genetic algorithm for clustering that includes the traditional encoding structure of grouping algorithms, and also novel crossover and mutation operators. We study the performance of the algorithm when using different fitness functions, and an island model in order to parallelize the algorithm's evolution. Experiments in synthetic and real data from public repositories complete our study on the grouping genetic algorithm for clustering problems.

The rest of the paper has been structured as follows: next section summarizes some important definitions on clustering and clustering measures and distances, needed to fully understand the rest of the paper. Section 3 presents the grouping genetic algorithm proposed in this paper for clustering problems. Section 4 contains the experimental part of the paper, where the performance of the proposed grouping approach is evaluated. Section 5 closes the paper by giving some final remarks and conclusions.

2. Clustering problems: definitions and evaluation measures

Mathematically, a clustering problem is composed of the following elements: let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of N vectors in a given feature space $\mathcal{S}$. The objective of a clustering problem is to find an optimal partition of $X, U^* = \{C_1^*, \ldots, C_k^*\}, C_i \bigcap C_j = \emptyset$, where C_i^* stands for the ith cluster of partition U^*, in such a way that the patterns belonging to the same cluster are similar whereas patterns belonging to different clusters are as different as possible, in terms of a given measure function $m(U)$.

2.1. Measure of distances in clustering problems

The measure of distances is one of the key elements when dealing with clustering problems, since usually the similarity between two different vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is related to a measure of distance in the feature space $\mathcal{S}$. In traditional clustering problems, the most commonly used distance is given by a norm defined by a symmetric and positive matrix A:

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_A = (\mathbf{x}_i - \mathbf{x}_j) \cdot A \cdot (\mathbf{x}_i - \mathbf{x}_j)^T, \quad (1)$$

where T stands for the transpose operation. Note that matrix A determines the shape and size of the set of vectors sited at a distance of a given vector under study $\mathbf{x}_i$. The most popular form of a norm A distance is the simplest case when $A = \mathbb{I}$, i.e., the identity matrix. In this case, the distance generated is the Euclidean distance, defined as:

$$d_E^2(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2 = (\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_i - \mathbf{x}_j)^T, \quad (2)$$

There are alternative distances for cases in which not all the clusters are ellipsoids with the same orientation and size. Specifically, the Mahalanobis distance takes care of these cases, by defining the following distance:

$$d_M^2(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_{\Sigma^{-1}} = (\mathbf{x}_i - \mathbf{x}_j) \cdot \Sigma^{-1} \cdot (\mathbf{x}_i - \mathbf{x}_j)^T \quad (3)$$

where Σ stands for the covariance matrix. Note that Mahalanobis distance considers correlation between the different features of the vectors involved in the distance. This way it is easy to consider different associations between variables in order to define orientation and size of the different clusters.

2.2. Clustering evaluation

Validation or evaluation of the resulting clustering allows analyzing the result in terms of objective measures (Halkidi, Batistakis, & Vazirgiannis, 2001). Depending on the information available, we can evaluate the clustering result in terms of unsupervised or supervised measures:

2.2.1. Unsupervised measures

This type of evaluation tries to determine the quality of a given obtained partition of the data without any external information available. This is why this unsupervised measure are sometimes called as *internal measures*. We describe some of the most useful ones:

- Sum of quadratic errors (SSE): This is probably the most straightforward and popular evaluation distance in the literature. It only considers cohesion of clusters in order to evaluate the quality of a given partition data.

$$SSE(U) = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(\mathbf{x}, \boldsymbol{\mu}_i) \quad (4)$$

where k stands for the number of clusters in the partition and $d(\mathbf{x}, \boldsymbol{\mu}_i)$ is the distance from observed vector $\mathbf{x}$ to the centroid of the cluster i, represented by the symbol $\boldsymbol{\mu}_i$.

- Davis–Bouldin Index (DB): The idea of the Davis–Boudin index (Davies & Bouldin, 1997) is to minimize the intra-cluster distances, and at the same time, maximize the distances among the different clusters.

$$DB(U) = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left\{ \frac{\sum_{x \in C_i} d^2(\mathbf{x}, \boldsymbol{\mu}_i) + \sum_{x \in C_j} d^2(\mathbf{x}, \boldsymbol{\mu}_j)}{d^2(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)} \right\} \quad (5)$$

note that, due to the definition, small values of the DB index correspond to compact and well separated clusters. Note that this index does not present a monotonic behavior with k, so the DB index allows also validating the optimal number of clusters for a given data set.

- Silhouette coefficient(S): This is a measure that has been quite often used in clustering problems since it allows evaluating the quality of a particular solution, and also the quality of each of the clusters that conform that solution, even more, it allows evaluating a given assignment for each particular observation vector $\mathbf{x}_i$. Thus, the silhouette coefficient is defined for the jth observation ($\mathbf{x}_j$),

$$s_j = \frac{a_j - b_j}{\max(a_j, b_j)}, \tag{6}$$

where the parameters a_j and b_j stand for the average distance between observation $\mathbf{x}_j$ and the other vectors in its and different clusters, respectively. Note that for b_j, the average distance is taken as the minimum distance obtained for all clusters different that the one assigned to $\mathbf{x}_j$. We can define the silhouette coefficient for a given cluster C_j, in the following way:

$$S_j = \sum_{\mathbf{x}_j \in C_j} s_j, \tag{7}$$

and thus, the silhouette coefficient for a given partition U of the data is:

$$S(U) = \frac{1}{k} \sum_{j=1}^{k} S_j. \tag{8}$$

Note that with this definition, $S(U)$ is in the interval $[-1,1]$, and the objective for a good partition is to maximize S.

2.2.2. Supervised measures

Sometimes there are available external (known) results for the clustering problem, that can help to evaluate the quality of a given algorithm's result. Supervised measures are sometimes called external measures. Let us define two of the most used in clustering literature.

