Expert Systems with Applications 39 (2012) 2194–2202

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A genetic clustering algorithm using a message-based similarity measure

Dongxia Chang [a,c,*], Yao Zhao [a], Changwen Zheng [b], Xianda Zhang [c]

[a] Institute of Information Science, Beijing Jiaotong University, Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044, China
[b] National Key Lab of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
[c] Tsinghua Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Keywords:
Clustering
Evolutionary computation
Genetic algorithms
Message passing
K-means algorithm

ABSTRACT

In this paper, a genetic clustering algorithm is described that uses a new similarity measure based message passing between data points and the candidate centers described by the chromosome. In the new algorithm, a variable-length real-value chromosome representation and a set of problem-specific evolutionary operators are used. Therefore, the proposed GA with message-based similarity (GAMS) clustering algorithm is able to automatically evolve and find the optimal number of clusters as well as proper clusters of the data set. Effectiveness of GAMS clustering algorithm is demonstrated for both artificial and real-life data set. Experiment results demonstrated that the GAMS clustering algorithm has high performance, effectiveness and flexibility.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Clustering analysis is a widely used unsupervised learning technique for data analysis and can be applied in a variety of engineering and scientific disciplines such as biology analysis, psychology, computer vision, communications, and remote sensing. The primary objective of clustering analysis is to partition a given data set of multidimensional vectors (patterns) into several homogeneous clusters such that patterns in the same cluster are similar to each other in some sense and differentiate from those of other clusters in the same sense. Extensive overviews of clustering algorithms can be found in the literature (Everitt, Landau, & Leese, 2001; Jain & Dubes, 1988; Jain, Murty, & Flynn, 1999; Tou & Gonzalez, 1974; Xu & Wunsch, 2005).

As an important tool for data exploration, clustering analysis examines unlabeled data, by either constructing a hierarchical structure, or forming a set of groups according to a prespecified number. Clustering algorithms may be broadly divided into two classes (Everitt et al., 2001; Xu & Wunsch, 2005): hierarchical and partitional. Both hierarchical clustering and partitional clustering have the drawback that the number of clusters need be specified a prior. For hierarchical clustering, the problem of cluster number selection is equivalent to decide in which level to cut the tree. Partitional clustering algorithms typically require the number of clusters as user input. However, the number of clusters in a data set is always not known beforehand in most situations. A variety of methods have been suggested try to estimate the number of clusters. The classical approach of determining the number of clusters is the use of some validity measures (Milligan & Cooper, 1985; Pal & Bezdek, 1995; Xie & Beni, 1991). For a given range of cluster number, the validity measure is evaluated for each given cluster number and then the value that optimizes the validity measure is chosen. The number of clusters searched by this method depends on the selected clustering algorithm, whose performance may rely on the initialization of the algorithm. Another method is progressive clustering (Krishnapuram & Freg, 1992; Krishnapuram, Frigui, & Nasraoui, 1995), the number of clusters is overspecified. After convergence, spurious clusters are eliminated and compatible clusters are merged. The main problem of this method is the measurement of spurious and compatible clusters. Moreover, they cannot guarantee that all clusters in the data set will be found. An alternative version of the progressive clustering is to seek one cluster at a time until no more "good" clusters can be found (Jolion, Meer, & Bataouche, 1991; Zhuang, Huang, Palaniappan, & Zhao, 1996). The performances of these techniques are also dependent on the validity functions, which are used to evaluate the individual clusters. In order to reduce the effect of the validity functions, a Weighted Sum Validity Function (WSVF), which is a weighted sum of several normalized validity functions, is proposed by Sheng, Swift, and Zhang (2005). Using more than one validity function via a weighted sum approach tends to increase the confidence of the clustering solutions. In this paper, we attempt to use a variable-length genetic algorithm to automatically evolve and find the optimal number of clusters as well as proper clusters of the data set.

Genetic algorithms (GAs) (Goldberg, 1989; Holland, 1975; Jong, 1975; Michalewicz, 1994), an imitation of natural selection and

* Corresponding author at: Institute of Information Science, Beijing Jiaotong University, Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044, China.
E-mail address: chang_dongxia@hotmail.com (D. Chang).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.07.009

survival of the fittest, have been proved to be an efficient way in dealing with the optimization problem. In the past years, several clustering algorithm based on GA have been developed. These algorithms fall into two broad categories based on the representations for the clustering solutions. The first category uses a fixed-length string that the user should specify the desired number of clusters in advance to describe the clustering results (Bandyopadhyay & Maulik, 2002; Hall, Bözyurt, & Bezdek, 1999; Laszlo & Mukherjee, 2007; Maulik & Bandyopadhyay, 2000; Murthy & Chowdhury, 1996; Tucker, Crampton, & Swift, 2005). As the a priori knowledge on the number of clusters is often unavailable in most practical applications, it is important to design an algorithm which can automatically evolve a proper value of the center number as well as provide the appropriate clustering. A large variety of the second-category algorithms are adopting variable-length string, in which the number of cluster centers encoded into an individual is variable. Srikanth, George, and Warsi (1995) proposed a Pittsbugh-style GA for clustering where each individual contains a set of ellipsoid-shaped cluster descriptions. In this method, each cluster description consists of a set of parameters specifying the size and shape of an ellipsoid and all the parameters are encoded using binary digits. Ghozeil and Fogel (1996) proposed an evolutionary programming algorithm for clustering where each individual contains a set of hyperbox-shaped cluster descriptions. Both these two algorithms were making an assumption on the shape of the data set, when the data set violates the assumption the clustering results will be unsatisfactory. In order to overcome this drawback, Bandyopadhyay and Maulik (2002) proposed an automatic clustering algorithm which does not assume any particular underlying shape of the data set. But when the clusters are overlapping, this method prefers to class these clusters into one cluster. Saha and Bandyopadhyay (2009) proposed a fuzzy, point symmetry based genetic clustering technique (fuzzy-VGAPS), which can automatically determine the number of clusters present in a data set as well as a good fuzzy partitioning of the data.

