Automatic Clustering Using an Improved Differential Evolution Algorithm

Swagatam Das, Ajith Abraham, *Senior Member, IEEE*, and Amit Konar, *Member, IEEE*

***Abstract*—Differential evolution (DE) has emerged as one of the fast, robust, and efficient global search heuristics of current interest. This paper describes an application of DE to the automatic clustering of large unlabeled data sets. In contrast to most of the existing clustering techniques, the proposed algorithm requires no prior knowledge of the data to be classified. Rather, it determines the optimal number of partitions of the data "on the run." Superiority of the new method is demonstrated by comparing it with two recently developed partitional clustering techniques and one popular hierarchical clustering algorithm. The partitional clustering algorithms are based on two powerful well-known optimization algorithms, namely the genetic algorithm and the particle swarm optimization. An interesting real-world application of the proposed method to automatic segmentation of images is also reported.**

***Index Terms*—Differential evolution (DE), genetic algorithms (GAs), particle swarm optimization (PSO), partitional clustering.**

I. Introduction

CLUSTERING means the act of partitioning an unlabeled data set into groups of similar objects. Each group, called a "cluster," consists of objects that are similar between themselves and dissimilar to objects of other groups. In the past few decades, cluster analysis has played a central role in a variety of fields, ranging from engineering (e.g., machine learning, artificial intelligence, pattern recognition, mechanical engineering, and electrical engineering), computer sciences (e.g., web mining, spatial database analysis, textual document collection, and image segmentation), and life and medical sciences (e.g., genetics, biology, microbiology, paleontology, psychiatry, and pathology) to earth sciences (e.g., geography, geology, and remote sensing), social sciences (e.g., sociology, psychology, archeology, and education), and economics (e.g., marketing and business) [1]–[8].

Data clustering algorithms can be *hierarchical* or *partitional* [9], [10]. Within each of the types, there exists a wealth of subtypes and different algorithms for finding the clusters. In hierarchical clustering, the output is a tree showing a sequence of clustering, with each cluster being a partition of the data set [10]. Hierarchical algorithms can be agglomerative (bottom-up) or divisive (top-down). Agglomerative algorithms begin with each element as a separate cluster and merge them in successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. Hierarchical algorithms have two basic advantages [9]. First, the number of classes need not be specified *a priori*, and second, they are independent of the initial conditions. However, the main drawback of hierarchical clustering techniques is that they are static; that is, data points assigned to a cluster cannot move to another cluster. In addition to that, they may fail to separate overlapping clusters due to lack of information about the global shape or size of the clusters [11]. Partitional clustering algorithms, on the other hand, attempt to decompose the data set directly into a set of disjoint clusters. They try to optimize certain criteria (e.g., a square-error function, which is to be detailed in Section II). The criterion function may emphasize the local structure of the data, such as by assigning clusters to peaks in the probability density function, or the global structure. Typically, the global criteria involve minimizing some measure of dissimilarity in the samples within each cluster while maximizing the dissimilarity of different clusters. The advantages of the hierarchical algorithms are the disadvantages of the partitional algorithms, and vice versa. An extensive survey of various clustering techniques can be found in [11].

Clustering can also be performed in two different modes: 1) crisp and 2) fuzzy. In crisp clustering, the clusters are disjoint and nonoverlapping in nature. Any pattern may belong to one and only one class in this case. In fuzzy clustering, a pattern may belong to all the classes with a certain fuzzy membership grade [11]. The work described in this paper concerns crisp clustering algorithms only.

The problem of partitional clustering has been approached from diverse fields of knowledge, such as statistics (multivariate analysis) [12], graph theory [13], expectation–maximization algorithms [14], artificial neural networks [15]–[17], evolutionary computing [18], [19], and so on. Researchers all over the globe are coming up with new algorithms, on a regular basis, to meet the increasing complexity of vast real-world data sets. Thus, it seems well nigh impossible to review the huge and multifaceted literature on clustering in the scope of this paper. We here, instead, confine ourselves to the field of evolutionary partitional clustering, where this paper attempts to make a humble contribution. In the evolutionary approach, clustering of a data set is viewed as an optimization problem and solved by using an evolutionary search heuristic such as a genetic algorithm (GA) [20], which is inspired by Darwinian

Manuscript received April 13, 2006; revised September 23, 2006. The work of A. Abraham was supported by the Centre for Quantifiable Quality of Service in Communication Systems (Q2S), Centre of Excellence, which is appointed by the Research Council of Norway and funded by the Research Council, Norwegian University of Science and Technology (NTNU), and UNINETT. This paper was recommended by Associate Editor R. Subbu.

S. Das and A. Konar are with the Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700032, India (e-mail: swagatamdas19@yahoo.co.in; konaramit@yahoo.co.in).

A. Abraham is with the Q2S, Centre of Excellence, NTNU, 7491 Trondheim, Norway (e-mail: ajith.abraham@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2007.909595

evolution and genetics. The key idea is to create a population of candidate solutions to an optimization problem, which is iteratively refined by alteration and selection of good solutions for the next iteration. Candidate solutions are selected according to a fitness function, which evaluates their quality with respect to the optimization problem. In the case of GAs, the alteration consists of mutation to explore solutions in the local neighborhood of existing solutions and crossover to recombine information between different candidate solutions. An important advantage of these algorithms is their ability to cope with local optima by maintaining, recombining, and comparing several candidate solutions simultaneously. In contrast, local search heuristics, such as the simulated annealing algorithm [21], only refine a single candidate solution and are notoriously weak in coping with local optima. Deterministic local search, which is used in algorithms like the K-means (to be introduced in the next section) [12], [22], always converges to the nearest local optimum from the starting position of the search.

Tremendous research effort has gone in the past few years to evolve the clusters in complex data sets through evolutionary computing techniques. However, not much research work has been reported to determine the optimal number of clusters at the same time. Most of the existing clustering techniques, based on evolutionary algorithms, accept the number of classes K as an input instead of determining the same on the run. Nevertheless, in many practical situations, the appropriate number of groups in a previously unhandled data set may be unknown or impossible to determine even approximately. For example, while clustering a set of documents arising from the query to a search engine, the number of classes K changes for each set of documents that result from an interaction with the search engine. Also, if the data set is described by high-dimensional feature vectors (which is very often the case), it may be practically impossible to visualize the data for tracking its number of clusters.

The objective of this paper is twofold. First, it aims at the automatic determination of the optimal number of clusters in any unlabeled data set. Second, it attempts to show that differential evolution (DE) [23], with a modification of the chromosome representation scheme, can give very promising results if applied to the automatic clustering problem. DE is easy to implement and requires a negligible amount of parameter tuning to achieve considerably good search results. We modified the conventional DE algorithm from its classical form to improve its convergence properties. In addition to that, we used a novel representation scheme for the search variables to determine the optimal number of clusters. In this paper, we refer to the new algorithm as the automatic clustering DE (ACDE) algorithm.

At this point, we would like to mention that the traditional approach of determining the optimal number of clusters in a data set is using some specially devised statistical–mathematical function (also known as a clustering validity index) to judge the quality of partitioning for a range of cluster numbers. A good clustering validity index is generally expected to provide global minima/maxima at the exact number of classes in the data set. Nonetheless, determination of the optimum cluster number using global validity measures is very expensive since clustering has to be carried out for a variety of possible cluster numbers. In the proposed evolutionary learning framework, a number of trial solutions come up with different cluster numbers as well as cluster center coordinates for the same data set. Correctness of each possible grouping is quantitatively evaluated with a global validity index (e.g., the CS or Davis–Bouldin (DB) measure [35]). Then, through a mechanism of mutation and natural selection, eventually, the best solutions start dominating the population, whereas the bad ones are eliminated. Ultimately, the evolution of solutions comes to a halt (i.e., converges) when the fittest solution represents a near-optimal partitioning of the data set with respect to the employed validity index. In this way, the optimal number of classes along with the accurate cluster center coordinates can be located in one runt of the evolutionary optimization algorithm. A downside to the proposed method is that its performance depends heavily on the choice of a suitable clustering validity index. An inefficient validity index may result into many false clusters (due to the overfitting of data) even when the actual number of clusters in the given data set may be very much tractable. However, with a judicious choice of the validity index, the proposed algorithm can automate the entire process of clustering and yield near-optimal partitioning of any previously unhandled data set in a reasonable amount of time. This is certainly a very desirable feature of a real-life pattern recognition task.

We have extensively compared the ACDE with two other state-of-the-art automatic clustering techniques [24], [25] based on GA and particle swarm optimization (PSO) [26]. In addition, the quality of the final solutions has been compared with a standard agglomerative hierarchical clustering technique. The following performance metrics have been used in the comparative analysis: 1) the accuracy of final clustering results; 2) the speed of convergence; and 3) the robustness (i.e., ability to produce nearly same results over repeated runs). The test suit chosen for this paper consists of five real-life data sets. Finally, an interesting application of the proposed algorithm has been illustrated with reference to the automatic segmentation of a few well-known grayscale images.

The rest of this paper is organized as follows. Section II defines the clustering problem in a formal language and gives a brief overview of a previous work done in the field of evolutionary partitional clustering. Section III outlines the proposed ACDE algorithm. Section IV describes the five real data sets used for experiments, the simulation strategy, the algorithms used for comparison, and their parameter setup. Results of clustering over five real-life data sets and an application in image pixel classification are presented in Section V. Conclusions are provided in Section VI.

II. Scientific Backgrounds

A. Problem Definition

A *pattern* is a physical or abstract structure of objects. It is distinguished from others by a collective set of attributes called *features*, which together represent a pattern [27]. Let $P = \{P_1, P_2, \ldots, P_n\}$ be a set of n patterns or data points, each having d features. These patterns can also be represented by a profile data matrix $\mathbf{X}_{n\times d}$ with n d-dimensional row vectors. The ith row vector $\vec{X}_i$ characterizes the ith object from the set P, and each element $X_{i,j}$ in $\vec{X}_i$ corresponds to the

jth real-value feature $(j = 1, 2, \ldots, d)$ of the ith pattern $(i = 1, 2, \ldots, n)$. Given such an $\mathbf{X}_{n \times d}$ matrix, a partitional clustering algorithm tries to find a partition $C = \{C_1, C_2, \ldots, C_K\}$ of K classes, such that the similarity of the patterns in the same cluster is maximum and patterns from different clusters differ as far as possible. The partitions should maintain three properties.

1) Each cluster should have at least one pattern assigned, i.e., $C_i \neq \Phi \; \forall i \in \{1, 2, \ldots, K\}$.
2) Two different clusters should have no pattern in common, i.e., $C_i \cap C_j = \Phi \; \forall i \neq j$ and $i, j \in \{1, 2, \ldots, K\}$.
3) Each pattern should definitely be attached to a cluster i.e., $\bigcup_{i=1}^{K} C_i = P$.

Since the given data set can be partitioned in a number of ways, maintaining all of the aforementioned properties, a fitness function (some measure of the adequacy of the partitioning) must be defined. The problem then turns out to be one of finding a partition $\mathbf{C}^*$ of optimal or near-optimal adequacy, as compared to all other feasible solutions $\mathbf{C} = \{C^1, C^2, \ldots, C^{N(n,K)}\}$, where

$$N(n, K) = \frac{1}{K!} \sum_{i=1}^{K} (-1)^i \binom{K}{i}^i (K - i)^i \tag{1}$$

is the number of feasible partitions. This is the same as

$$\underset{C}{\text{Optimize}} \; f(X_{n \times d}, C) \tag{2}$$

where C is a single partition from the set $\mathbf{C}$, and f is a statistical–mathematical function that quantifies the goodness of a partition on the basis of the distance measure of the patterns (please see Section II-C). It has been shown in [28] that the clustering problem is NP-hard when the number of clusters exceeds 3.