- Rand index (R): Rand index (Rand, 1971) calculates the similarity between the obtained partition and the known optimal solution, i.e it is a measure of the percentage of correct decisions taken by the algorithm.

$$R(U) = \frac{TP + TN}{TP + FP + TN + FN} \tag{9}$$

where TP and FP are the number of correct and incorrect assignments, respectively, when the decision consists of assigning two elements to the same cluster, and TN and FN are the number of correct and incorrect assignments, respectively, when the decision consists of assigning two elements to different clusters. Note that R is in the interval $[0,1]$, and values of R closer to 1 indicate a better quality of the solution tested.
- Jaccard index (J): Jaccard index is quite similar to Rand index, but this measure does not consider the number of correct assignments when two elements are assigned to different clusters.

$$J(U) = \frac{TP}{TP + FP + FN} \tag{10}$$

3. Proposed grouping genetic algorithm

The grouping genetic algorithm (GGA) is a class of evolutionary algorithm especially modified to tackle grouping problems, i.e., problems in which a number of items must be assigned to a set of predefined groups. It was first proposed by Falkenauer (1992, 1998), who realized that traditional genetic algorithms had difficulties when they were applied to grouping problems. Thus, in the GGA, the encoding, crossover and mutation operators of traditional GAs are modified to obtain a compact algorithm, with a high performance in grouping-based problems.

In spite of its good performance in very different applications (Agustín-Blas, Salcedo-Sanz, Ortiz-García, Portilla-Figueras, & Pérez-Bellido, 2009; Agustín-Blas, Salcedo-Sanz, Vidales, Urueta, & Portilla-Figueras, 2011; Brown & Sumichrast, 2003, 2004, 2005; Hung, Sumichrast, & Brown, 2003; James, Vroblefski, & Nottingham, 2007; James, Brown, & Keeling, 2007; Kreng & Lee, 2004; De Lit, Falkenauer, & Delchambre, 2000), to our knowledge GGAs have not been tested in clustering problems. In this paper we develop an efficient GGA for clustering problems in which we include different improvements over the standard GGA version of Falkenauer. We also discuss the performance of the proposed algorithm including different objective functions, characterized by different distance definitions, as shown in Section 2. In the next subsections, we show the main characteristics of the hybrid GGA that we propose. Special attention will be paid to the encoding, genetic operators, implementation of a local search to improve the quality of solutions, and finally, the parallelization of the algorithm by including an island model of evolution.

3.1. Problem encoding

The proposed GGA for clustering follows the classical grouping encoding initially proposed by Falkenauer, i.e. it is a variable-length genetic algorithm. The encoding is carried out by separating each individual in the algorithm into two parts: $\mathbf{c} = [\mathbf{l}|\mathbf{g}]$, the first part is the *element* section, whereas the second part is called the *group* section of the individual. As an example, following our notation, in a solution for a clustering problem with N elements (observations) and k clusters, the individual will have the following aspect:

$$l_1, l_2, \ldots, l_N | g_1, g_2, \ldots, g_k$$

Note that l_j represents the cluster to which jth observation is assigned, whereas group section keeps a list of tags associated to each of the clusters of the solution. In a formal way:

$$l_j = g_i \Longleftrightarrow \mathbf{x}_j \in C_i. \tag{11}$$

Note also that the length of the element section is fixed for a given problem (equals N), but the group section's length is not fixed, it varies from one individual to another. Thus the GGA does not need as input parameter the number of clusters, but it searches for the best k in terms of the objective function.

As an example to fully clarify the GGA encoding in clustering, let us suppose the following individual:

1 3 2 1 4 1 1 2 3 2 1 3 4 2 1 | 1 2 3 4

This individual represents a solution with 4 clusters, and the following partition of the input data: $\{x_1,x_4,x_6,x_7,x_{11},x_{15}\}$, $\{x_3,x_8,x_{10},x_{14}\}$, $\{x_2,x_9,x_{12}\}$ and $\{x_5,x_{13}\}$.

3.2. Selection operator

In this paper we use a rank-based wheel selection mechanism, similar to the one described in James et al. (2007). First, the individuals are sorted in a list based on their quality. The position of the individuals in the list is called *rank of the individual*, and denoted R_i, $i = 1,\ldots,\xi$, with ξ number of individuals in the population of the GGA. We consider a rank in which the best individual x is assigned $R_x = \xi$, the second best y, $R_y = \xi - 1$, and so on. A *fitness* value associated to each individual is then defined, as follows:

$$f_i = \frac{2 \cdot R_i}{\xi \cdot (\xi + 1)} \tag{12}$$

Note that these values are normalized between 0 and 1, depending on the position of the individual in the ranking list. It is important to note that this rank-based selection mechanism is *static*, in the sense that probabilities of survival (given by f_i) do not depend on the gen-

eration, but on the position of the individual in the list. As a small example, consider a population formed by 5 individuals, in which individual 1 is the best quality one ($R_1 = 5$), individual 2 the second best ($R_2 = 4$), and so on. In this case, the fitness associated to the individuals are $\{0.33, 0.26, 0.2, 0.13, 0.06\}$, and the associated intervals for the roulette wheel are $\{0 - 0.33, 0.34 - 0.6, 0.61 - 0.8, 0.81 - 0.93, 0.94 - 1\}$.

The process carried out in our algorithm consists of selecting the parents for crossover using this selection mechanism. This process is performed with replacement, i.e., a given individual can be selected several times as one of the parents, however, individuals in the crossover operator must be different.