In order to improve the performance of the GA-based algorithms, a new genetic clustering algorithm using a message-based similarity measure (GAMS) is presented in this paper. By utilizing a problem-specific chromosome structure and a set of genetic operators, the GAMS clustering algorithm can find the optimal number of clusters as well as proper structure of the data set automatically. In the new algorithm, a new similarity measure which we call the message-based similarity is proposed. This measure takes into account the messages exchanged between the data points and the candidate centers described by the chromosome. The usage of this new similarity improves the performance of the clustering greatly.

The rest of this paper is organized as follows. Section 2 provides the message-based similarity measure. Then a description of our GAMS clustering algorithm is presented in Section 3. The details of the new algorithm including the representation, the fitness evaluation function, the genetic operators are given in this section. Experimental results are provided for several artificial and real-life data sets are given in Section 4. The experimental results demonstrate the effectiveness of the GAMS clustering algorithm. Finally, conclusions are drawn in Section 5.

2. Message-based similarity

In this section, we propose a new similarity measure for the clustering criteria, which we call the message-based similarity. The measure is so called because it uses two kinds of message, responsibility and availability, exchanged between data points and the candidate centers, and each takes into account a different kind of competition. Here, the responsibility and availability between the data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and the candidate centers set $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$ are defined.

For the candidate center set $\mathbf{C}$, an input preference that candidate center $\mathbf{c}_k \in \mathbf{C}$ be chosen as a center is defined firstly. The candidate centers with larger values of input preference are more likely to be chosen as a center. If a priori, this value can be set according to the priori information. Here, we define it as

$$IP(k) = -\frac{1}{n}\sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{c}_k) = -\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{c}_k\|^2, \quad k = 1, 2, \ldots, K. \tag{1}$$

And this is the mean distance between a center and all the data points in the data set. Obviously, this value will be optimized when c_k is the center of the data set. Note that the distance measure here is chosen with the Euclidean norm. However, any suitable distance measure can be used to replace the Euclidean norm. Throughout this paper, we use the Euclidean norm. In the following, the responsibility and availability are defined.

The responsibility $r(i,k)$, sent from data point $\mathbf{x}_i$ to the candidate center $\mathbf{c}_k$, reflects the evidence for how well-suited $\mathbf{c}_k$ is to sever as the center for point $\mathbf{x}_i$, taking into account other potential centers for point $\mathbf{x}_i$. The availability $a(i,k)$, sent from candidate center $\mathbf{c}_k$ to point $\mathbf{x}_i$, reflects the evidence for how appropriate it would be for point $\mathbf{x}_i$ to choose $\mathbf{c}_k$ as its center, taking into account the support from other points that $\mathbf{c}_k$ should be an center. The responsibilities are computed using the rule

$$r(i,k) = d(i,k) - \max_{k', s.t. k' \neq k} \{d(i,k')\}, \quad i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, K, \tag{2}$$

where $d(i,k)$ denotes the distance between data point $\mathbf{x}_i$ and the candidate cluster center $\mathbf{c}_k$. Here the distance measure used is the Euclidean distance, i.e., $d(i,k) = -\|\mathbf{x}_i - \mathbf{c}_k\|^2$. A self-responsibility, $R(k)$, is defined as

$$R(k) = IP(k) - \max_{k', s.t. k' \neq k} \{d(\mathbf{c}_k, \mathbf{c}_{k'})\}, \quad k = 1, 2, \ldots, K, \tag{3}$$

i.e., the self-responsibility of $\mathbf{c}_k$ is defined as its input preference $IP(k)$ minus the largest of the similarities between center $\mathbf{c}_k$ and all other candidate centers. This self-responsibility reflects evidence that center $\mathbf{c}_k$ is a center, based on its input preference tempered by how ill-suited it is to be assigned to another center. A negative self-responsibility $R(k)$ indicates that center $\mathbf{c}_k$ is currently better suited as belonging to another center rather than being a center itself.

Whereas the above responsibility update lets all candidate centers compete for ownership of a data point, the following availability update gathers evidence from data points as to whether each candidate centers would make a good center

$$a(i,k) = \min\left\{0, R(k) + \sum_{i', s.t. i' \neq i} \max\{0, r(i',k)\}\right\},$$
$$i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, K. \tag{4}$$

This availability $a(i,k)$ reflects evidence that point $\mathbf{c}_k$ is a center, based on the positive responsibilities sent to candidate center from other points. Here, only the positive responsibilities are added, because it is only necessary for a good center to explain some data points well, regardless of how poorly it explains other data points.

After the computation of the responsibility and availability, the similarity between the data point and the candidate center is defined by the sum of the responsibility r and the availability a. That is to say, the similarities between data point $\mathbf{x}_i$ and the candidate centers $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$ are

$$s(i,j) = r(i,j) + a(i,j), \quad j = 1, 2, \ldots, K, \tag{5}$$

then $\mathbf{x}_i$ will be assign to the cluster with the maximum similarity.

3. The GAMS clustering algorithm

In this section, the GAMS clustering algorithm is proposed. This algorithm uses a variable-length encoded chromosome to automatically clustering the data set. The number of the cluster centers and the structure of the data set are evolved simultaneously. In the evolutionary process, a penalized fitness evaluation function is used to compute the fitness of the individual. A set of problem-specific evolutionary operators used are described in detail. Finally, the number of the centers and the clusters of the data set are acquired. We will introduce the GAMS clustering algorithm in the following.

3.1. Chromosome representation

For any GA, a chromosome representation is needed to describe each individual in the population of interest. The representation method determines how the problem is structured in the algorithm and the genetic operators that are used. Each chromosome is made up of a sequence of genes from certain alphabet. An alphabet can consist of binary digits (0 and 1), floating-point numbers, integers, symbols (i.e., A, B, C, D), etc. In early GAs, the binary digit was used. It has been shown that more natural representations can get more efficient and better solutions. Michalewicz (1994) has performed extensive experiments comparing real-valued and binary GAs and shown that the real-valued GA is more efficient in terms of CPU time.