B. Similarity Measures

As previously mentioned, clustering is the process of recognizing natural groupings or clusters in multidimensional data based on some similarity measures. Hence, defining an appropriate similarity measure plays a fundamental role in clustering [11]. The most popular way to evaluate similarity between two patterns amounts to the use of a *distance measure*. The most widely used distance measure is the Euclidean distance, which between any two d-dimensional patterns $\vec{X}_i$ and $\vec{X}_j$ is given by

$$d(\vec{X}_i, \vec{X}_j) = \sqrt{\sum_{p=1}^{d} (X_{i,p} - X_{j,p})^2} = \|\vec{X}_i - \vec{X}_j\|. \tag{3}$$

The Euclidean distance measure is a special case (when $\alpha = 2$) of the Minowsky metric [11], which is defined as

$$d^{\alpha}(\vec{X}_i, \vec{X}_j) = \left(\sum_{p=1}^{d} (X_{i,p} - X_{j,p})^{\alpha} \right)^{1/\alpha} = \|\vec{X}_i - \vec{X}_j\|^{\alpha}. \tag{4}$$

When $\alpha = 1$, the measure is known as the Manhattan distance [28].

The Minowsky metric is usually not efficient for clustering data of high dimensionality, as the distance between the patterns increases with the growth of dimensionality. Hence, the concepts of *near* and *far* become weaker [29]. Furthermore, according to Jain *et al.* [11], for the Minowsky metric, the large-scale features tend to dominate over the other features. This can be solved by normalizing the features over a common range. One way to do the same is by using the cosine distance (or vector dot product), which is defined as

$$\langle \vec{X}_i, \vec{X}_j \rangle = \frac{\sum_{p=1}^{d} X_{i,p} \cdot X_{j,p}}{\|\vec{X}_i\| \|\vec{X}_j\|}. \tag{5}$$

The cosine distance measures the angular difference of the two data vectors (patterns) and not the difference of their magnitudes. Another distance measure that needs mention in this context is the Mahalanabis distance, which is defined as

$$d_M(\vec{X}_i, \vec{X}_j) = (\vec{X}_i - \vec{X}_j) \Sigma^{-1} (\vec{X}_i - \vec{X}_j) \tag{6}$$

where Σ is the covariance matrix of the patterns. The Mahalanabis distance assigns different weights to different features based on their variances and pairwise linear correlations [11].

C. Clustering Validity Indexes

Cluster validity indexes correspond to the statistical–mathematical functions used to evaluate the results of a clustering algorithm on a quantitative basis. Generally, a cluster validity index serves two purposes. First, it can be used to determine the number of clusters, and second, it finds out the corresponding best partition. One traditional approach for determining the optimum number of classes is to repeatedly run the algorithm with a different number of classes as input and then to select the partitioning of the data resulting in the best validity measure [30]. Ideally, a validity index should take care of the two aspects of partitioning.

1) *Cohesion*: The patterns in one cluster should be as similar to each other as possible. The fitness variance of the patterns in a cluster is an indication of the cluster's cohesion or compactness.
2) *Separation*: Clusters should be well separated. The distance among the cluster centers (may be their Euclidean distance) gives an indication of cluster separation.

For crisp clustering, some of the well-known indexes available in the literature are the Dunn's index (DI) [31], the Calinski–Harabasz index [32], the DB index [33], the Pakhira Bandyopadhyay Maulik (PBM) index [34], and the CS measure [35]. All these indexes are optimizing in nature, i.e., the maximum or minimum values of these indexes indicate the appropriate partitions. Because of their optimizing character, the cluster validity indexes are best used in association with any optimization algorithm such as GA, PSO, etc. In what follows, we will discuss only two validity measures in detail, which have been employed in the study of our automatic clustering algorithm.

1) DB Index: This measure is a function of the ratio of the sum of within-cluster scatter to between-cluster separation, and it uses both the clusters and their sample means. First, we define the *within* ith *cluster scatter* and the *between* ith and jth *cluster distance*, respectively, i.e.,

$$S_{i,q} = \left[\frac{1}{N_i} \sum_{\vec{X} \in C_i} \|\vec{X} - \vec{m}_i\|_2^q \right]^{1/q} \tag{7}$$

$$d_{ij,t} = \left\{ \sum_{p=1}^{d} |m_{i,p} - m_{j,p}|^t \right\}^{1/t} = \|\vec{m}_i - \vec{m}_j\|_t \tag{8}$$

where $\vec{m}_i$ is the ith cluster center, q, $t \geq 1$, q is an integer, and q and t can be independently selected. N_i is the number of elements in the ith cluster C_i. Next, $R_{i,qt}$ is defined as

$$R_{i,qt} = \max_{j \in K, j \neq i} \left\{ \frac{S_{i,q} + S_{j,q}}{d_{ij,t}} \right\}. \tag{9}$$

Finally, we define the DB measure as

$$\text{DB}(K) = \frac{1}{K} \sum_{i=1}^{K} R_{i,qt}. \tag{10}$$

The smallest $\text{DB}(K)$ index indicates a valid optimal partition.

2) CS Measure: Recently, Chou *et al.* have proposed the CS measure [35] for evaluating the validity of a clustering scheme. Before applying the CS measure, the centroid of a cluster is computed by averaging the data vectors that belong to that cluster using

$$\vec{m}_i = \frac{1}{N_i} \sum_{x_j \in C_i} \vec{x}_j. \tag{11}$$

A distance metric between any two data points $\vec{X}_i$ and $\vec{X}_j$ is denoted by $d(\vec{X}_i, \vec{X}_j)$. Then, the CS measure can be defined as

$$\text{CS}(K) = \frac{\frac{1}{K} \sum_{i=1}^{K} \left[\frac{1}{N_i} \sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i} \left\{ d(\vec{X}_i, \vec{X}_q) \right\} \right]}{\frac{1}{K} \sum_{i=1}^{K} \left[\min_{j \in K, j \neq i} \{ d(\vec{m}_i, \vec{m}_j) \} \right]} = \frac{\sum_{i=1}^{K} \left[\frac{1}{N_i} \sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i} \left\{ d(\vec{X}_i, \vec{X}_q) \right\} \right]}{\sum_{i=1}^{K} \left[\min_{j \in K, j \neq i} \{ d(\vec{m}_i, \vec{m}_q) \} \right]}. \tag{12}$$

As can easily be perceived, this measure is a function of the ratio of the sum of within-cluster scatter to between-cluster separation and has the same basic rationale as the DI and DB measures. According to Chou *et al.*, the CS measure is more efficient in tackling clusters of different densities and/or sizes than the other popular validity measures, the price being paid in terms of high computational load with increasing K and n.

D. Brief Review of the Existing Works

The most widely used iterative K-means algorithm [22] for partitional clustering aims at minimizing the intracluster spread (ICS), which for K cluster centers can be defined as

$$\text{ICS}(C_1, C_2, \ldots, C_K) = \sum_{i=1}^{K} \sum_{\vec{X}_i \in C_i} \|\vec{X}_i - \vec{m}_i\|^2. \tag{13}$$

The K-means (or hard C-means) algorithm starts with K cluster centroids (these centroids are initially randomly selected or derived from some *a priori* information). Each pattern in the data set is then assigned to the closest cluster center. The centroids are updated by using the mean of the associated patterns. The process is repeated until some stopping criterion is met. The K-means has two main advantages [11].

1) It is very easy to implement.
2) The time complexity is only $O(n)$ (n being the number of data points), which makes it suitable for large data sets.

However, it suffers from three disadvantages.

1) The user has to specify in advance the number of classes.
2) The performance of the algorithm is data dependent.
3) The algorithm uses a greedy approach and is heavily dependent on the initial conditions. This often leads K-means to converge to suboptimal solutions.

The remaining paragraphs of this section provide a summary of the most important applications of evolutionary computing techniques to the partitional clustering problem.

The first application of GAs to clustering was introduced by Raghavan and Birchand [36], and it was the first approach of using a direct encoding of the object–cluster association. The idea in this approach is to use a genetic encoding that directly allocates n objects to K clusters, such that each candidate solution consists of n genes, each with an integer value in the interval $[1, K]$. For example, for $n = 5$ and $K = 3$, the encoding "11322" allocates the first and second objects to cluster 1, the third object to cluster 3, and the fourth and fifth objects to cluster 2; thus, the clusters $(\{1, 2\}, \{3\}, \{4, 5\})$ are identified. Based on this problem representation, the GA tries to find the optimal partition according to a fitness function that measures the partition goodness. It has been shown that such an algorithm outperforms K-means in the analysis of simulated and real data sets (e.g., [37]). However, the representation scheme has a major drawback because of its redundancy; for instance, "11322" and "22311" represent the same grouping solution $(\{1, 2\}, \{3\}, \{4, 4\})$. Falkenauer [18] tackled this problem in an elegant way: in addition to the encoding of n genes representing each object–cluster association, they represent the group labels as additional genes in the encoding and apply *ad hoc* evolutionary operators on them.

The second kind of GA approach to partitional clustering is to encode cluster-separating boundaries. Bandyopadhyay *et al.* [38] used GAs to determine hyperplanes as decision boundaries, which divide the attribute feature space to separate the clusters. For this, they encode the location and orientation of a set of hyperplanes with a gene representation of flexible length. Apart from minimizing the number of misclassified objects, their approach tries to minimize the number of required hyperplanes. The third way to use GAs in partitional clustering is to encode a representative variable (typically a centroid or medoid) and, optionally, a set of parameters to describe the extent and shape of the variance for each cluster. Srikanth *et al.* [39] proposed an approach that encodes the center, extend, and orientation of an ellipsoid for each cluster.

Some researchers introduced hybrid clustering algorithms, combining classical clustering techniques with GAs [40]. For example, Krishna and Murty [41] introduced a GA with direct encoding of object–cluster associations as in [39], but applied K-means to determine the quality of the GA candidate solutions. Kuo *et al.* [42] used adaptive resonance theory 2 (ART2) neural network to determine an initial solution and then applied genetic K-means algorithm to find the final solution for analyzing Web-browsing paths in electronic commerce. The proposed method was also compared with ART2, followed by K-means.

Finding an optimal number of clusters in a large data set is usually a challenging task. The problem has been investigated by several researchers [43], [44], but the outcome is still unsatisfactory [45]. Lee and Antonsson [46] used an evolutionary strategy (ES) [47]-based method to dynamically cluster a data set. The proposed ES implemented variable-length individuals to search for both centroids and optimal number of clusters. An approach to dynamically classify a data set using evolutionary programming [48] can be found in [49], where two fitness functions are simultaneously optimized: one gives the optimal number of clusters, whereas the other leads to a proper identification of each cluster's centroid. Bandyopadhyay *et al.* [24] devised a variable string-length genetic algorithm to tackle the dynamic clustering problem using a single fitness function.