3.3. Crossover operator

The crossover operator implemented in the grouping genetic algorithm used in this paper is a modified version of the one initially proposed by Falkenauer (1992) to adapt it to the clustering problem. The process follows a two parents one offspring schema, with following steps:

a. First, two individuals are randomly selected, and two crossing points are chosen in their group part.
b. Insert the elements belonging to the selected groups of the first individual into the offspring.
c. Insert the elements belonging to the selected groups of the second individual into the offspring, if they have not been assigned by the first individual.
d. Randomly complete the elements not yet assigned with elements from the current groups.
e. Remove empty clusters, if any.
f. Modify the labels of the current groups in the offspring in order to numerate them from 1 to k.

Fig. 1 shows an example of the crossover procedure implemented in this work. The probability of crossover must be high in the first stages of the algorithm, and moderate in the last ones in order to properly explore the search space. Thus, we have implemented an adaptive crossover probability, defined in the following way:

$$P_c(j) = P_{ci} + \frac{j}{TG}(P_{ci} - P_{cf}) \tag{13}$$

where $P_c(j)$ is the crossover probability used in a given generation j, TG stands for the total number of generations of the algorithm, and P_{ci} and P_{cf} are the initial and final values of probability considered, respectively.

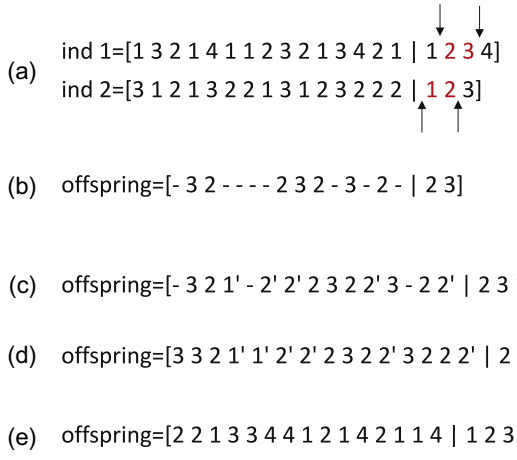

Fig. 1. Example of the crossover operator implemented in the proposed grouping genetic algorithm for clustering problems.

3.4. Mutation operator

Mutation operator includes small modifications in each individual of the population with a low probability, in order to explore new regions of the search space and also scape from local optima when the algorithm is near convergence. In this case, we have implemented two different mutation operators adapted to the clustering problems:

- Mutation by cluster splitting: it consists of splitting a selected cluster into two different ones. The samples belonging to the original cluster are assigned to the new clusters with equal probability. Note that one of the new generated clusters will keep its label in the group section of the individual, whereas the other will be assigned a new label ($k + 1$). The selection to the initial cluster to be split is carried out depending on the clusters' size, with more probability of split given to larger clusters. As an example, we illustrate an application of this operator in the individual used to illustrate the crossover operator (Fig. 1), in the case when the initial cluster chosen to be split is cluster 1:

  2 2 1 3 3 4 4 5 2 1 4 2 5 1 4 | 1 2 3 4 5

- Mutation by clusters merging: it consists of merging two existing clusters, randomly selected, into just one. As in mutation by cluster splitting, the probability of choosing the clusters depends on their size. In order to illustrate this mutation, again an example in the individual used to illustrate the crossover operator (Fig. 1) is given. In this case, let us suppose that the selected clusters are clusters 2 and 4:

  2 2 1 3 3 2 2 1 2 1 2 2 1 1 2 | 1 2 3

Similarly to the crossover case, we also consider an adaptive version of the probability of applying the mutation operators described above. Note that we apply the two mutation operators in a serial fashion (one after the other), with independent probabilities of application. In this case, probability of mutation is smaller in the first generations of the algorithm and larger in the last ones, in order to have more opportunities to scape from local minimums in the last stages of the evolution:

$$P_m(j) = P_{mi} + \frac{j}{TG}(P_{mf} - P_{mi}) \tag{14}$$

where $P_m(j)$ is the probability of mutation used in a given generation j, TG stands for the total number of generations of the algorithm, and P_{mf} and P_{mi} are the final and initial values of probability considered, respectively.

3.5. Replacement and elitism

In the proposed GGA, the population at a given generation $j + 1$ is obtained by replacement of the individuals in the population at generation j, through the application of the selection, crossover, and mutation operators described above. An elitist schema is also applied, the best individual in generation j is automatically passed onto the population of generation $j + 1$, ensuring that the best solution encountered so far in the evolution is always kept by the algorithm.

3.6. Local search

We use a local search procedure to try to find local optimums in a close neighborhood of an individual. The local search proposed is based on slight modifications of the current individual, as far as they produce an increase of the associated objective function. In this case, the implemented local search works over the element section of the individuals. For each observation, this operator determines the objective function variation obtained when the observation is

assigned to the other clusters in the solution. Finally, we keep the assignment with the largest objective function. Since this is a quite time-consuming operation it is applied to a given individual with a small probability, p_b, that is modified between an initial and final value in the algorithm in the same way that the crossover probability.

3.7. An island model to improve the algorithm's performance

In order to improve the performance of the proposed GGA, an island model is considered to parallelize it. $\mathcal{S}$ sub-populations (islands) are taken into account, in such a way that the evolution in each island is independent, but the migration of good individuals between islands is allowed. We consider an elitist migration model, in which only the best individual in each island migrates, and substitutes a randomly chosen individual in one of the other islands. There is a probability of migration p_e predefined in the algorithm. The migration process is as follows:

1. Choose the best individual in each island.
2. Randomly choose the island toward each individual will migrate.
3. Randomly choose an individual in the destiny island and change it by the migrating individual.

4. Experiments and results

The experimental part of the paper has been structured in two subsections. First, we briefly describe two classical clustering algorithms against we will compare the proposed GGA, i.e, the *K*-means and DBSCAN algorithms. Second, we show different experiments and results obtained in synthetic and real data problems obtained from public repositories.