In our method, a real-valued problem-specific chromosome representation is used, e.g., a chromosome corresponds to a clustering result that described by the cluster centers. Each chromosome is described by a sequence of $M = N \times K_i$ real-valued numbers where N is the dimension of the feature space, K_i is the number of clusters described by the chromosome. That is to say, the chromosome of the algorithm is written as

$$\mathbf{v} = \left[v_{11}, v_{12}, \ldots, v_{1N}, v_{21}, v_{22}, \ldots, v_{2N}, \ldots, v_{K_i1}, v_{K_i2}, \ldots, v_{K_iN}\right], \tag{6}$$

where the first N values represent the first cluster center, the next N points represent the second center, and so on. Because K_i is different for every chromosome and will change in the evolutionary process, the representation is of variable-length.

3.2. Population initialization

In the GAMS clustering algorithm, an initial population of size P can be randomly generated. For each population, a number $K_i \in [K_{\min}, K_{\max}]$ is generated randomly, where $K_{\min}$ and $K_{\max}$ is the lower and upper bound of the number of clusters, respectively. Then K_i points are chosen randomly from the data set but on the condition that there are no identical points to form a chromosome, presenting the K_i cluster centers. This process is repeated until P chromosomes are generated. Here, the trivial clustering (that all the data points are belong to one cluster) will not be considered, so $K_{\min}$ is chosen to be 2. (The trivial partitions, i.e., all the data points are belong to one cluster, are not considered here). And $K_{\max}$ is taken to be $\sqrt{n}$, which is rule of thumb used by many investigators in the literature (Pal & Bezdek, 1995).

After the population initialization, each data point of the set is assigned to the cluster with the message-based similarity measure.

3.3. Fitness evaluation

The fitness function is used to define a fitness value to each candidate solution. Due to the number of the cluster centers is not definitely, the fitness function should consider the three measures at the same time, as follows.

(1) A measure of the cohesiveness of clusters, favoring dense clusters.
(2) A measure of the distance between clusters and the global center, which favors well-separated clusters.
(3) A measure of the "implicit" of the number of clusters, which favors candidate solutions with a smaller number of clusters.

That is to say, the aim is to find a set of centers for which the within-cluster spread is small, the between-cluster spread is large in some sense, and the number of clusters is moderate. For the circumstance where the number of clusters is know, one popular measure is (Fukunaga, 1990)

$$J = \max\left\{Tr\{\mathbf{S}_W^{-1}\mathbf{S}_B\}\right\}, \tag{7}$$

where $\mathbf{S}_W$ is the within-cluster variation and $\mathbf{S}_B$ is the between-cluster variation. $\mathbf{S}_W$ measures how compact or tight the clusters are and it is defined as

$$\mathbf{S}_W = \sum_{i=1}^{K}\sum_{\mathbf{x}_j \in C_i}(\mathbf{x}_j - \boldsymbol{\mu}_i)^T(\mathbf{x}_j - \boldsymbol{\mu}_i), \tag{8}$$

where $\boldsymbol{\mu}_i$ is the center of C_i and is defined as $\boldsymbol{\mu}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in C_i}\mathbf{x}$, n_i is the cardinality of C_i, i.e., the number of points in cluster C_i. $\mathbf{S}_B$ measures how scatter the cluster centers are from the sample mean and it is given by

$$\mathbf{S}_B = \sum_{i=1}^{K} n_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T(\boldsymbol{\mu}_i - \boldsymbol{\mu}), \tag{9}$$

where $\boldsymbol{\mu}$ is the sample mean

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{l=1}^{n}\mathbf{x}_l = \frac{1}{n}\sum_{i=1}^{K} n_i\boldsymbol{\mu}_i. \tag{10}$$

But this measure is not considering the influence of the number of the centers on the clustering result. Here, we define a penalized cost function as

$$J = \frac{1}{K^2}Tr\{\mathbf{S}_W^{-1}\mathbf{S}_B\}, \tag{11}$$

where K is the number of cluster centers of the chromosome. This cost function ensures that the individual is penalized to discourage partitions that have more clusters.

3.4. Evolutionary operators

In the evolutionary process, seven types of operators: one crossover operator and six mutation operators are used. These operators manipulate the centers to evolve chromosomes into possibly better ones. The application of each operator is controlled by a probability.

3.4.1. Crossover

The main goal of the crossover operator is to create diversified and potentially promising new chromosomes. Crossover combines the features of two parent chromosomes to form two offspring by swapping corresponding segments of the parents. The intuition behind the applicability of the crossover operator is information exchange between different potential solutions. Simple crossover is used in this paper. Note that although the crossover points can fall in different locations in the two individuals, they are restricted to fall on the same location within each cluster description. For instance, suppose crossover occurs between the following two individuals

[(6.65, 9.84)(9.98, |6.18)(13.65, 11.03)]
[(12.60, 10.21)(6.49, 9.90)(11.30, |9.72)(9.75, 13, 95)(9.90, 7.11)(18.21, 12.20)].

The crossover points fell in the second and the third cluster descriptions of the two parent individuals, respectively, but both crossover points fell right after the first coordinate within each cluster description. The number of centers of the parent is three and six, respectively. The children of the above crossover are:

[(6.65, 9.84)(9.98, |9.72)(9.75, 13, 95)(9.90, 7.11)(18.21, 12.20)]
[(12.60, 10.21)(6.49, 9.90)(11.30, |6.18)(13.65, 11.03)].

The number of the centers of the two children is five and four, respectively.

3.4.2. Mutation

Mutation arbitrarily alters one or more genes of a selected chromosome, by a random change with a probability equal to the mutation rate. The intuition behind the mutation operator is the introduction of some extra variability into the population. The following mutation operators are investigated.