Recently, researchers working in this area have started taking some interest on two promising approaches to numerical optimization, namely the PSO and the DE. Paterlinia and Krink [50] used a DE algorithm and compared its performance with a PSO and a GA algorithm over the partitional clustering problem. Their work is focused on nonautomatic clustering with a preassigned number of clusters. In [51], Omran *et al.* proposed an image segmentation algorithm based on the PSO. The algorithm finds the centroids of a user-specified number of clusters, where each cluster groups together the similar pixels. They used a crisp criterion function for evaluating the partitions on the image data. Very recently, the same authors have come up with another automatic hard clustering scheme [25]. The algorithm starts by partitioning the data set into a relatively large number of clusters to reduce the effect of the initialization. Using a binary PSO [52], an optimal number of clusters is selected. Finally, the centroids of the chosen clusters are refined through the K-means algorithm. The authors applied the algorithm for segmentation of natural, synthetic, and multispectral images. Omran *et al.* also devised a nonautomatic crisp clustering scheme based on DE and illustrated the application of the algorithm to image segmentation problems in [53]. However, to the best of our knowledge, DE has not been applied to the *automatic* clustering of large real-life data sets as well as image pixels until date.

III. DE-Based Automatic Clustering

A. *Classical DE Algorithm and Its Modification*

The classical DE [23] is a population-based global optimization algorithm that uses a floating-point (real-coded) representation. The ith individual vector (chromosome) of the population at time-step (generation) t has d components (dimensions), i.e.,

$$\vec{Z}_i(t) = [Z_{i,1}(t), Z_{i,2}(t), \ldots, Z_{i,d}(t)]\,. \tag{14}$$

For each individual vector $\vec{Z}_k(t)$ that belongs to the current population, DE randomly samples three other individuals, i.e., $\vec{Z}_i(t)$, $\vec{Z}_j(t)$, and $\vec{Z}_m(t)$, from the same generation (for distinct k, i, j, and m). It then calculates the (componentwise) difference of $\vec{Z}_i(t)$ and $\vec{Z}_j(t)$, scales it by a scalar F (usually $\in$ [0, 1]), and creates a trial offspring $\vec{U}_i(t+1)$ by adding the result to $\vec{Z}_m(t)$. Thus, for the nth component of each vector

$$U_{k,n}(t+1) = \begin{cases} Z_{m,n}(t) + F(Z_{i,n}(t) - Z_{j,n}(t))\,, & \text{if } rand_n(0,1) < \mathrm{Cr} \\ Z_{k,n}(t), & \text{otherwise.} \end{cases} \tag{15}$$

$\mathrm{Cr} \in [0,1]$ is a scalar parameter of the algorithm, called the *crossover rate*. If the new offspring yields a better value of the objective function, it replaces its parent in the next generation; otherwise, the parent is retained in the population, i.e.,

$$\vec{Z}_i(t+1) = \begin{cases} \vec{U}_i(t+1), & \text{if } f\left(\vec{U}_i(t+1)\right) > f\left(\vec{Z}_i(t)\right) \\ \vec{Z}_i(t), & \text{if } \left(\vec{U}_i(t+1)\right) \le f\left(\vec{Z}_i(t)\right) \end{cases} \tag{16}$$

where $f(\cdot)$ is the objective function to be maximized.

To improve the convergence properties of DE, we have tuned its parameters in two different ways here. In the original DE, the difference vector $(\vec{Z}_i(t) - \vec{Z}_j(t))$ is scaled by a constant factor F. The usual choice for this control parameter is a number between 0.4 and 1. We propose to vary this scale factor in a random manner in the range (0.5, 1) by using the relation

$$F = 0.5 * (1 + rand(0,1)) \tag{17}$$

where $rand(0,1)$ is a uniformly distributed random number within the range [0, 1]. The mean value of the scale factor is 0.75. This allows for stochastic variations in the amplification of the difference vector and thus helps retain population diversity as the search progresses. In [54], we have already shown that the DE with random scale factor (DERANDSF) can meet or beat the classical DE and also some versions of the PSO in a statistically significant manner. In addition to that, here, we also linearly decrease the crossover rate Cr with time from $\mathrm{Cr}_{\max} = 1.0$ to $\mathrm{Cr}_{\min} = 0.5$. If $\mathrm{Cr} = 1.0$, it means that all components of the parent vector are replaced by the difference vector operator according to (12). However, at the later stages of the optimizing process, if Cr is decreased, more components of the parent vector are then inherited by the offspring. Such a tuning of Cr helps exhaustively explore the search space at

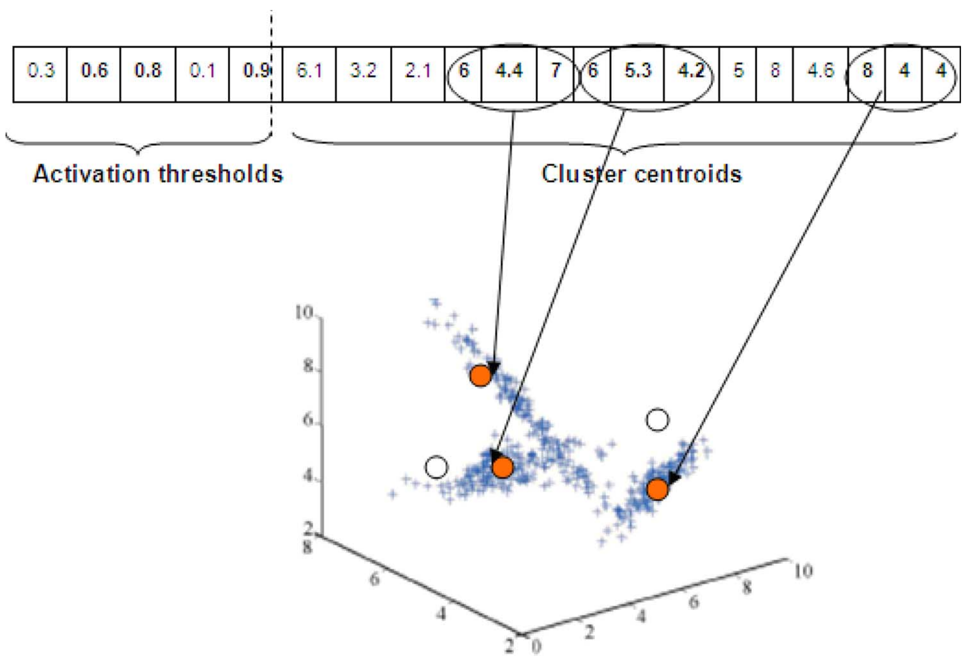


Fig. 1. Chromosome encoding scheme in the proposed method. A total of five cluster centers have been encoded for a 3-D data set. Only the activated cluster centers have been shown as orange circles.

the beginning but finely adjust the movements of trial solutions during the later stages of search, so that they can explore the interior of a relatively small space in which the suspected global optimum lies. The time variation of Cr may be expressed in the form of the following equation:

$$\mathrm{Cr} = (\mathrm{Cr}_{\max} - \mathrm{Cr}_{\min}) * (\mathrm{MAXIT} - \mathrm{iter})/\mathrm{MAXIT} \qquad (18)$$

where $\mathrm{Cr}_{\max}$ and $\mathrm{Cr}_{\min}$ are the maximum and minimum values of crossover rate Cr, respectively; iter is the current iteration number; and MAXIT is the maximum number of allowable iterations.

B. Chromosome Representation

In the proposed method, for n data points, each d dimensional, and for a user-specified maximum number of clusters $K_{\max}$, a chromosome is a vector of real numbers of dimension $K_{\max} + K_{\max} \times d$. The first $K_{\max}$ entries are positive floating-point numbers in [0, 1], each of which controls whether the corresponding cluster is to be activated (i.e., to be really used for classifying the data) or not. The remaining entries are reserved for $K_{\max}$ cluster centers, each d dimensional. For example, the velocity vector $\vec{V}_i(t)$ of the ith chromosome is shown in the equation at the bottom of the page.

The jth cluster center in the ith chromosome is active or selected for partitioning the associated data set if $T_{i,j} > 0.5$. On the other hand, if $T_{i,j} < 0.5$, the particular jth cluster is inactive in the ith chromosome. Thus, the $T_{i,j}$'s behave like control genes (we call them *activation thresholds*) in the chromosome governing the selection of the active cluster centers. The rule for selecting the actual number of clusters specified by one chromosome is

$$\begin{aligned}&\textbf{IF } T_{i,j} > 0.5, \textbf{ THEN} \text{ the } j\text{th cluster center}\\ &\quad \vec{m}_{i,j} \text{ is } \textbf{ACTIVE}\\ &\textbf{ELSE } \vec{m}_{i,j} \text{ is } \textbf{INACTIVE}. \qquad (19)\end{aligned}$$

As an example, consider the chromosome encoding scheme in Fig. 1. There are at most five 3-D cluster centers, among which, according to the rule presented in (19), the second (6, 4.4, 7), third (5.3, 4.2, 5), and fifth ones (8, 4, 4) have been activated for partitioning the data set. The quality of the partition yielded by such a chromosome can be judged by an appropriate cluster validity index.

When a new offspring chromosome is created according to (15) and (16), at first, the T values are used to select [using (19)] the active cluster centroids. If due to mutation some threshold $T_{i,j}$ in an offspring exceeds 1 or becomes negative, it is forcefully fixed to 1 or 0, respectively. However, if it is found that no flag could be set to 1 in a chromosome (all activation thresholds are smaller than 0.5), we randomly select two thresholds and reinitialize them to a random value between 0.5 and 1.0. Thus, the minimum number of possible clusters is 2.

$$\vec{V}_i(t) = \underbrace{\boxed{T_{i,1}}\boxed{T_{i,2}}\boxed{\ldots}\boxed{T_i, K_{\max}}}_{\textbf{Activation Threshhold}}\underbrace{\boxed{\vec{m}_{i,1}}\boxed{\vec{m}_{i,2}}\boxed{\ldots}\boxed{\vec{m}_{i,K_{\max}}}}_{\textbf{Cluster Centroids}}$$

C. Fitness Function

One advantage of the ACDE algorithm is that it can use any suitable validity index as its fitness function. Here, we conducted two different sets of experiments with two different fitness functions. These two functions are built on two clustering validity measures, namely the CS measure and the DB measure (refer to Sections II-C1 and C2). The CS-measure-based fitness functions can be described as

$$f_1 = \frac{1}{\mathrm{CS}_i(K) + \mathrm{eps}}. \tag{20}$$

Similarly, we may express the DB-index-based fitness function as

$$f_2 = \frac{1}{\mathrm{DB}_i(K) + \mathrm{eps}} \tag{21}$$

where DB_i is the DB index, which is evaluated on the partitions yielded by the ith chromosome (or the ith particle for PSO), and eps is the same as before.

D. Avoiding Erroneous Chromosomes

There is a possibility that, in our scheme, during computation of the CS and/or DB measures, a division by zero may be encountered. This may occur when one of the selected cluster centers is outside the boundary of distributions of the data set. To avoid this problem, we first check to see if any cluster has fewer than two data points in it. If so, the cluster center positions of this special chromosome are reinitialized by an average computation. We put n/K data points for every individual cluster center, such that a data point goes with a center that is nearest to it.