4.1. Clustering algorithms for comparison

In this section we revise two of the most used classical approaches for clustering problems, that we will use for comparison purposes, i.e, the *K*-means algorithm and the DBSCAN approach. The idea is to situate the behavior of the proposed GGA in terms of these two classical approaches, which have obtained very good results in many real clustering problems and applications.

4.1.1. K-means

K-means algorithm was proposed by MacQueen in McQueen (1968), it has turned into the most popular and compared clustering algorithm. The *K*-means approach requires as input parameter the number of clusters *k*, and then operates in the following way, in order to obtain a partition of the data into *k* clusters:

Algorithm 1. *K*-means algorithm

Require: Initial number of clusters *k*.
Ensure: A partition of the data *U*.

1: Initialize the *k* centroids $\boldsymbol{\mu}_1^{(1)}, \ldots, \boldsymbol{\mu}_k^{(1)}$.

2: Assign each observation to the closest centroid:

$$C_i^{(t)} = \left\{ \mathbf{x} : d\left(\mathbf{x}, \boldsymbol{\mu}_i^{(t)}\right) \leqslant d\left(\mathbf{x}, \boldsymbol{\mu}_j^{(t)}\right), j = 1, \ldots, k \right\}$$

3: Update the centroids from the observations assigned to each cluster:

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x \in C_i^{(t)}} \mathbf{x}$$

4: Repeat previous steps until a convergence criterium is fulfilled.

The *K*-means algorithm is simple and easy to apply in a large variety of problems. Note, however, that the *K*-means will obtain poor results in problems in which clusters have different sizes and densities. It also performs poorly in data with many outliers. In addition, the *K*-means algorithm has a major dependency of the initialization of the centroids, what can make that the algorithm ends up in a local minimum. In our experiments, we use the well-known *elbow method* to automatically set the number of centroids in the *K*-means algorithm.

4.1.2. DBSCAN

An alternative clustering algorithm which is also quite common in the literature is the so called Density Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996). DBSCAN requires two parameters to initialize, i.e, a threshold ϵ, and the minimum number of points required to form a cluster (*minpts*). The clustering process is then based on the classification of the observations as *core points*, *border points* and *noise points*, and on the use of density relations between points (directly density-reachable, density-reachable, density-connected) to form the clusters. A point $\mathbf{x}_i$ is directly density-reachable from another point $\mathbf{x}_j$ if the distance

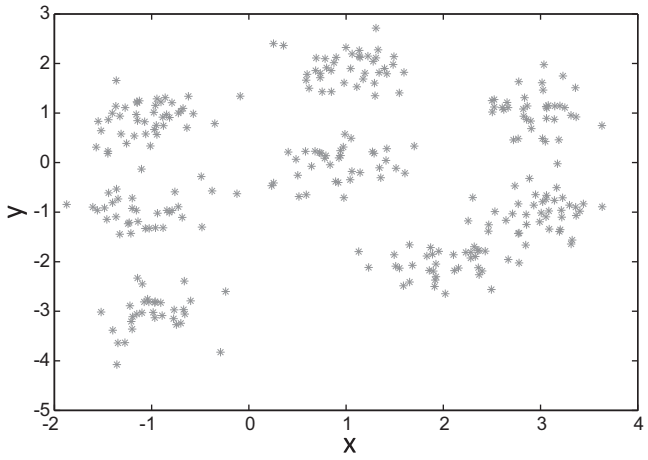

Fig. 2. Data of the first synthetic clustering example: spheric data.

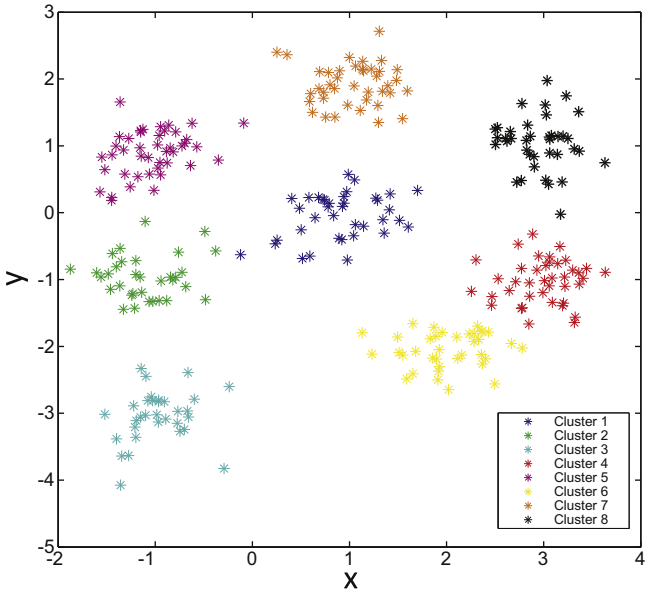

Fig. 3. Best clustering obtained with the proposed GGA with DB index in the first synthetic clustering problem considered.

Table 1
Comparison of the results obtained by the proposed grouping genetic algorithm with DB and S index, DBSCAN and *K*-means algorithms in the first synthetic clustering problem considered.

Algorithm	# Clusters	Rand index
GGA (DB index)	8	**0.9814**
GGA (S index)	7	0.9578
DBSCAN	7	0.9555
K-means	9	0.9493

Table 2
Comparison of the results obtained by the proposed grouping genetic algorithm with DB and S index, DBSCAN and *K*-means algorithms in the second synthetic clustering problem considered.

Algorithm	# Clusters	Rand index
GGA (DB index)	3	0.9177
GGA (S index)	3	**0.9511**
DBSCAN	1	0.3760
K-means	4	0.8755

between them is less than the threshold ϵ, and $\mathbf{x}_j$ is a core point. A point $\mathbf{x}_i$ is density-reachable from $\mathbf{x}_j$ if there exists a sequence of points among them in such a way that each point is directly density-reachable from the previous one.