1. Perturb mutator
   The perturb mutator is used to impose a random change to the coordinates of a cluster center in an individual. Given an individual, the operator randomly selects a cluster center and changes the coordinates of this center randomly. This mutate operator is not change the number of the cluster centers. In the following K_i denotes the number of the cluster centers of the individual under mutation.
2. Insert mutator
   The insert mutator operates on an individual by inserting randomly generated new center into the individual. Given an individual representing a partition:
   - Randomly generate $m \in [1, K_i]$;
   - Select a point in the data set at random and insert it into the individual at the mth location;
   - Set $K_i = K_i + 1$.
3. Delete mutator
   The delete mutator randomly deletes a center from an individual which has the opposite effect of the insert mutator. Given an individual representing a partition:
   - Randomly generate $m \in [1, K_i]$;
   - Delete the mth center of the individual;
   - Set $K_i = K_i - 1$.
4. Merge mutator
   The merge mutator randomly merges two sets represented by two centers. Given an individual representing a partition:
   - Pick values m_1 and m_2 satisfying $m_1, m_2 \in [1, K_i]$ at random;
   - Delete the cluster described by the m_2th center;
   - Recompute the value of the m_1th center of the individual according to the following equation

     $$\mathbf{c}_{m_1}^{new} = \frac{|C_{m_1}|\mathbf{c}_{m_1} + |C_{m_2}|\mathbf{c}_{m_2}}{|C_{m_1}| + |C_{m_2}|},$$

     where $|C_{m_i}|$ be the cardinality of the m_ith cluster and $\mathbf{c}_{m_i}$ be the center of the m_ith cluster, $i = 1, 2$;
   - Set $K_i = K_i + 1$;
5. Split mutator
   The split mutator splits one cluster into two clusters. Given an individual representing a partition:
   - Randomly generate $m \in [1, K_i]$;
   - Split the mth center into two subclusters each of k_1 and k_2 elements and $k_1 + k_2 = |C_m|$;
   - Compute the centers of the two new clusters;
   - Set $K_i = K_i + 1$.
6. Move mutator
   Move mutator transfers one data point from one cluster to another cluster. Given an individual representing a partition:
   - Randomly generate $m \in [1, N]$, $l \in [1, K_i]$ and let $\mathbf{x}_m \in C_j$;
   - If $C_l \neq C_j$, then move $\mathbf{x}_m$ from C_j into C_l, else goto the first step;
   - Recompute the ceters of C_j and C_l, and change the corresponding values of the individual.

3.5. Description of the algorithm

In our new algorithm, a chromosome with variable-length representing the cluster is used and each chromosome is individually evaluated by using the fitness function described in Section 3.3. The data set with the proposed algorithm can self-organize the cluster number and cluster structures. In the evolutionary loop, a set of individuals is selected for evolutionary crossover and mutation. A roulette wheel selection of P slots is used to implement the selection process. The chance for a chromosome to be selected is proportional to its fitness value.

An evolutionary operator is selected on the basis of a probability distribution. The crossover operator transforms two individuals (parents) into two offspring by combining parts from each parent. The mutation operator operates on a single individual and creates an offspring by mutating that individual (see Section 3.4 for details on evolutionary operators). The newly generated individuals are evaluated on the basis of the fitness function and form the new generation. The elitist strategy (Jong, 1975) is used in each generation by replacing the worst chromosome of currently population with the best one seen up to the previous generation. The process terminates after some number of generations, which can be fixed either by the user or determined dynamically by the program itself, and the best chromosome obtained is taken to be the best solution.

The GAMS clustering algorithm is described as follows:

1. Initialize a group of cluster centers with size of P, only nontrivial clustering (that all the data points are not belong to one cluster) are considered. Each data point of the set is assigned to the cluster with the new similarity measure described in Section 2.
2. Evaluate each chromosome and copy the best chromosome say p_{best} of the initial population in a separate location.
3. If the termination condition is not reached, go to Step 4. Otherwise, select the best individual from the population as the best clustering result.
4. Select individuals from the population for crossover and mutation.
5. Apply crossover operator to the selected individuals based on the crossover probability.
6. Apply mutation operator to the selected individuals based on the mutation probability.
7. Evaluate the newly generated candidates.
8. Compare the worst chromosome in the new population with p_{best} in term of their fitness values. If the worst one is worse than p_{best}, then replace it by p_{best}.
9. Find the best chromosome in the new population and replace p_{best}.
10. Go back to Step 3.

4. Experiments results

In this section, the performances of the GAMS clustering, GCUK-clustering (Bandyopadhyay & Maulik, 2002) and K-means algorithms are compared through the experiments. The experiments were conducted on both artificial data and real-life data from the UCI Machine Learning Repository. The results show that GAMS

clustering algorithm has high performance, effectiveness and flexibility.

In the experiments, the following artificial data sets are similar to those used in Bandyopadhyay and Maulik (2002) these were generated by us, but keeping the structure of the clusters as close as possible to that described in Bandyopadhyay and Maulik (2002) and three real-life data sets were used. These data sets used are divided into three groups.

(1) Group 1: This group contains three data sets. The clusters present in these data sets are without overlap and completely separable.

Data 1: This data set consists of 76 two dimensional data points distributed over three clusters. This data set is shown in Fig. 1(a).

Data 2: This data set consists of 400 three dimensional data points distributed over four hyper-spherical disjoint clusters where each cluster contains 100 data points. This data set is shown in Fig. 2(a).

Data 3: This data set is two dimensional and comprises a ring-shaped cluster, a rectangular cluster and a linear cluster as shown in Fig. 3(a). The total number of points equal to 400 and three clusters.

(2) Group 2: This group consists of two data sets. The clusters present in these data sets are highly overlapping.

Data 4: This data set consists of 150 three dimensional data points distributed over three clusters as shown in Fig. 4(a). Each cluster is consisting of 50 data points. Two of the clusters are overlapping.

Data 5: This data set consists of 250 two dimensional data points distributed over five spherically shaped clusters as shown in Fig. 5(a). The clusters present here are highly overlapping, each consisting of 50 data points.

(3) Group 3: This category consists of three real-life data sets: Iris, Breast Cancer and Wine.

Iris: Iris data set consists of 150 data points distributed over three clusters. Each cluster has 50 points. This data set represents different categories of irises characterized by 4 feature values in centimeters: the sepal length, sepal width, petal length and the petal width. This data set has three classes Setosa, Versicolor and Virginica. It is known that the last two classes have a large amount of overlap while the first class is linearly separable.

Breast cancer: This data set consists of 683 sample points. Each pattern has nine features corresponding to clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. There are two categories in the data.