E. Pseudocode of the ACDE Algorithm

The pseudocode for the complete ACDE algorithm is given here.

```
Step 1) Initialize each chromosome to contain $K$ number of randomly selected cluster centers and $K$ (randomly chosen) activation thresholds in [0, 1].
Step 2) Find out the active cluster centers in each chromosome with the help of the rule described in (19).
Step 3) For $t = 1$ to $t_{\max}$ do
    a) For each data vector $\vec{X}_p$, calculate its distance metric $d(\vec{X}_p, \vec{m}_{i,j})$ from all active cluster centers of the $i$th chromosome $\vec{V}_i$.
    b) Assign $\vec{X}_p$ to that particular cluster center $\vec{m}_{i,j}$, where
       $d(\vec{X}_p, \vec{m}_{i,j}) = \min_{\forall b \in \{1,2,\ldots,K\}} \left\{ d(\vec{X}_p, \vec{m}_{i,b}) \right\}.$
    c) Check if the number of data points that belong to any cluster center $m_{i,j}$ is less than 2. If so, update the cluster centers of the chromosome using the concept of average described earlier.
    d) Change the population members according to the DE algorithm outlined in (15)–(18). Use the fitness of the chromosomes to guide the evolution of the population.
Step 4) Report as the final solution the cluster centers and the partition obtained by the globally best chromosome (one yielding the highest value of the fitness function) at time $t = t_{\max}$.
```

IV. Experiments and Results for the Real-Life Data Sets

In this section, we compare performance of the ACDE algorithm with two recently developed partitional clustering algorithms and one standard hierarchical agglomerative clustering based on the linkage metric of average link [55]. The former two algorithms are well known as the genetic clustering with an unknown number of clusters K (GCUK) [24] and the dynamic clustering PSO (DCPSO) [25]. Moreover, to investigate the effects of the changes made in the classical DE algorithm, we have compared the ACDE with an ordinary DE-based clustering method, which uses the same chromosome representation scheme and fitness function as the ACDE. The classical DE scheme that we have used is referred in the literature as the DE/rand/1/bin [23], where "bin" stands for the binomial crossover method.

A. Data Sets Used

The following real-life data sets [56], [57] are used in this paper. Here, n is the number of data points, d is the number of features, and K is the number of clusters.

1) Iris plants database ($n = 150$, $d = 4$, $K = 3$): This is a well-known database with 4 inputs, 3 classes, and 150 data vectors. The data set consists of three different species of iris flower: *Iris setosa*, *Iris virginica*, and *Iris versicolour*. For each species, 50 samples with four features each (sepal length, sepal width, petal length, and petal width) were collected. The number of objects that belong to each cluster is 50.
2) Glass ($n = 214$, $d = 9$, $K = 6$): The data were sampled from six different types of glass: 1) building windows float processed (70 objects); 2) building windows nonfloat processed (76 objects); 3) vehicle windows float processed (17 objects); 4) containers (13 objects); 5) tableware (9 objects); and 6) headlamps (29 objects). Each type has nine features: 1) refractive index; 2) sodium; 3) magnesium; 4) aluminum; 5) silicon; 6) potassium; 7) calcium; 8) barium; and 9) iron.
3) Wisconsin breast cancer data set ($n = 683$, $d = 9$, $K = 2$): The Wisconsin breast cancer database contains nine relevant features: 1) clump thickness; 2) cell size uniformity; 3) cell shape uniformity; 4) marginal adhesion; 5) single epithelial cell size; 6) bare nuclei; 7) bland chromatin; 8) normal nucleoli; and 9) mitoses. The data set has two classes. The objective is to classify each data vector into benign (239 objects) or malignant tumors (444 objects).
4) Wine ($n = 178$, $d = 13$, $K = 3$): This is a classification problem with "well-behaved" class structures. There are 13 features, three classes, and 178 data vectors.

TABLE I
PARAMETERS FOR THE CLUSTERING ALGORITHMS

GCUK		DCPSO		ACDE		Classical DE	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Pop_size	50	Pop_size	100	Pop_size	10*dim	Pop_size	10*dim
Cross-over Probability μ_c	0.8	Inertia Weight	0.72	CR_{max}	1.0	CR	0.9
Mutation probability μ_m	0.001	C_1, C_2	1.494				
		P_{ini}	0.75	CR_{min}	0.5	F	0.8
K_{max} K_{min}	20 2	K_{max} K_{min}	20 2	K_{max} K_{min}	20 2	K_{max} K_{min}	20 2

TABLE II
FINAL SOLUTION (MEAN AND STANDARD DEVIATION OVER 40 INDEPENDENT RUNS) AFTER EACH ALGORITHM WAS TERMINATED AFTER RUNNING FOR 10^6 FEs WITH THE CS-MEASURE-BASED FITNESS FUNCTION

Dataset	Algorithm	Avg no. of clusters found	CS measure	Mean Intra cluster Distance	Mean Inter cluster Distance
Iris	ACDE	**3.25±0.0382**	**0.6643±0.097**	**3.1164±0.033**	**2.5931±0.027**
	DCPSO	2.23±0.0443	0.7361±0.671	3.6516±1.195	2.2104±0.773
	GCUK	2.35±0.0985	0.7282±2.003	3.5673±2.792	2.5058±1.409
	Classical DE	2.50±0.0473	0.7633±0.039	3.9439±1.874	2.1158±1.089
	Average-link	3.00	0.7863±0.00	3.9808±0.00	2.0817±0.00
Wine	ACDE	**3.25±0.0391**	**0.9249±0.032**	**4.046±0.002**	**3.1483±0.078**
	PSO	3.05±0.0352	1.8721±0.037	4.851±0.184	2.6113±1.637
	GCUK	2.95±0.0112	1.5842±0.328	4.163±1.929	2.8058±1.365
	Classical DE	3.50±0.0143	1.7964±0.802	4.949±1.232	2.6118±1.384
	Average-link	3.00	1.8921±0.00	4.982±0.00	2.5009±0.00
Breast-Cancer	ACDE	**2.00±0.00**	**0.4532±0.034**	**4.2439±0.143**	**3.2577±0.138**
	DCPSO	2.25±0.0632	0.4854±0.009	4.8511±0.373	2.3613±0.021
	GCUK	2.00±0.0083	0.6089±0.016	4.9944±0.904	2.3944±1.744
	Classical DE	2.25±0.0261	0.8984±0.381	4.6944±0.654	2.9635±1.464
	Average-link	2.00	0.9007±0.00	5.0098±0.00	2.2817±0.00
Vowel	ACDE	**5.75±0.0751**	**0.9089±0.051**	**1412.63±0.792**	**2724.85±0.124**
	DCPSO	7.25±0.0183	1.1827±0.431	1482.51±3.973	1923.93±1.154
	GCUK	5.05±0.0075	1.9978±0.966	1495.13±12.334	1944.38±0.747
	Classical DE	7.50±0.0569	1.0844±0.067	1493.72±10.833	2434.45±1.213
	Average-link	6.00	1.7221±0.00	1493.98±0.00	2357.62±0.00
Glass	ACDE	**6.05±0.0148**	**0.3324±0.487**	**563.247±134.2**	**853.62±9.044**
	DCPSO	5.95±0.0093	0.7642±0.073	599.535±10.34	889.32±4.233
	GCUK	5.85±0.0346	1.4743±0.236	594.673±30.62	869.93±1.789
	Classical DE	5.60±0.0754	0.7782±0.643	608.837±20.92	891.82±4.945
	Average-link	6.00	1.0221±0.00	610.033±0.00	895.47±0.00

5) Vowel data set ($n = 871$, $d = 3$, $K = 6$): This data set consists of 871 Indian Telugu vowel sounds. The data set has three features, namely F_1, F_2, and F_3, corresponding to the first, second and, third vowel frequencies, and six overlapping classes {d (72 objects), a (89 objects), i (172 objects), u (151 objects), e (207 objects), o (180 objects)}.

B. Population Initialization

For the ACDE algorithm, we randomly initialize the activation thresholds (control genes) within [0, 1]. The cluster centroids are also randomly fixed between $X_{\max}$ and $X_{\min}$, which denote the maximum and minimum numerical values of any feature of the data set under test, respectively. For example, in the case of the grayscale images (discussed in Section IV-F), since the intensity value of each pixel serves as a feature, we choose $X_{\min} = 0$ and $X_{\max} = 255$. To make the comparison fair, the populations for both the ACDE and the classical DE-based clustering algorithms (for all problems tested) were initialized using the same random seeds. For the GCUK, each string in the population initially encodes the centers of K_i clusters, where $K_i = rand() \cdot K_{\max}$. Here, $K_{\max}$ is a soft estimate of the upper bound of the number of clusters. The K_i centers encoded in the chromosome are randomly selected points from the data set. In the case of the DCPSO algorithm, the initial position of the ith particle $\vec{Z}_i(0)$ (for a binary PSO) is fixed depending on a user-specified probability P_{ini}, as follows:

$$Z_{i,k}(0) = \begin{cases} 0, & \text{if } r_k \le p_{\text{ini}} \\ 1, & \text{if } r_k < p_{\text{ini}} \end{cases}$$

TABLE III
MEAN CLASSIFICATION ERROR OVER NOMINAL PARTITION AND STANDARD DEVIATION OVER 40 INDEPENDENT RUNS, WHERE EACH RUN WAS CONTINUED UP TO 10^6 FEs FOR THE FIRST FOUR EVOLUTIONARY ALGORITHMS (USING THE CS MEASURE)

Dataset	Mean Classification Error				
	ACDE	DCPSO	GCUK	Classical DE	Average-link
Iris	**2.35±0.00**	4.15±0.0	5.00±0.00	3.96±0.00	4.00±0.00
Wine	**36.65±0.0**	99.4±1.09	100.24±1.05	114.50±1.53	134.00±0.00
Breast Cancer	**22.25±0.28**	27.01±1.25	29.00±1.55	29.15±0.50	26.00±0.00
Vowel	**418.75±3.10**	453.58±6.61	476.42±6.92	473.72±4.25	496.00±0.00
Glass	**92.55±0.19**	102.1±0.68	98.21±0.08	105.36±0.54	111.00±0.00

TABLE IV
RESULTS OF THE UNPAIRED t-TEST BETWEEN THE BEST AND THE SECOND BEST PERFORMING ALGORITHMS (FOR EACH DATA SET) BASED ON THE CS MEASURES OF TABLE II

Dataset	Standard Error	t	95% Confidence Interval	Two-tailed P	Significance
Iris	0.095	4.0595	-0.142225 to -0.047575	0.0002	**Extremely significant**
Wine	0.074	4.0615	-0.448480 to -0.150120	0.0002	**Extremely significant**
Breast Cancer	0.008	34.8354	-0.309710 to -0.275690	< 0.0001	**Extremely significant**
Vowel	0.019	6.1344	-0.153616 to -0.077384	< 0.0001	**Extremely Significant**
Glass	3.9214	0.110	-0.654712 to -0.208888	< 0.0001	**Extremely Significant**

TABLE V
MEAN AND STANDARD DEVIATIONS OF THE NUMBER OF FITNESS FEs (OVER 40 INDEPENDENT RUNS) REQUIRED BY EACH ALGORITHM TO REACH A PREDEFINED CUTOFF VALUE OF THE CS VALIDITY INDEX

Dataset	Algorithm	Mean no. of function evaluations required	CS Cutoff Value	Mean Intra cluster Distance	Mean Inter cluster Distance
Iris	ACDE	**459888.95±20.50**	0.95	**3.7836±0.509**	**2.0758±0.239**
	DCPSO	679023.85±31.75		3.9637±1.666	2.0093±0.795
	GCUK	707723.70±120.21		3.9992±2.390	1.9243±1.843
	Classical DE	698043.80±93.36		4.5793±2.454	1.9564±1.767
Wine	ACDE	**67384..25**±56.45	1.90	**4.9872±0.148**	**3.1275±0.0357**
	PSO	700473.35±31.42		4.0743±0.093	1.9967±1.828
	GCUK	785333.05±21.75		5.9870±1.349	2.1323±1.334
	Classical DE	675472.95±14.83		4.9927±1.236	2.6842±1.356
Breast-Cancer	ACDE	**292102.50**±29.73	1.10	**4.9744±0.105**	**3.0096±0.246**
	DCPSO	587832.50±7.34		5.6546±0.241	2.1173±0.452
	GCUK	914033.85±24.83		8.0442±0.435	2.0542±1.664
	Classical DE	575484.70±10.26		5.3407±0.652	2.2315±2.885
Vowel	ACDE	**437533.35**±51.73	2.50	**1494.12±0.378**	**2739.85±0.163**
	DCPSO	500493.15±35.47		1575.51±3.786	1923.93±1.154
	GCUK	498354.10±74.60		1593.72±1.789	2633.45±1.213
	Classical DE	667342.80±53.54		1674.13±6.564	1947.38±0.747
Glass	ACDE	**443233.30**±74.65	1.80	**590.572±34.24**	**853.62±0.44**
	DCPSO	566335.80±25.73		619.980±15.98	846.67±0.804
	GCUK	574938.65±82.64		615.88±20.95	857.34±1.465
	Classical DE	542355.95±32.85		608.85±30.62	829.07±1.765

where r_k is a uniformly distributed random number in [0, 1]. The initial velocity vector of each particle $\vec{V}_i(0)$ is randomly set in the interval $[-5, 5]$ following [25].