The algorithm starts by selecting an arbitrary point. If the selected point is a core one, i.e. there are *minpts* points within a distance ϵ, then a cluster is started, and all the density-reachable points from that core are assigned to be in its cluster. The process continues until all the points in the dataset have been processed. The points that stay out of the formed clusters are called noise points (not assigned), whereas the points that are not noise nor core, are the border points.

Algorithm 2.DBSCAN algorithm

Require: A threshold ϵ for defining density-reachability.
Require: The minimum number of points *minpts* needed to start a cluster.
Ensure: A partition of the data *U*.
1: Set a counter $t = 1$.
2: Select an arbitrary point among the data.
3: **if** the selected point is a core point **then**
4: start the cluster *t*.
5: **end if**
6: Assign all the density-reachable points to cluster *t*.
7: $t = t + 1$
8: Process next points to locate other core points.
9: Label remaining points as noise.

Note that DBSCAN does not need to know beforehand the number of clusters for being run. It needs, however, two input parameters ϵ and *minpts*. DBSCAN is able to recognize clusters with different shapes and sizes, and takes into account the notion of noise, in such a way that outlier samples do not influence the algorithm's performance. Note, however, that it will have a poor performance in clustering problems with significant differences in density areas.

4.2. Results

This section presents the results obtained with the proposed GGA and the *K*-means and DBSCAN algorithms for comparison. We have structured the experiments in experiments over synthetic data, and experiments over repository (real) data, from public repositories.

4.2.1. Synthetic data: spheric clusters

In this first experiment, we test the performance of the proposed GGA in a 2-dimension clustering problem, defined by 300 objects, randomly generated using a Gaussian distribution from 8 equiprobable classes, with means: $\mu_1 = (-1,0)$, $\mu_2 = (-1,-1)$, $\mu_3 = (-1,-3)$, $\mu_4 = (3,-1)$, $\mu_5 = (-1,1)$, $\mu_6 = (2,-2)$, $\mu_7 = (1,2)$, $\mu_8 = (3,1)$, and the following covariance matrices:

$$\Sigma_1 = \cdots = \Sigma_8 = \begin{bmatrix} 0.35^2 & 0 \\ 0 & 0.35^2 \end{bmatrix} \tag{15}$$

Note that this results in a problem of spheric clusters. Fig. 2 shows the observations randomly generated following the statistical distribution considered.

Table 1 summarizes the results obtained in this instance by different algorithm. We show the results of the proposed GGA with the Davies–Boulding (DB) index and the Silhouette(S) coefficient as fitness functions. Note that the results are given in terms of

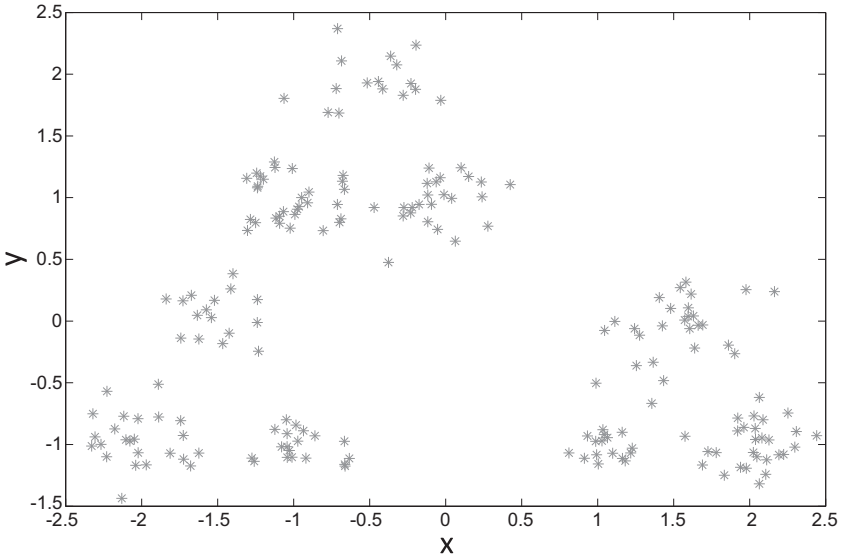

Fig. 4. Data of the second synthetic clustering example: structured data.

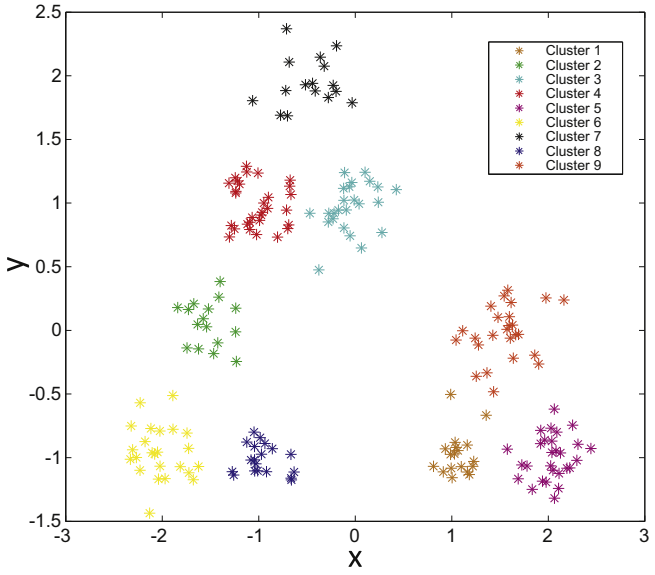

Fig. 5. Best clustering obtained with the proposed GGA with S index in the second synthetic clustering problem considered.

the Rand index, obtained considering that the best partition of the data is the one corresponding to the original classes. Note also that the GGA solutions improve the results of the DBSCAN and *K*-means approaches. Fig. 3 shows the result of the GGA solution obtained with the DB index.