Wine: This is the wine recognition data consisting of 178 instances with 13 features resulting from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wine.

For convenience, we summarize the eight sets in Table 1 with the characteristics of the data sets. The four columns show the number of data points n, the number of clusters K, the dimension of the feature space d, and the number of points in every cluster for each data set.

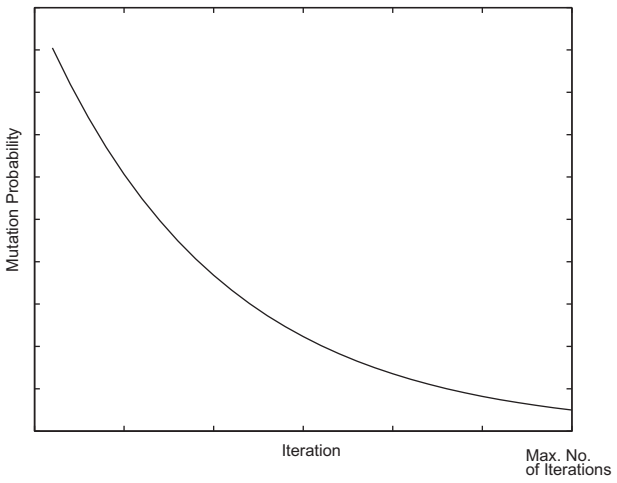

Fig. 1. The variation of mutation probability with the number of iterations adopted in the GAMS clustering algorithm.

In the experiments, the population size is taken as 50. The crossover and mutation probabilities for GUCK-clustering algorithm are $p_c = 0.8$ and $p_m = 0.001$, respectively. For the GAMS clustering algorithm, the mutation probability varies follow the function (Murthy & Chowdhury, 1996) shown in Fig. 1. We have started with a mutation probability value of $p_m = 0.5$. The value is then varied as a step function of the number of iterations until it reaches a value of 0.001. The minimum value of the mutation probability is taken to be 0.001. The probabilities for the six mutation operators are set to be equal to 1/6. The total number of generations is equal to 50. For the K-means algorithm, the actual number of clusters is known prior. All the experiments run for 20 independent times. Table 2 shows the mean and standard deviations of the number of classes obtained by the two automatic clustering algorithms (GCUK-clustering and GAMS), averaged over 20 independent runs (Here we only show the results for the data sets of Group 2 and Group 3). It also shows the percentage of runs that managed to yield the correct number of classes for each data set. From Table 2, it can be seen that for all the data sets in Group 2 and Group 3, GAMS obtains the exact K (the final result of K is obtained through voting). But the GCUK-clustering just gets the optimal result in Data 5 data set and Breast data set and cannot get the exact K in the other data sets.

In the following, the clustering results of artificial data sets obtained by K-means, GCUK-clustering and GAMS algorithm are given in Figs. 2–6.

From Figs. 2–4, we can see that all algorithms work well for the data sets in Group1. For Data 1 and Data 2, all algorithms can separate the data points correctly. For Data 3, the performance of GAMS algorithm is superior to other algorithms for the number of data points with wrong cluster number is small.

Figs. 5 and 6 provide the clustering results for Data 4 and Data 5. It is clear that the performance of the GAMS algorithm is better than other algorithms. Especially for Data 4, GCUK-clustering algorithm only gets two clusters. In order to compare them further, the Adjusted Rand Index is used.

The Adjusted Rand Index (Hubert, 1985) measures the agreement of the clustering result with the true cluster structure. Here, it is used to measure the clustering performance for the data sets of Group 2 and Group 3. Let n_{ij} denote the number of points in cluster i of partition produced by the algorithm ($i = 1,2,\ldots,c_m$) and group j of the true cluster structure ($j = 1,2,\ldots,c_t$), where c_m and c_t are the number of clusters of obtained by the algorithm and the true structure, respectively. Then, $n_{i\cdot} = \sum_{j=1}^{c_t} n_{ij}, n_{\cdot j} \sum_{i=1}^{c_m} n_{ij}$, and n give the mar-

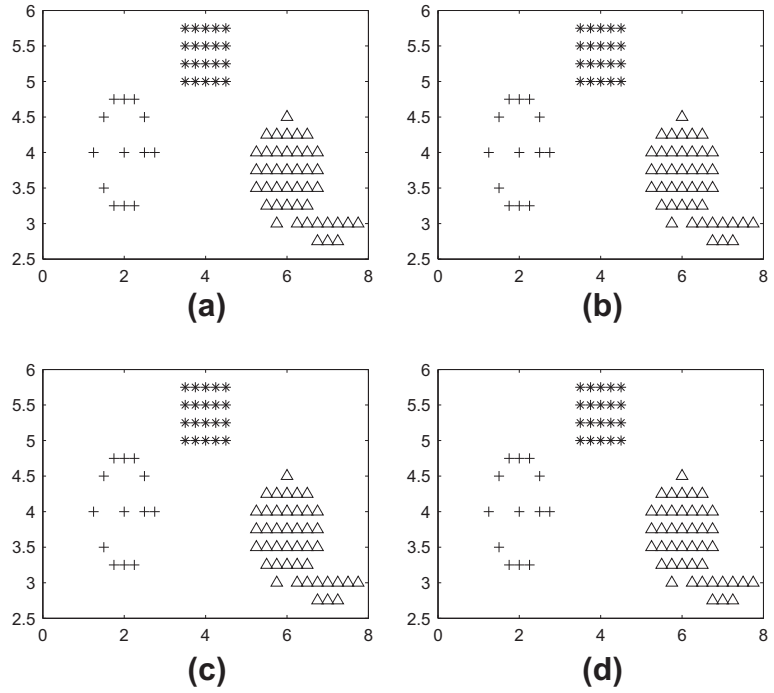

Fig. 2. The original Data 1 and the clustering results of Data 1 obtained by the three algorithms for (a) the original data set; (b) *K*-means; (c) GCUK-clustering; (d) GAMS.