C. Parameter Setup for the Compared Algorithms

We used the best possible parameter settings recommended in [24] and [25] for the GCUK and DCPSO algorithms,

TABLE VI
MEAN CLASSIFICATION ERROR OVER NOMINAL PARTITION AND STANDARD DEVIATION OVER 40 INDEPENDENT RUNS, WHICH WERE STOPPED AS SOON AS THEY REACHED THE PREDEFINED CUTOFF CS VALUE

Dataset	Mean Classification Error			
	ACDE	DCPSO	GCUK	Classical DE
Iris	**8.68±0.34**	14.34±0.45	9.34±0.57	9.34±0.00
Wine	**100.32±0.79**	105.23±2.59	139.3±6.34	106.68±1.45
Breast Cancer	**37.50±0.33**	49.67±0.34	43.34±0.69	40.08±1.35
Vowel	**468.00±4.79**	492.34±0.64	491.40±4.25	496.44±1.38
Glass	**98.57±0.77**	102.1±0.68	105.36±0.54	98.21±0.08

TABLE VII
FINAL SOLUTION (MEAN AND STANDARD DEVIATION OVER 40 INDEPENDENT RUNS) WHEN EACH ALGORITHM WAS TERMINATED AFTER RUNNING FOR 10_6 FEs WITH THE DB-MEASURE-BASED FITNESS FUNCTION

Problem	Algorithm	Avg no. of clusters found	DB measure	Mean Intra cluster Distance	Mean Inter cluster Distance
Iris	ACDE	**3.05±0.0712**	**0.4645±0.022**	**3.1633±0.076**	**2.8387±0.658**
	DCPSO	2.25±0.0593	0.6899±0.008	3.8536±0.122	2.2544±0.039
	GCUK	2.30±0.0738	0.7377±0.065	3.8436±0.076	2.1438±0.022
	Classical DE	2.50±0.0092	0.5822±0.067	3.8876±0.092	2.0358±0.055
	Average-link	3.00	0.8471±0.00	3.9098±0.00	2.2817±0.00
Wine	ACDE	**3.25±0.0931**	**3.0432±0.021**	**4.4212±0.096**	**3.1029±0.047**
	DCPSO	3.05±0.0024	4.3432±0.232	4.8668±0.154	2.6113±1.635
	GCUK	2.95±0.0173	5.3424±0.343	5.1312±1.342	2.7565±2.128
	Classical DE	3.50±0.0143	3.3923±0.092	4.263±1.907	2.8158±1.786
	Average-link	3.00	5.7206±0.00	4.982±0.00	2.5009±0.00
Breast-Cancer	ACDE	2.05±0.0563	0.5203±0.006	4.5463±0.023	3.1002±0.064
	DCPSO	2.50±0.0621	0.5754±0.073	4.9232±0.373	2.2684±0.063
	GCUK	2.50±0.0352	0.6328±0.002	6.5541±0.433	1.8032±0.016
	Classical DE	**2.10±0.0081**	**0.5199±0.007**	**5.2234±0.042**	**2.0236±0.058**
	Average-link	2.00	0.7634±0.00	5.0098±0.00	2.2817±0.00
Vowel	ACDE	**5.75±0.0241**	**0.9224±0.334**	**1449.12±0.834**	**2289.85±0.163**
	DCPSO	7.25±0.0562	1.2821±0.009	1500.57±3.748	1747.76±1.764
	GCUK	5.05±0.0561	2.9482±0.028	1573.23±4.675	2271.89±1.222
	Classical DE	7.50±0.0819	1.4488±0.075	1498.78±2.725	1962.31±0.993
	Average-link	6.00	3.0581±0.00	1493.98±0.00	2357.62±0.00
Glass	ACDE	**6.05±0.0248**	**1.0092±0.083**	**501.757±4.3**	**893.46±3.32**
	DCPSO	5.95±0.0193	1.5152±0.073	514.554±9.5	856.00±8.07
	GCUK	5.85±0.0346	1.8371±0.034	518.903±2.9	852.32±5.43
	Classical DE	5.60±0.0446	1.6673±0.004	514.849±3.4	862.21±2.53
	Average-link	6.00	1.8519±0.00	610.033±0.00	895.47±0.00

TABLE VIII
MEAN CLASSIFICATION ERROR OVER NOMINAL PARTITION AND STANDARD DEVIATION OVER 40 INDEPENDENT RUNS, WHERE EACH RUN WAS CONTINUED UP TO 10^6 FEs FOR THE FIRST FOUR EVOLUTIONARY ALGORITHMS (USING THE DB MEASURE)

Dataset	Mean Classification Error				
	ACDE	DCPSO	GCUK	Classical DE	Average-link
Iris	**2.22±0.00**	2.79±0.55	2.75±0.08	3.14±0.00	4.00±0.00
Wine	**40.15±0.0**	112.5±2.50	118.45±1.77	102.22±1.05	134.00±0.00
Breast Cancer	26.75±0.25	30.23±0.46	26.50±0.80	**25.00±1.09**	26.00±0.00
Vowel	**418.35±7.50**	435.00±3.75	473.46±3.57	475.65±2.67	496.00±0.00
Glass	**8.86±0.42**	14.35±0.26	17.98±0.67	15.69±0.85	111.00±0.00

respectively. For the ACDE algorithm, we choose an optimal set of parameters after experimenting with many possibilities. Table I summarizes these settings. In Table I, Pop_size indicates the size of the population, dim implies the dimension of each chromosome, and P_{ini} is a user-specified probability used for initializing the position of a particle in the DCPSO algorithm. For details on this issue, please refer to [25]. Once set, we allow no hand tuning of the parameters to make the comparison fair enough.

D. Simulation Strategy

In this paper, while comparing the performance of our ACDE algorithm with other state-of-the-art clustering techniques, we

TABLE IX
RESULTS OF THE UNPAIRED *t*-TEST BETWEEN THE BEST AND THE SECOND BEST PERFORMING ALGORITHMS (FOR EACH DATA SET) BASED ON THE DB MEASURES OF TABLE VII

Dataset	Standard Error	*t*	95% Confidence Interval	Two-tailed *P*	Significance
Iris	0.011	10.5559	-0.139898 to -0.095502	< 0.0001	**Extremely significant**
Wine	0.015	23.3971	-0.378805 to -0.319395	< 0.0001	**Extremely significant**
Breast Cancer	0.001	0.2744	-0.003302 to 0.002502	0.7845	Not significant
Vowel	0.053	6.8087	-0.464875 to -0.254525	< 0.0001	**Extremely Significant**
Glass	0.017	28.9521	-0.540794 to -0.471206	< 0.0001	**Extremely Significant**

TABLE X
MEAN AND STANDARD DEVIATIONS OF THE NUMBER OF FITNESS FEs (OVER 40 INDEPENDENT RUNS) REQUIRED BY EACH ALGORITHM TO REACH A PREDEFINED CUTOFF VALUE OF THE DB VALIDITY INDEX

Dataset	Algorithm	Mean no. of function evaluations required	DB Cutoff Value	Mean Intra cluster Distance	Mean Inter cluster Distance
Iris	ACDE	**504783.45**±12.65	0.80	**3.9928±0.029**	**2.1029±0.842**
	DCPSO	679084.75±16.57		3.7852±1.842	1.7641±0.439
	GCUK	790865.90±10.21		4.4587±3.782	1.9383±1.307
	Classical DE	658796.35±17.28		4.0393±1.542	1.6278±1.681
Wine	ACDE	**464653.35**±5.50	6.00	**4.8292±0.732**	**3.0219±0.069**
	DCPSO	486885.85±2.85		5.1472±0.472	2.1161±1.623
	GCUK	598743.35±8.09		4.9383±1.722	2.9121±0.353
	Classical DE	477869.95±8.12		4.7531±2.043	2.8158±0.389
Breast-Cancer	ACDE	424732.30±8.93	0.90	5.4489±0.342	3.0234±0.683
	DCPSO	467854.60±10.12		5.2885±0.552	2.0124±1.596
	GCUK	678874.90±7.82		6.8832±0.733	2.1637±1.458
	Classical DE	**418765.55±1.23**		**5.8684±0.467**	**1.9235±0.164**
Vowel	ACDE	**435743.05±2.65**	3.00	**1544.92±0.834**	**2081.31±0.679**
	DCPSO	556865.00±4.26		1652.58±2.341	1264.87±3.069
	GCUK	575854.65±1.29		1582.55±7.332	1989.38±7.734
	Classical DE	546859.60±2.05		1608.22±5.866	1604.43±1.674
Glass	ACDE	**506754.00**±12.27	2.00	**132.757±15.8**	**13.46±2.54**
	DCPSO	569787.95±10.83		154.564±39.6	13.56±2.65
	GCUK	687678.75±10.97		155.856±24.7	10.42±4.69
	Classical DE	527585.35±7.50		178.809±30.3	10.21±1.09

TABLE XI
MEAN CLASSIFICATION ERROR OVER NOMINAL PARTITION AND STANDARD DEVIATION OVER 40 INDEPENDENT RUNS, WHICH WERE STOPPED AS SOON AS THEY REACHED THE PREDEFINED CUTOFF DB VALUE

Dataset	Mean Classification Error			
	ACDE	DCPSO	GCUK	Classical DE
Iris	**2.22±0.00**	2.79±0.55	2.75±0.08	3.14±0.00
Wine	**40.15±0.0**	112.5±2.50	118.45±1.77	102.22±1.05
Breast Cancer	**26.75±0.25**	30.23±0.46	26.50±0.80	29.00±1.09
Vowel	**418.35±7.50**	435.00±3.75	473.46±3.57	475.65±2.67
Glass	**8.86±0.42**	14.35±0.26	17.98±0.67	15.69±0.85

focus on three major issues: 1) quality of the solution as determined by the CS and DB measures; 2) ability to find the optimal number of clusters; and 3) computational time required to find the solution.