4.2.2. Synthetic data: structured clusters

We test following the performance of the proposed GGA in a different 2-dimension clustering problem, defined by 400 objects, randomly generated using a Gaussian distribution from 3 classes with probability, $p_1 = 0.5$, $p_2 = 0.33$ and $p_3 = 0.17$. The means of each classes are: $\mu_1 = (0,2)$, $\mu_1 = (-1,-1)$, $\mu_2 = (2,-1)$ and $\mu_3 = (0,2)$, and they have the following covariance matrices:

$$\Sigma_1 = \begin{bmatrix} 1^2 & 0 \\ 0 & 0.8^2 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.6^2 & 0 \\ 0 & 0.4^2 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 0.3^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}, \quad (16)$$

It is easy to see that, in this case, the classes are not spherical, and have different distributions. Fig. 4 displays the observations generated for this instance.

Table 3
Comparison of the results obtained by the proposed grouping genetic algorithm with DB and S index, DBSCAN and *K*-means algorithms in the third synthetic clustering problem considered.

Algorithm	# Clusters	Rand index
GGA (DB index)	9	**1.0000**
GGA (S index)	9	0.9936
DBSCAN	8	0.9545
K-means	3	0.7809

Table 2 shows the results obtained by the proposed GGA, with DB and S indexes, and the results obtained by the DBSCAN and *K*-means algorithms for comparison. In this case the best results is obtained by the GGA with S index as fitness function, and it is also interesting the poor result obtained by the DBSCAN approach. Fig. 5 shows the best result obtained by the GGA with S index as fitness function.

4.2.3. Synthetic data: unbalanced clusters

In this final synthetic experiment, we test the performance of the proposed GGA in a 2-dimension clustering problem, defined by 200 objects, randomly generated using a Gaussian distribution from 9 equiprobable classes, with means: $\mu_1 = (1,-1)$, $\mu_2 = (-1.5,0)$, $\mu_3 = (0,1)$, $\mu_4 = (-1,1)$, $\mu_5 = (2,-1)$, $\mu_6 = (-2,-1)$, $\mu_7 = (-0.5,2)$, $\mu_8 = (-1,-1)$, $\mu_9 = (1.5,0)$ and the following covariance matrices:

$$\Sigma_1 = \cdots = \Sigma_8 = \begin{bmatrix} 0.2^2 & 0 \\ 0 & 0.2^2 \end{bmatrix} \quad (17)$$

Fig. 6 shows the observations randomly generated following the statistical distribution considered. In this case, there are three groups of clusters that form different clusters structures. The idea is to check out that the tested algorithms are able to correctly separate all the cases.

Table 3 summarizes the results obtained in this instance by the different algorithms considered. The proposed GGA with the DB index obtains a perfect reconstruction of the clusters (Rand index equals 1). The GGA with the S coefficient as fitness function also obtains an excellent clustering of the data. In this case, the DBSCAN algorithm obtains good results, and the *K*-means algorithm is the one which fails to obtain a good result since it is not able to cor-

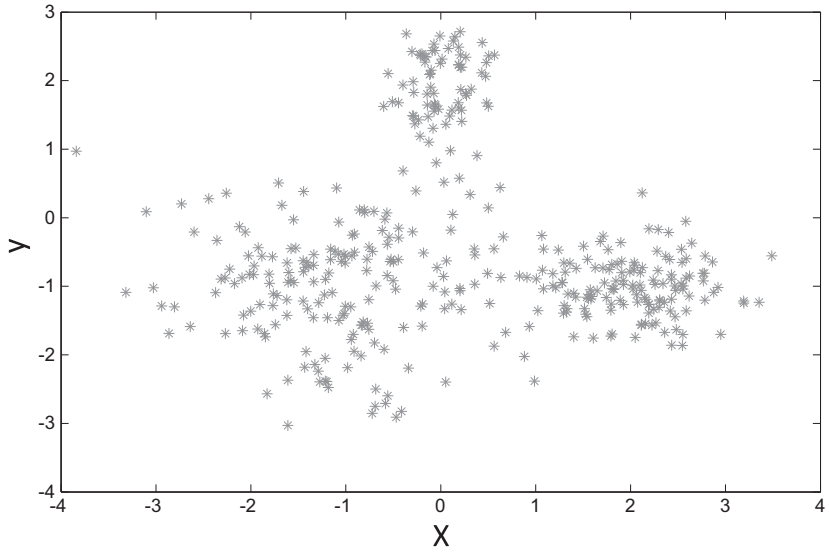

Fig. 6. Data of the third synthetic clustering example: unbalanced data.

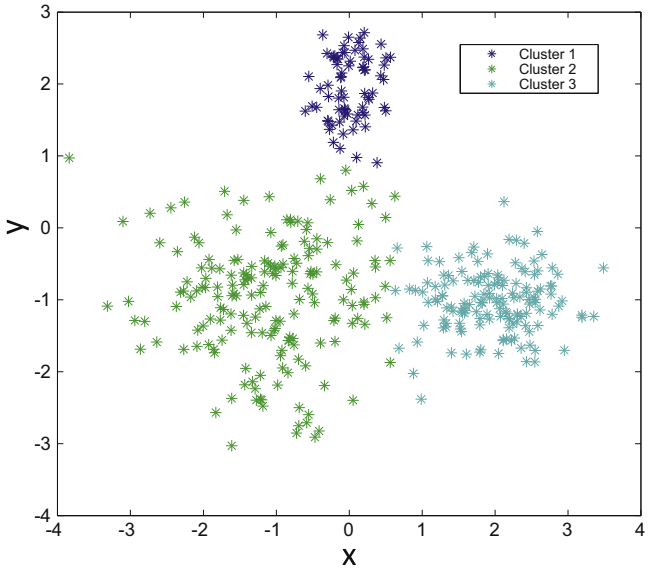

Fig. 7. Best clustering obtained with the proposed GGA with DB index in the third synthetic clustering problem considered.

rectly adjust the optimal number of clusters. Fig. 7 shows the best result obtained by the GGA with DB index as fitness function.