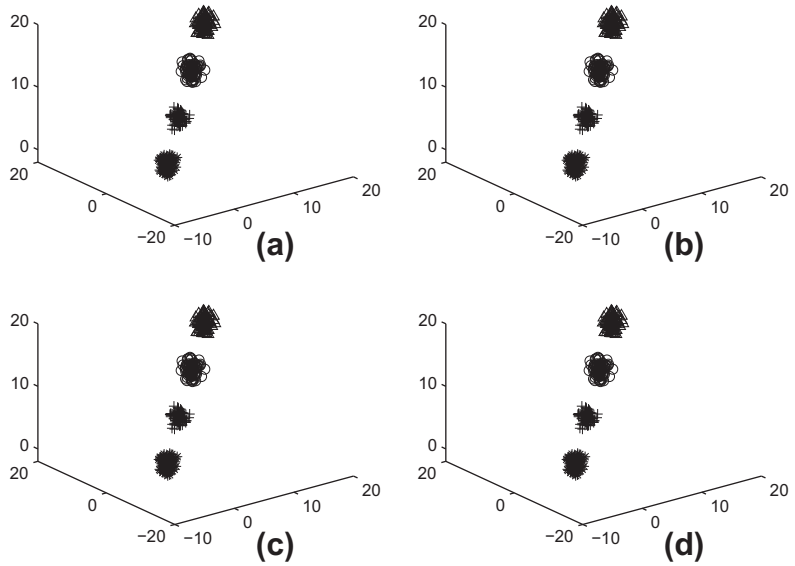

Fig. 3. The original Data 2 and the clustering results of Data 2 obtained by the three algorithms for (a) the original data set; (b) *K*-means; (c) GCUK-clustering; (d) GAMS.

ginals and grand total for such a classification table. Given these values, the Adjusted Rand index is defined as (Hubert, 1985)

$$ARI = \frac{\sum_{l=1}^{c_m}\sum_{k=1}^{c_t} C_{n_{lk}}^2 - \left[\sum_{l=1}^{c_m} C_{n_l}^2 \cdot \sum_{k=1}^{c_t} C_{n_k}^2\right]/C_n^2}{\left[\sum_{l=1}^{c_m} C_{n_l}^2 + \sum_{k=1}^{c_t} C_{n_k}^2\right]/2 - \left[\sum_{l=1}^{c_m} C_{n_l}^2 \cdot \sum_{k=1}^{c_t} C_{n_k}^2\right]/C_n^2}, \quad (12)$$

where $C_n^2 = n(n-1)/2$. The Adjusted Rand Index return values in the interval [0,1] and the optimum score is 1, with higher scores being "better". We will use this index to measure the performance of the clustering results obtained by the algorithms for Group 2 and Group 3.

For each data set we have conducted the experiment 20 independent times with randomly generated initialization and the average value recorded to account for the stochastic nature of the algorithm. Table 3 gives the mean and variance of the Adjusted Rand Index obtained by the three techniques for Group 2 and Group 3. As seen from the table, the Adjusted Rand Index obtained by GAMS clustering algorithm is always better than that of the other algorithms for the data sets. For the Iris data set, GCUK-clustering algorithm only gets two clusters, one corresponding to the

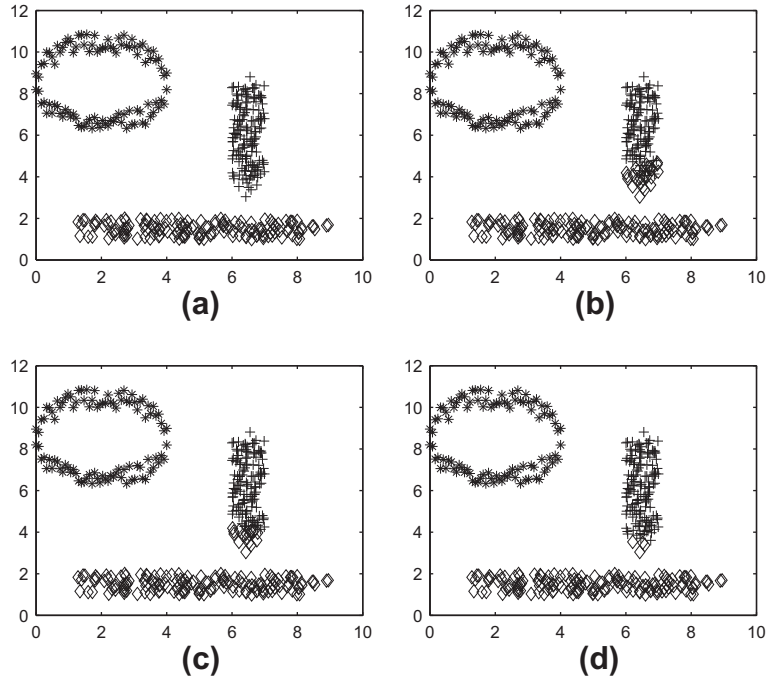

Fig. 4. The original Data 3 and the clustering results of Data 3 obtained by the three algorithms for (a) the original data set; (b) *K*-means; (c) GCUK-clustering; (d) GAMS.

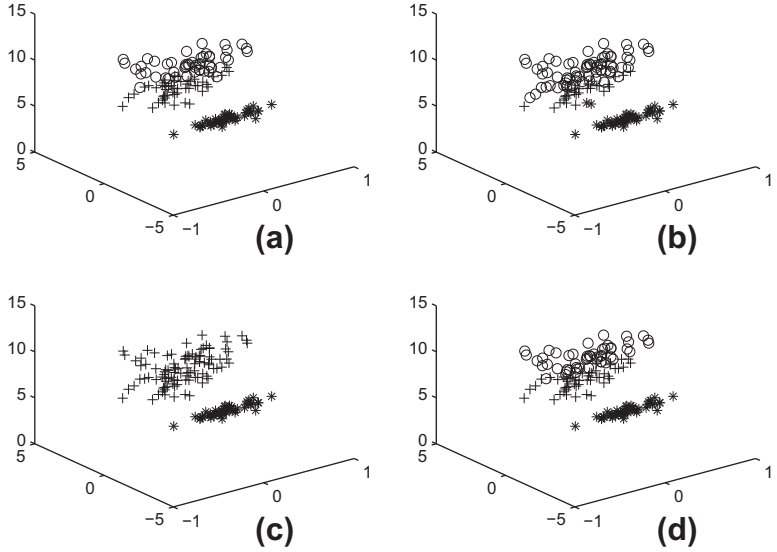

Fig. 5. The original Data 4 and the clustering results of Data 4 obtained by the three algorithms for (a) the original data set; (b) *K*-means; (c) GCUK-clustering; (d) GAMS.