For comparing the speed of the stochastic algorithms such as GA, PSO, or DE, the first thing we require is a fair time measurement. The number of iterations or generations cannot be accepted as a time measure since the algorithms perform different amount of works in their inner loops, and they have different population sizes. Hence, we choose the *number of fitness function evaluations (FEs)* as a measure of computation time instead of generations or iterations.

Since four of the other algorithms used for comparison are stochastic in nature, the results of two successive runs usually

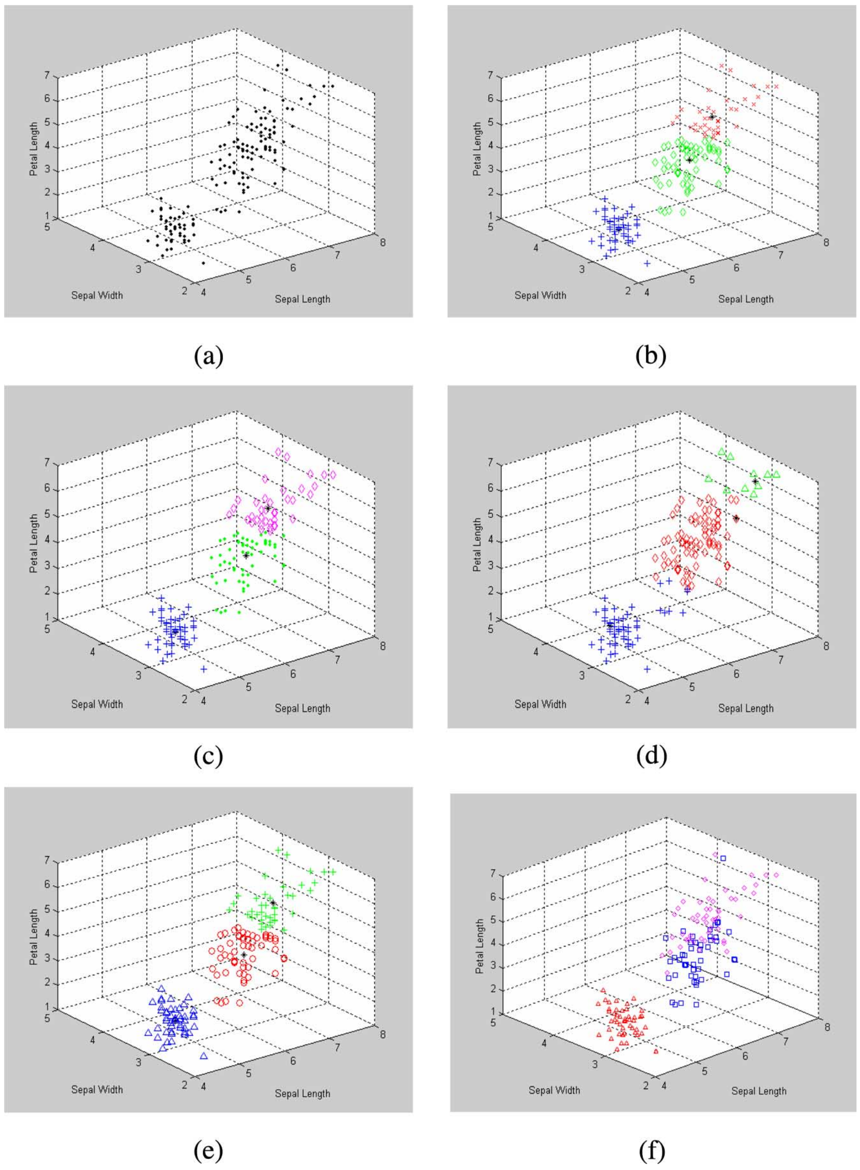


Fig. 2. (a) Three-dimensional plot of the unlabeled iris data set using the first three features. Clustering of iris data by (b) ACDE, (c) DCPSO, (d) GCUK, (e) classical DE, and (f) average-link-based hierarchical clustering algorithm.

do not match for them. Hence, we have taken 40 independent runs (with different seeds of the random number generator) of each algorithm. The results have been stated in terms of the mean values and standard deviations over the 40 runs in each case. As the hierarchical agglomerative algorithm (marked in Table II as "average-link") used here does not use any evolutionary technique, the number of FEs is not relevant to this method. This algorithm is supplied with the correct number of clusters for each problem, and we used the Ward updating formula [58] to efficiently recompute the cluster distances.

We used unpaired t-tests to compare the means of the results produced by the best and the second best algorithms. The unpaired t-test assumes that the data have been sampled from a normally distributed population. From the concepts of the central limit theorem, one may note that as sample sizes increase, the sampling distribution of the mean approaches a normal distribution regardless of the shape of the original population. A sample size around 40 allows the normality assumptions conducive for performing the unpaired t-tests [59].

The four evolutionary clustering algorithms can go with any kind of clustering validity measure serving as their fitness functions. We executed two sets of experiments: one using the CS-measure-based fitness function that is shown in (20) while the other using the DB-measure-based fitness function that is shown in (21), with all the four algorithms. For each data set, the quality of the final solution yielded by the four partitional clustering algorithms has been compared with the average-link metric-based hierarchical method in terms of the CS and DB measures.

Finally, we would like to point out that all the algorithms discussed here have been developed in a Visual C++ platform on a Pentium-IV 2.2-GHz PC, with a 512-kB cache and a 2-GB main memory in Windows Server 2003 environment.

E. Experimental Results

To judge the accuracy of the ACDE, DCPSO, GCUK, and classical DE-based clustering algorithms, we let each of them run for a very long time over every benchmark data set, until the number of FEs exceeded 10^6. Then, we note the final fitness value, the number of clusters found, the intercluster distance, i.e., the mean distance between the centroids of the clusters (where the objective is to maximize the distance between clusters), and the intracluster distance, i.e., the mean distance between data vectors within a cluster (where the objective is to minimize the intracluster distances). The latter two objectives respectively correspond to crisp compact clusters that are well separated. In the case of the hierarchical algorithm, the CS value (as well as the DB index) has been calculated over the final results obtained after its termination. In columns 3, 4, 5, and 6 of Table II, we report the mean number of classes found, the final CS value, the intercluster distance, and the intracluster distance obtained for each competitor algorithm, respectively.

Since the benchmark data sets have their nominal partitions known to the user, we also compute the mean number of misclassified data points. This is the average number of objects that were assigned to clusters other than according to the nominal classification. Table III reports the corresponding mean values and standard deviations over the runs obtained in each case of Table II. Table IV shows results of unpaired t-tests taken on the basis of the CS measure between the best two algorithms (standard error of difference of the two means, 95% confidence interval of this difference, the t value, and the two-tailed P value). For all the cases in Table IV, sample size $= 40$. To compare the speeds of different algorithms, we selected a threshhold value of CS measure for each of the data sets. This cutoff CS value is somewhat larger than the minimum CS value found by each algorithm in Table II. Now, we run a clustering algorithm on each data set and stop as soon as the algorithm achieves the proper number of clusters, as well as the CS cutoff value. We then note down the number of fitness FEs that the algorithm takes to yield the cutoff CS value. A lower number of FEs corresponds to a faster algorithm. In columns 3, 4, 5, and 6 of Table V, we report the mean number of FEs, the CS cutoff value, the mean and standard deviation of the final intercluster distance, and the mean and standard deviation of the final intracluster distance (on termination of the algorithm) over 40 independent runs for each algorithm, respectively. In Table VI, we report the misclassification errors (with respect to the nominal classification) for the experiments conducted for Table V. In this table, we exclude the hierarchical average-link algorithm as its time complexity cannot be measured using the number of FEs. It is, however, noted that the runtime of a standard hierarchical algorithm quadratically scales [55].

Tables VII–XI exactly correspond to Tables II–VI with respect to the experimental results, the only difference being that all the experiments conducted for the former group of tables use a DB-measure-based fitness function [see (21)]. In all the tables, the best entries are marked in boldface. Fig. 2 provides a visual feel of the performance of the four clustering methods over the iris data set. The data set has been plotted in three dimensions using the first three features only (Fig. 3).

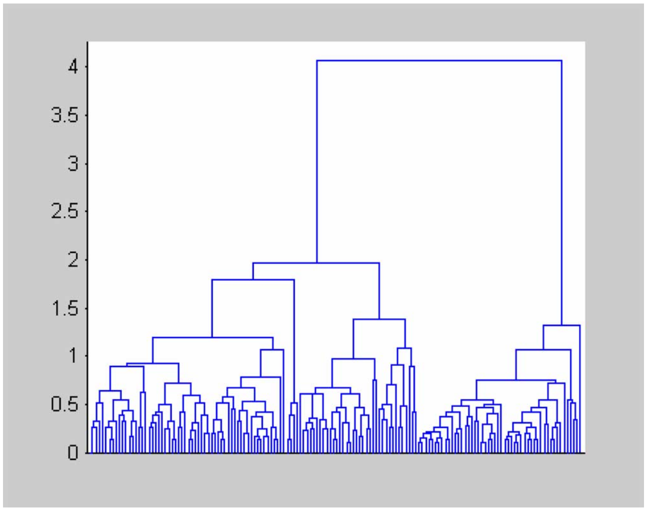


Fig. 3. Dendrogram plot for the iris data set using the average-link hierarchical algorithm.

F. Discussion on the Results (for Real-Life Data Sets)

A scrutiny of Tables II and V reveals the fact that for the iris data set, all the five competitor algorithms terminated with nearly comparable accuracy. The final CS and DB measures were the lowest for the ACDE algorithm. In addition, the ACDE was successful in finding the nearly correct number of classes (three for iris) over repeated runs. However, in Table III, we also find that the GCUK, DCPSO, and classical DE yield two clusters, on average, for the iris data set. One of the clusters corresponds to the Setosa class, whereas the others correspond to the combination of Veriscolor and Virginica. This happens because the latter two classes are considerably overlapping. There are indexes other than the CS or DB measure available in the literature, which yield two clusters for the iris data set [60], [61]. Although the hierarchical algorithm was supplied with the actual number of classes, its performance remained poorer than all the four evolutionary partitional algorithms in terms of the final CS measure, the mean intracluster distance, and the mean intercluster distance.

Substantial performance differences occur for the rest of the more challenging clustering problems with a large number of data items and clusters, as well as overlapping cluster shapes. Tables II and V conform to the fact that the ACDE algorithm remains clearly and consistently superior to the other three competitors in terms of the clustering accuracy. For the breast cancer data set, we observe that both the DCPSO and ACDE yield very close final values of the CS index, and both find two clusters in almost every run. Entries of Table IV testify that the ACDE meets or beats its competitors in a statistically significant manner. We also note that the average-link-based hierarchical algorithm remained the worst performer over these data sets as well.