4.2.4. Repository data: Iris and Wine databases

In a final set of experiments, we show the performance of the proposed GGA in two different real problems, from UCI database (Asuncion & Newman, 2007). The first experiment is carried out in the *Iris* problem. This is maybe one the most known and used problems in data mining. It is formed by 3 classes, with 50 observations each. The problem consists of separating three different Iris plants considering characteristics of the flowers: *Iris Sentosa*, *Iris Viginica* and *Iris Versicolor*. Every observation is formed by 4 features, i.e., length and width of sepal and petal of the flower, in centimeters.

Table 4 shows the results obtained by the different compared algorithms. In this problem, the best result is given by the GGA with S index. In this case, the *K*-means obtains a really good solution, slightly worse than the GGA with S index, but better than the solution provided by the GGA with DB index. The DBSCAN approach is not accurate in this problem. The best solution, given by the GGA with S index, finds a solution formed by 50, 45 and 55 objects in each class, with a value of the S coefficient of $S(U) = 0.5325$.

The second instance from the UCI repository considered is the *Wine* problem. Wine is formed by 3 classes, with 59, 41 and 78 data, where each observation represents a class of wine from a different region of Italy. Each sample is formed by 13 features, corresponding to chemical analysis of the wines. Table 5 shows the results obtained by the different compared algorithms. Again the proposed GGA obtains the best results, outperforming *K*-means and DBSCAN. In this case, the GGA with DB index obtains the best result, selecting the correct number of clusters. Note that the DBSCAN approach fails to locate the optimal number of clusters, and provide a poor solution to this problem.

Table 4
Comparison of the results obtained by the proposed grouping genetic algorithm with DB and S index, DBSCAN and *K*-means algorithms in the Iris data set.

Algorithm	# Clusters	Rand index
GGA (DB index)	3	0.8731
GGA (S index)	3	**0.8995**
DBSCAN	2	0.7777
K-means	3	0.8805

Table 5
Comparison of the results obtained by the proposed grouping genetic algorithm with DB and S index, DBSCAN and *K*-means algorithms in the Wine data set.

Algorithm	# Clusters	Rand index
GGA (DB index)	3	**0.7310**
GGA (S index)	3	0.7220
DBSCAN	2	0.3490
K-means	4	0.7019

5. Conclusions

In this paper we have presented a new grouping genetic algorithm for clustering problems. Clustering is an important class of unsupervised learning techniques that have deserved a large amount of research work in the last few years, including machine learning and soft-computing approaches. The grouping evolutionary approach that we have described in this paper is a novel contribution, which uses concepts of grouping encoding and novel adaptations of evolutionary operators. In this work we have given a full description of the encoding and operators, and other implementation details of the algorithm, such as a local search and a parallelization using an island model to improve its performance. We have tested the proposed approach in different synthetic and real clustering problem, obtaining very good results, that improve classical approaches such as *K*-means and DBSCAN algorithms.

Acknowledgement

This work has been partially supported by Spanish Ministry of Science and Innovation, under a project number ECO2010-22065-C03-02.

References

Agustín-Blas, L. E., Salcedo-Sanz, S., Ortiz-García, E. G., Portilla-Figueras, A., & Pérez-Bellido, A. M. (2009). A hybrid grouping genetic algorithm for assigning students to preferred laboratory groups. *Expert Systems with Applications, 36*, 7234–7241.

Agustín-Blas, L. E., Salcedo-Sanz, S., Vidales, P., Urueta, G., & Portilla-Figueras, J. A. (2011). Near optimal citywide WiFi network deployment using a hybrid grouping genetic algorithm. *Expert Systems with Applications, 38*(8), 9543–9556.

Asuncion, A., & Newman, D. J. (2007). UCI Machine Learning Repository, http://www.ics.uci.edu/mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science.

Brown, E. C., & Sumichrast, R. T. (2003). Impact of the replacement heuristic in a grouping genetic algorithm. *Computers & Operations Research, 30*(11), 1575–1593.

Brown, E. C., & Sumichrast, R. T. (2005). Evaluating performance advantages of grouping genetic algorithms. *Engineering Applications of Artificial Intelligence, 18*(1), 1–12.

Brown, E. C., & Vroblefski, M. (2004). A grouping genetic algorithm for the microcell sectorization problem. *Engineering Applications of Artificial Intelligence, 17*(6), 589–598.

Chang, D. X., Zhang, X. D., & Zheng, C. W. (2009). A genetic algorithm with gene rearrangement for *K*-means clustering. *Pattern Recognition, 42*, 1210–1222.

Chang, D., Zhao, Y., Zheng, C., & Zhang, X. (2012). A genetic clustering algorithm using a message-based similarity measure. *Expert Systems with Applications, 39*(2), 2194–2202.

Cura, T. (2012). A particle swarm optimization approach to clustering. *Expert Systems with Applications, 39*(1), 1582–1588.

Das, S., Abraham, A., & Konar, A. (2008). Automatic kernel clustering with a Multi-Elitist Particle Swarm Optimization Algorithm. *Pattern Recognition Letters, 29*(5), 688–699.

Davies, D., & Bouldin, D. (1997). A cluster sparation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 1*(2), 224–227.