Table 1
Eight data sets used in our experiments.

Data set	*M*	*K*	*d*	Points per cluster
Data 1	76	3	2	43, 20, 13
Data 2	400	4	3	100 per cluster
Data 3	400	3	2	100, 150, 150
Data 4	150	3	3	50 per cluster
Data 5	250	5	2	50 per cluster
Iris	150	3	4	50 per cluster
Breast	683	2	10	444, 239
Wine	178	3	13	59, 71, 48

Table 2
Mean number of clusters found (with standard deviation) and percentage of successful runs obtained by the two automatic algorithms over 20 independent runs on the five data sets in Group 2 and Group 3.

Data set	Actual number of clusters	GCUK-clustering	GAMS
Data 4	3	2.05 ± 0.0500, 1	2.9 ± 0.0947, 18
Data 5	5	4.70 ± 0.4316, 13	4.85 ± 0.2395, 15
Iris	3	5.75 ± 1.5658, 2	3 ± 0.0947, 18
Breast	2	2.3 ± 0.2211, 14	2 ± 0, 20
Wine	3	9.1792 ± 0.6122, 0	2.95 ± 0.1553, 18

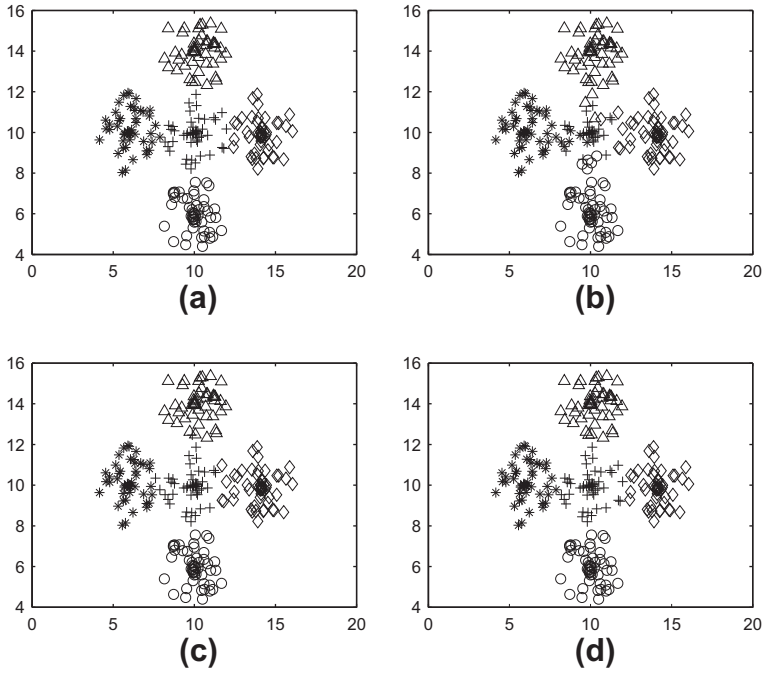

Fig. 6. The original Data 5 and the clustering results of Data 5 obtained by the three algorithms for (a) the original data set; (b) *K*-means; (c) GCUK-clustering; (d) GAMS.

Table 3
The mean and variance of the Adjusted Rand Index obtained by *K*-means, GCUK-clustering, and GAMS algorithms for Group 2 and Group 3.

	K-means	GCUK-clustering	GAMS
Data 4	0.6091 ± 0.0078	*	0.7776 ± 0.0013
Data 5	0.7056 ± 0.0152	0.7322 ± 0.0215	0.9883 ± 9.3683e^{-4}
Iris	0.6357 ± 0.0017	*	0.8138 ± 0.0037
Breast	0.8261 ± 0.0041	0.8695 ± 1.4442e^{-4}	0.8884 ± 6.5947e^{-5}
Wine	0.6807 ± 0.0054	*	0.7662 ± 0.0165

Table 4
Result of MANOVA testing by *K*-means, GCUK-clustering and GAMS algorithms on two data sets, here *gm* stands for Mahalanobis distance and $dataname_i$ denotes the cluster number of the data set.

	K-means			GCUK-clustering			GAMS		
	d	*p*	*gm*	*d*	*p*	*gm*	*d*	*p*	*gm*
$Iris_1$	0	1	0				0	1	0
$Iris_2$	0	0.2640	0.2417		*		0	0.4110	0.1199
$Iris_3$	0	0.0798	0.3143				0	0.4129	0.1169
$Breast_1$	0	0.9594	0.0169	0	0.9638	0.0164	0	0.9863	0.0125
$Breast_2$	0	0.9992	0.0036	0	0.9999	0.0072	0	1	0.0016
$wine_1$	0	0.9485	0.2032				0	0.9797	0.1641
$Wine_2$	0	0.9081	0.2528		*		0	0.9955	0.1146
$Wine_3$	0	0.8956	0.2976				0	0.9999	0.0753

first class, and the other to the combination of the last clusters. While the GAMS clustering algorithm provides three clusters for this data set, and the performance is better than that of the *K*-means.