In Table VII, we find that it is only in one case (for the breast cancer data) that the classical DE-based algorithm yields

TABLE XII
PARAMETER SETUP OF THE CLUSTERING ALGORITHMS FOR THE IMAGE SEGMENTATION PROBLEMS

GCUK		DCPSO		ACDE		Classical DE	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Pop_size	70	Pop_size	40	Pop_size	10*dim	Pop_size	10*dim
Cross-over Probability μ_c	0.85	Inertia Weight	0.75	CR_{max}	1.0	CR	0.95
Mutation probability μ_m	0.005	C_1, C_2	1.494				
		P_{ini}	0.80	CR_{min}	0.5	F	0.9
K_{max} K_{min}	10 2	K_{max} K_{min}	10 2	K_{max} K_{min}	10 2	K_{max} K_{min}	10 2

TABLE XIII
NUMBER OF CLASSES FOUND OVER FIVE REAL-LIFE GRAYSCALE IMAGES AND THE FOLIAGE IMAGE DATABASE USING THE CS-BASED FITNESS FUNCTION (MEAN AND STANDARD DEVIATION OF THE NUMBER OF CLASSES FOUND OVER 40 INDEPENDENT RUNS, WHERE EACH RUN WAS CONTINUED FOR 10^6 FITNESS FEs)

Image	Optimal no. of Clusters	ACDE	DCPSO	GCUK	Classical DE
Clouds	4	**4.15±0.211**	4.50±0.132	4.75±0.432	3.00±0.00
Robot	3	**4.25±0.428**	2..30±0.012	3.35±0.982	3.00±0.004
Science Magazine	4	**4.05±0.772**	3.25±0.082	6.35±0.093	3.50±0.059
Peppers Image	7	**7.05±0.038**	6.85±0.064	3.90±0.447	8.50±0.067
IRS Mumbai Image	6	**6.10±0.079**	4.65±0.674	7.45±0.043	5.25±0.007
Foliages dataset	7	**7.00±0.00**	10.50±1.132	9.50±0.192	**7.00±0.00**

TABLE XIV
AUTOMATIC CLUSTERING RESULT OVER FIVE REAL-LIFE GRAYSCALE IMAGES AND TWO IMAGE DATA SETS USING THE CS-BASED FITNESS FUNCTION (MEAN AND STANDARD DEVIATION OF THE FINAL CS MEASURE FOUND OVER 40 INDEPENDENT RUNS, WHERE EACH RUN WAS CONTINUED FOR 10^6 FITNESS FEs)

Image	CS Measure			
	ACDE	DCPSO	GCUK	Classical DE
Clouds	**0.1317±0.0028**	0.5235±0.0587	0.7806±0.0754	0.1765±0.0332
Robot	**0.19354±0.0028**	0.3478±0.0092	0.9847±0.0846	0.26494±0.00582
Science Magazine	**0.2526±0.0722**	0.4677±0.0493	0.5349±0.0201	0.3509±0.05332
Pepper Image	**0.516±0.0245**	0.9334±0.092	0.7924±0.0160	0.9329±0.0829
IRS Mumbai Image	**0.3892±0.0647**	0.9439±0.0938	0.6992±0.0854	0.3938±0.0693
Foliages dataset	**12.4308±0.3831**	19.8438±0.0921	20.0074±0.3782	14.4965±0.4932

TABLE XV
RESULTS OF THE UNPAIRED t-TEST BETWEEN THE BEST AND THE SECOND BEST PERFORMING ALGORITHMS (FOR EACH DATA SET) BASED ON THE CS MEASURES OF TABLE XIV

Image Dataset	Std. Err	t	95% Conf. Intvl	Two-tailed P	Significance
Clouds	0.001	7.1968	-0.0121 to -0.0068	< 0.0001	**Extremely Significant**
Robot	0.002	3.8990	-0.0129 to -0.0040	< 0.0001	**Extremely Significant**
Science Magazine	0.007	34.9267	-0.2665 to -0.2373	< 0.0001	**Extremely Significant**
Pepper Image	0.001	3.0961	-0.0051 to -0.0010	0.0037	**Very Significant**
IRS Mumbai Image	0.003	3.0684	-0.0156 to -0.0032	0.0040	**Very Significant**
Foliages dataset	0.002	3.0584	-0.0109 to -0.0022	0.0041	**Very Significant**

TABLE XVI
MEAN AND STANDARD DEVIATIONS OF THE NUMBER OF FITNESS FEs (OVER 40 INDEPENDENT RUNS) REQUIRED BY EACH ALGORITHM TO REACH A PREDEFINED CUTOFF VALUE OF THE CS VALIDITY INDEX FOR THE IMAGE CLUSTERING APPLICATIONS

Image Dataset	Algorithm	CS Cut-off Value	Mean No. of FE required
Clouds	ACDE	10.00	**623210.45**±19.32
	DCPSO		827984.75±16.39
	GCUK		790865.90±18.38
	Classical DE		758796.35±27.57
Robot	ACDE	1.00	**434587.35**±25.59
	PSO		733095.85±20.37
	GCUK		498233.75±12.79
	Classical DE		437369.45±10.38
Science Magazine	ACDE	0.60	384732.35±8.39
	DCPSO		564354.60±12.60
	GCUK		678874.90±12.09
	Classical DE		**448949.55±5.35**
Pepper Image	ACDE	1.00	447874.05±9.34
	DCPSO		548738.55±27.42
	GCUK		475854.65±20.48
	Classical DE		676859.60±17.34
IRS Mumbai Image	ACDE	1.50	**503002.00**±18.48
	DCPSO		906543.95±8.42
	GCUK		554362.75±4.48
	Classical DE		786737.35±3.49
Foliage Image dataset	ACDE	21.00	**903002.00**±52.48
	DCPSO		986543.35±35.29
	GCUK		954974.70±44.84
	Classical DE		976473.40±53.20

a lower DB measure, as compared to the ACDE. However, from Table IX, we may note that this difference is not statistically significant.

Results of Tables III and VIII reveal that the ACDE yields the least number of misclassified items once the clustering is over. In this regard, we would like to mention that despite the convincing performance of all the five algorithms, none of the experiments was without misclassification with respect to the nominal classification, which was what we expected. Interestingly, we found that the final fitness values obtained by our evolutionary clustering algorithms were much better than the fitness of the nominal classification, which shows that the misclassification could not be explained by the optimization performance. Instead, misclassification is the result of the underlying assumptions of the clustering fitness criteria (such as the spherical shape of the clusters), outliers in the data set, errors in collecting data, and human errors in the nominal solutions. This is indeed not a negative result. In fact, the differences of a clustering solution based on statistical criteria compared to the nominal classification can reveal interesting data points and anomalies in the data set. In this way, a clustering algorithm can be used as a very useful tool for data preanalysis.

From Tables V and X, we can see that the ACDE was able to reduce both the CS and DB index to the cutoff value within the minimum number of FEs for majority of the cases. Both the DCPSO and classical DE took lesser computational time than the GCUK algorithm over most of the data sets. One possible reason of this may be the use of less complicated variation operators (like mutation) in PSO and DE, as compared to the operators used for GA.

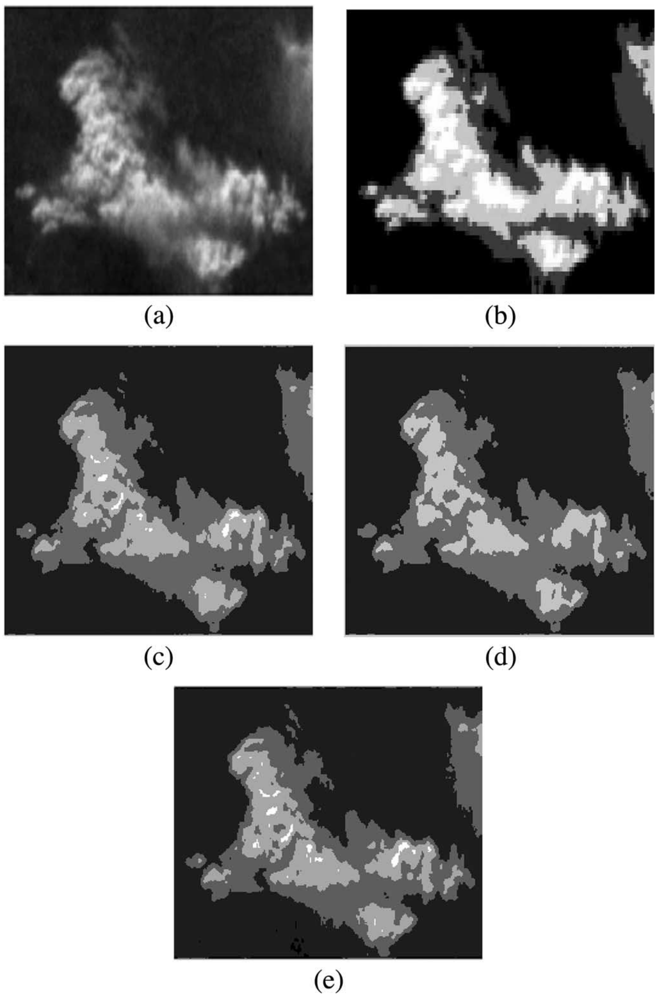


Fig. 4. (a) Original clouds image. (b) Segmentation by ACDE $(K = 4)$. (c) Segmentation by DCPSO $(K = 4)$. (d) Segmentation with GCUK $(K = 4)$. (e) Segmentation with classical DE (provided $K = 3$).

V. APPLICATION TO IMAGE SEGMENTATION

A. Image Segmentation as a Clustering Problem

Image segmentation may be defined as the process of dividing an image into disjoint homogeneous regions. These homogeneous regions usually contain similar objects of interest or part of them. The extent of homogeneity of the segmented regions can be measured using some image property (e.g., pixel intensity [11]). Segmentation forms a fundamental step toward several complex computer vision and image analysis applications, including digital mammography, remote sensing, and land cover study. Segmentation of nontrivial images is one of the most difficult tasks in image processing. Image segmentation can be treated as a clustering problem, where the features describing each pixel correspond to a pattern, and each image region (i.e., segment) corresponds to a cluster [11]. Therefore, many clustering algorithms have widely been used to solve the segmentation problem (e.g., K-means [62], fuzzy C-means [63], ISODATA [64], Snob [65], and, recently, the PSO- and DE-based clustering techniques [51], [53]).

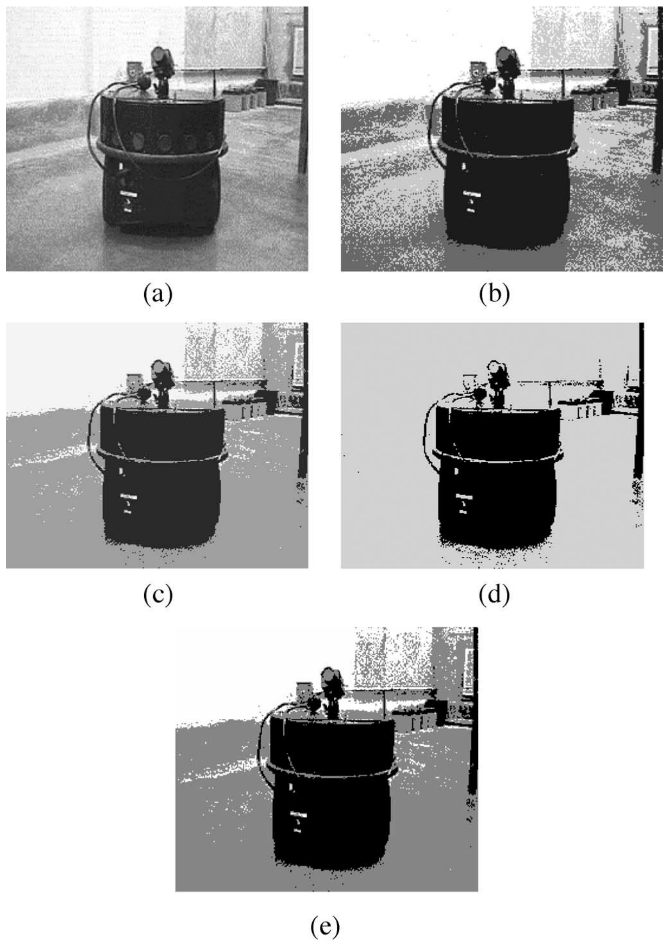


Fig. 5. (a) Original robot image. (b) Segmentation by ACDE $(K = 3)$. (c) Segmentation by DCPSO $(K = 2)$. (d) Segmentation with GCUK $(K = 3)$. (e) Segmentation with classical DE (provided $K = 3$).