Dehuri, S., Ghosh, A., & Mall, R. (2006). Genetic algorithms for multi-criterion classification and clustering in data mining. *International Journal of Computing and Information Systems, 4*(3), 143–154.

De Lit, P., Falkenauer, E., & Delchambre, A. (2000). Grouping genetic algorithms: an efficient method to solve the cell formation problem. *Mathematics and Computers in Simulation, 51*(3-4), 257–271.

Deng, S., He, Z., & Xu, X. (2010). G-ANMI: A mutual information based genetic clustering algorithm for categorical data. *Knowledge-Based Systems, 23*(2), 144–149.

Divina, F., & Aguilar-Ruiz, J. (2006). Biclustering of expression data with evolutionary computation. *IEEE Transactions on Knowledge & Data Engineering, 18*(5), 590–602.

Ester, M., Kriegel, H. P., & Sander, J. (1996). A density-based algorithm for discovering clusters in large spatial data bases with noise. In *Proceedings of the 2nd international conference on knowledge discovery and data mining* (pp. 226–231).

Falkenauer, E. (1992). The grouping genetic algorithm–widening the scope of the GAs. In *Proceedings of the Belgian Journal of operations research, statistics and computer science* (Vol. 33, pp. 79–102).

Falkenauer, E. (1998). *Genetic algorithms for grouping problems*. New York: Wiley.

Gomez-Muñoz, V. M., & Porta-Gándara, M. A. (2002). Local wind patterns for modeling renewable energy systems by means of cluster analysis techniques. *Renewable Energy, 2*, 171–182.

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On Clustering Validation Techniques. *Journal of Intelligent Information Systems, 17*(2-3), 107–145.

Hong, Y., Kwong, S., Chang, Y. C., & Ren, Q. S. (2008). Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. *Pattern Recognition, 41*, 2742–2756.

Hruschka, E. R., & Ebecken, N. F. (2003). A genetic algorithm for cluster analysis. *Intelligent Data Analysis, 7*(1), 15–25.

Hung, C., Sumichrast, Robert T., & Brown, E. C. (2003). CPGEA: a grouping genetic algorithm for material cutting plan generation. *Computers & Industrial Engineering, 44*(4), 651–672.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys, 31*(3), 264–322.

James, T. L., Brown, E. C., & Keeling, K. B. (2007). A hybrid grouping genetic algorithm for the cell formation problem. *Computers & Operations Research, 34*, 2059–2079.

James, T., Vroblefski, M., & Nottingham, Q. (2007). A hybrid grouping genetic algorithm for the registration area planning problem. *Computer Communications, 30*(10), 2180–2190.

Jiang, H., Yi, S., Li, J., Yang, F., & Hu, X. (2010). Ant clustering algorithm with *K*-harmonic means clustering. *Expert Systems with Applications, 37*(12), 8679–8684.

Kanungo, T., Mount, D., Netanyahu, N. S., Piatko, C., Silverman, R., & Wu, A. (2002). An efficient *K*-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(7), 881–892.

Kreng, V. B., & Lee, T. (2004). Modular product design with grouping genetic algorithm - a case study. *Computers & Industrial Engineering, 46*(3), 443–460.

Krishna, K., & Murty, M. N. (1999). Genetic *K*-means algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part B, 29*, 433–439.

Liao, T. W. (2005). Clustering of time series data: a survey. *Pattern Recognition, 38*, 1857–1874.

Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global *K*-means clustering algorithm. *Pattern Recognition, 36*, 452–461.

Lingras, P., & Huang, X. (2005). Statistical, evolutionary, and neurocomputing clustering techniques: cluster-based vs object-based approaches. *Artificial Intelligence Review, 23*(1), 3–29.

Liu, Y., Wu, X., & Shen, Y. (2011). Automatic clustering using genetic algorithms. *Applied Mathematics and Computation, 218*(4), 1267–1279.

Madeira, S. C., & Oliveira, A. L. (2004). Biclustering algorithms for biological data analysis: a survey. *IEEE Transactions on Computational Bioinformatics, 1*(1), 24–45.

Mclachlan, G. J., & Basford, K. E. (1988). *Mixture models: inference and applications to clustering*. New York: Marcel Dekker.

McQueen, J. (1968). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematics and Statistics* (pp. 281–297).

Mitra, S., & Banka, H. (2006). Multi-objective evolutionary biclustering of gene expression data. *Pattern Recognition, 39*, 2464–2477.

Murthy, C. A., & Chowdhury, N. (1996). In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters, 17*, 825–832.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Society, 66*, 846–850.

Sarkar, M., Yegnanarayana, B., & Khemani, D. (1997). A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters, 18*, 975–986.

Scheunders, P. (1997). A genetic c-means clustering algorithm applied to color image quantization. *Pattern Recognition, 30*(6), 859–866.

Tseng, L. Y., & Yang, S. B. (2001). A genetic approach to the automatic clustering problem. *Pattern Recognition, 34*(2), 415–424.

Xiao, J., Yan, Y., Zhang, J., & Tang, Y. (2010). A quantum-inspired genetic algorithm for *K*-means clustering. *Expert Systems with Applications, 37*(7), 4966–4973.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks, 16*(3), 645–678.

Yang, F., Sun, T., & Zhang, C. (2009). An efficient hybrid data clustering method based on *K*-harmonic means and Particle Swarm Optimization. *Expert Systems with Applications, 36*, 9847–9852.

Zahraie, B., & Roozbahani, A. (2011). SST clustering for winter precipitation prediction in southeast of Iran: Comparison between modified *K*-means and genetic algorithm-based clustering methods. *Expert Systems with Applications, 38*(5), 5919–5929.

Zhang, C., Ouyang, D., & Ning, J. (2010). An artificial bee colony approach for clustering. *Expert Systems with Applications, 37*(7), 4761–4767.