In order to compare the algorithms more careful, the multivariate analysis of variance (MANOVA) technique (Anderson, 1984) is used to assess the cluster differences between the actual clusters and those obtained by *K*-means, GCUK-clustering and GAMS. MANOVA is a powerful statistical tool used to provide information on the nature and predictive power of the independent measures. It measures the group difference between two or more metric dependent variables simultaneously, using a set of categorical non-metric variables. Here, the categorical non-metric variables are the cluster labels. The results are given in Table 3. In the experiment, MANOVA tests the null hypothesis that the mean of each group is the same dimensional multivariate vector, and that any difference observed in the sample is due to random chance. There are three outputs, d, p and a distance between the group means, in the experiment. If $d = 0$, there is no evidence to reject the null hypothesis; while if $d = 1$, we can reject the null hypothesis that the means are the same but we cannot reject the hypothesis that the multivariate means lie on the same line. Similarly, if $d = 2$, then the multivariate means may lie on the same plane in n-dimensional space, but not on the same line. p is the probability, computed assuming that the null hypothesis is true. The smaller the p is, the stronger is the evidence against the null hypothesis provided by the data. If $p < 0.05$, then we will reject the null hypothesis. The *gm* in Table 4 represents the Mahalanobis distance between each pair of group means.

It can be seen from Table 4 that GAMS clustering algorithm is superior to the other algorithms. For the Iris data set, GAMS clustering algorithm can find three clusters. But GCUK-clustering algorithm only provided two clusters, one corresponding to the first class, and the other to the combination of the last clusters. For the first cluster of the Iris data set, all the algorithms are able to find the first cluster correctly, ($d = 0$, $p = 1$, $gm = 0$). This signifies that means of data items forming cluster 1 after application of the algorithms and that of the actual cluster are the same. GAMS cannot find the other two clusters accurately for the two sets. But p value of GAMS is better than p values of *K*-means algorithm. This is also evident from the *gm* value obtained by GAMS is smaller than the *gm* value obtained by all the other algorithms. For the breast data set, all the algorithms give high correspondence to the actual clusters, for both cluster one and cluster two. This is evident from the d, p and *gm* values in Table 4. For all the algorithms, we obtain $d = 0$, but the p value of GAMS is much better than p values of other algorithms. The *gm* value of GAMS is also smaller than the values obtained by other algorithms.

5. Conclusions

In this paper, a novel evolutionary algorithm, called GAMS, has been developed for clustering problem with unknown cluster number. The GAMS clustering algorithm utilizes domain specific knowledge to make decision and can find the optimal number of clusters as well as the cluster centers automatically. As the number of clusters is not known a priori in most practical circumstance, GAMS clustering algorithm can be used more widely. The new similarity measure used takes into account both the distance between the data point with the nearest center and that with neighboring data points. This will improve the performance of the clustering greatly. In order to evaluate the performance of the individual, a new cost function which penalizes the clusters that have more clusters was defined. The superiority of the GAMS clustering algorithm over GCUK-clustering, and *K*-means algorithm has been demonstrated by the experiments. All the experiment results described in this paper demonstrated that our algorithm can find proper structures of various data sets.

Acknowledgments

This paper was supported by the Fundamental Research Funds for the Central Universities of China (2011JBM026); the China Postdoctoral Science Foundation (00480190); the Beijing Municipal Natural Science Foundation (4113075).

References

Anderson, T. W. (1984). *An introduction to multivariate statistical analysis*. New York: Wiley.

Bandyopadhyay, S., & Maulik, U. (2002). An evolutionary technique based on *K*-means algorithm for optimal clustering in RN. *Information Sciences, 146*, 221–237.

Bandyopadhyay, S., & Maulik, U. (2002). Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition, 35*, 1197–1208.

Everitt, B., Landau, S., & Leese, M. (2001). Cluster Analysis. London Arnold.

Fukunaga, K. (1990). *Statistical pattern recognition* (2nd ed.). San Diego, CA: Academic Press.

Ghozeil, A., & Fogel, D. B. (1996). Discovering patterns in spatial data using evolutionary programming. In *Genetic programming 1996: Proceedings of the 1st annual conference* (pp. 521–527). MIT Press.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Hall, L. O., Bözyurt, I., & Bezdek, J. C. (1999). Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation, 3*(2), 103–112.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Hubert, A. (1985). Comparing partitions. *Journal of Classification, 2*, 193–198.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood Cliffs: Prentice-Hall.

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

Jolion, J. M., Meer, P., & Bataouche, S. (1991). Robust clustering with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*(8), 791–802.

Jong, K. A. De (1975). An analysis of the behavior of a class of genetic adaptive Systems, Doctoral dissertation, University of Michigan, Ann Arbor, Michigan.

Krishnapuram, R., & Freg, C. P. (1992). Fitting an unknown number of lines and planes to image data through compatible cluster merging. *Pattern Recognition, 25*(4), 385–400.

Krishnapuram, R., Frigui, R., & Nasraoui, O. (1995). Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation. *IEEE Transactions on Fuzzy Systems, 3*(1), 29–60.

Laszlo, M., & Mukherjee, S. (2007). A genetic algorithm that exchanges neighboring centers for *k*-means clustering. *Pattern Recognition Letter, 28*, 2359–2366.

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm based clustering technique. *Pattern Recognition, 33*, 1455–1465.

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs*. New York: Springer. AI Series.

Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika, 50*, 159–179.

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

Pal, N. R., & Bezdek, J. C. (1995). On cluster validity for fuzzy *c*-means model. *IEEE Transactions on Fuzzy Systems, 1*, 370–379.

Saha, S., & Bandyopadhyay, S. (2009). A new point symmetry based fuzzy genetic clustering technique for automatic evolution of clusters. *Information Science, 179*(19), 3230–3246.

Sheng, W. G., Swift, S., Zhang, L. S., et al. (2005). A weighted sum validity function for clustering with a hybrid niching genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 35*(6), 1156–1167.

Srikanth, R., George, R., Warsi, N., et al. (1995). A variable-length genetic algorithm for clustering and classification. *Pattern Recognition Letters, 16*, 789–800.

Tou, J. T., & Gonzalez, R. C. (1974). *Pattern recognition principles*. Reading, MA: Addison-Wesley.

Tucker, A., Crampton, J., & Swift, S. (2005). RAFGA: An efficient representation and crossover for grouping genetic algorithms. *Evolutionary Computation, 13*(4), 477–499.

Xie, X. L., & Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*, 841–847.

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

Zhuang, X., Huang, Y., Palaniappan, Y., & Zhao, Y. (1996). Gaussian mixture density modeling, decomposition and applications. *IEEE Transactions on Image Processing, 5*(9), 1293–1302.