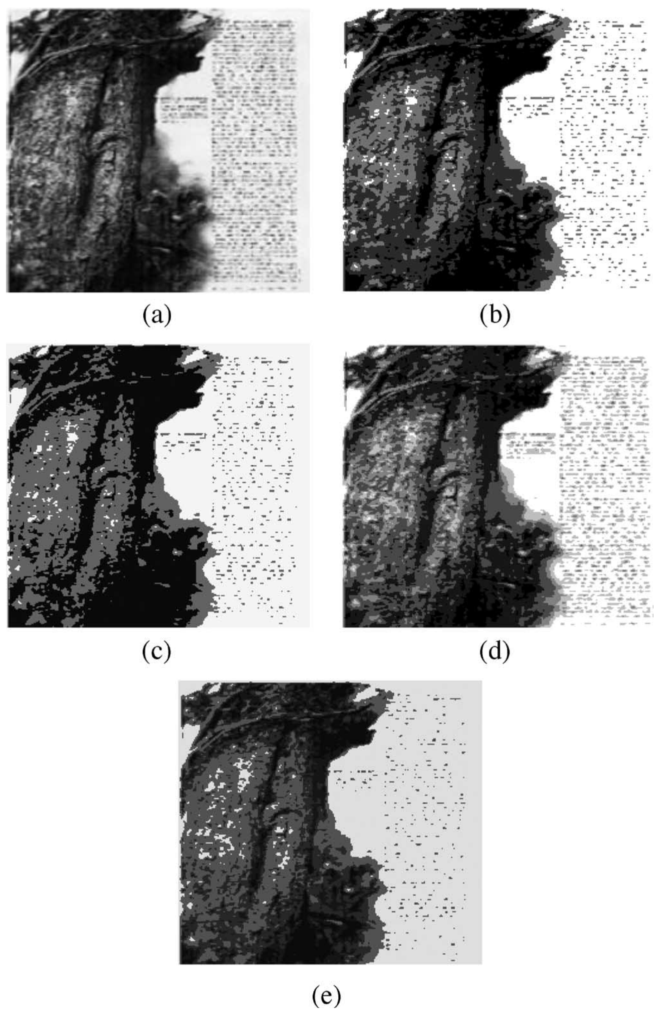


Fig. 6. (a) Original science magazine image. (b) Segmentation by ACDE $(K = 4)$. (c) Segmentation by DCPSO $(K = 3)$. (d) Segmentation with GCUK $(K = 6)$. (e) Segmentation with classical DE (provided $K = 3$).

B. Experimental Details and Results

In this section, we report the results of applying four evolutionary partitional clustering algorithms (ACDE, DCPSO, GCUK, and classical DE) to the segmentation of five 256 $\times$ 256 grayscale images. The intensity level of each pixel serves as a feature for the clustering process. Hence, although the data points are single dimensional, the number of data items is as high as 65 536. Finally, the same four algorithms have been applied to classify an image database, which contains 28 small grayscale images of seven distinct kinds of foliages. Each foliage occurs in the form of a 30 $\times$ 30 digital image. In this case, each data item corresponds to one 30 $\times$ 30 image. Taking the intensity of each pixel as a feature, the dimension of each data point becomes 900. We run two sets of experiments with two fitness functions that are shown in (20) and (21). However, to save space, we only report the CS-measure-based results in this section. To tackle the high-dimensional data points in the last aforementioned problem, we use a cosine distance measure that is described in (5) following the guidelines in [11]. For the rest of the problems, the Euclidean distance measure is used the same as before.

We carried out a thorough experiment with different parameter settings of the clustering algorithms. In Table XII, we report an optimal setup of the parameters that we found best suited for the present image-related problems. With these sets of parameters, we observed each algorithm to achieve considerably good solutions within an acceptable computational time. Note that the parameter settings do not deviate much for the DCPSO and GCUK algorithms than what is recommended in [24] and [25].

Tables XIII and XIV summarize the experimental results obtained over five grayscale images in terms of the mean and standard deviations of the number of classes found and the final CS measure reached at by the four adaptive clustering algorithms. Table XV shows the results of the unpaired t-tests taken based on the final CS measure of Table XIV between the best two algorithms (standard error of difference of the two means, 95% confidence interval of this difference, the t value, and the two-tailed P value). Table XVI records the mean number of FEs required by each algorithm to reach a predefined cutoff CS value. This table helps in comparing the speeds of different algorithms as applied to image pixel classification.

Figs. 4–8 show the five original images and their segmented counterparts obtained using the ACDE, DCPSO, GCUK, and classical DE-based clustering algorithms. Fig. 9 shows the

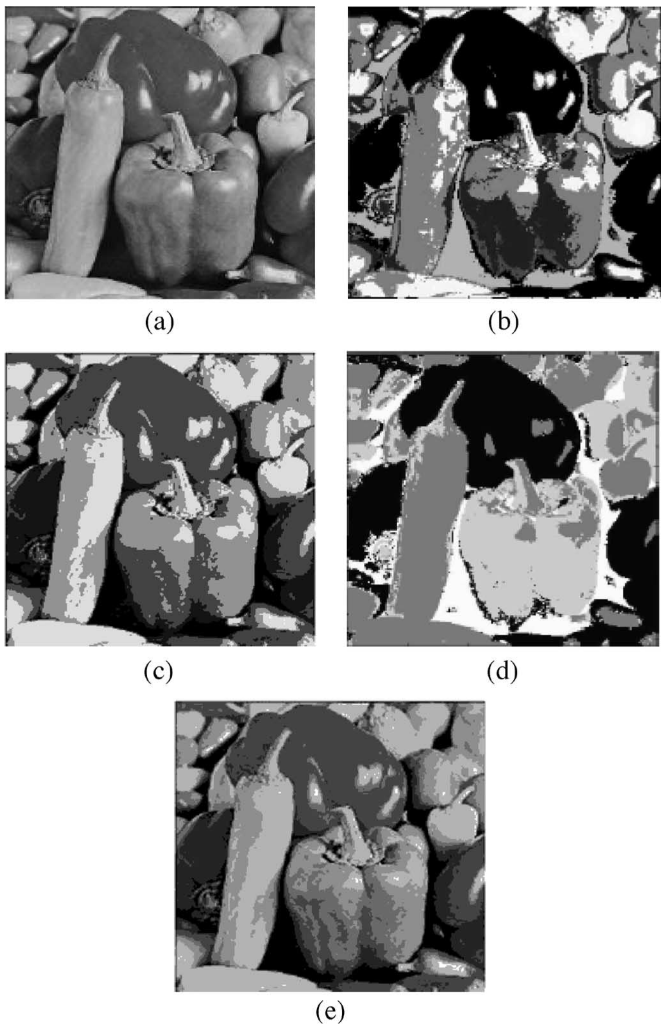


Fig. 7. (a) Original peppers image. (b) Segmentation by ACDE ($K = 7$). (c) Segmentation by DCPSO ($K = 7$). (d) Segmentation with GCUK ($K = 4$). (e) Segmentation with classical DE (provided $K = 8$).

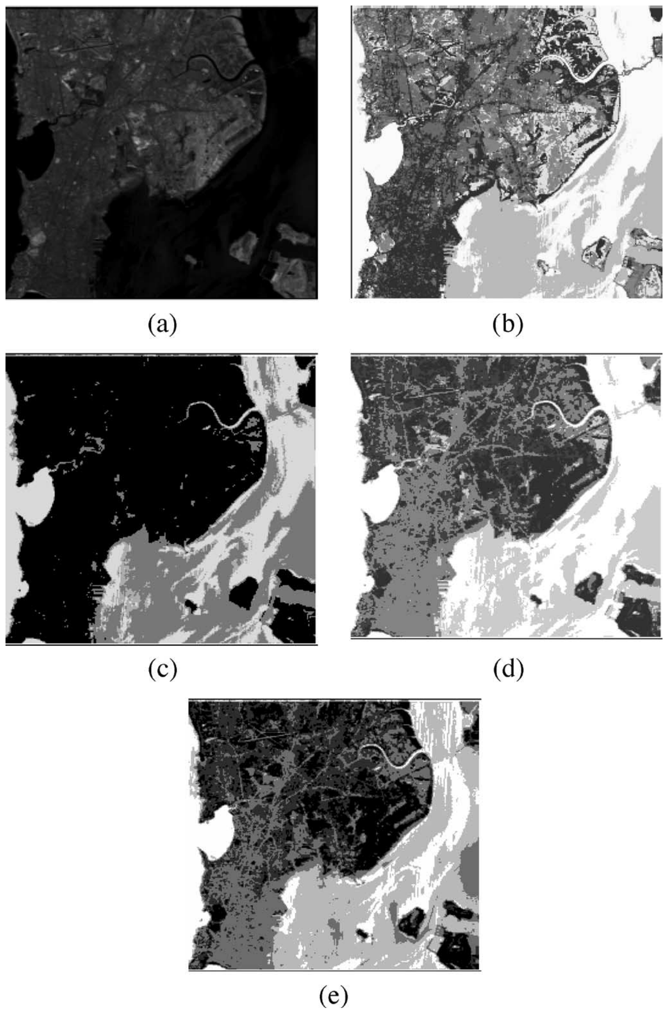


Fig. 8. (a) Original Indian Remote Sensing image of Mumbai. (b) Segmentation by ACDE ($K = 6$). (c) Segmentation by DCPSO ($K = 4$). (d) Segmentation with GCUK ($K = 7$). (e) Segmentation with classical DE (provided $K = 5$).

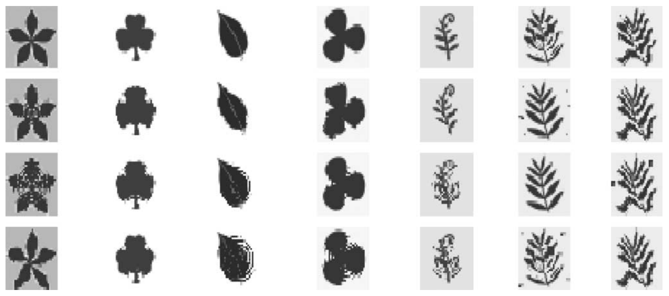


Fig. 9. Nine hundred dimensional training patterns of seven different kinds of foliages.

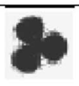
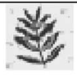
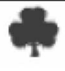
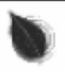
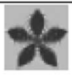
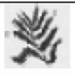
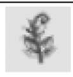
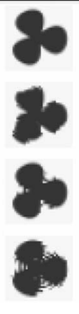
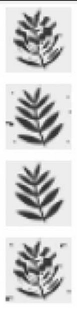
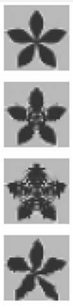
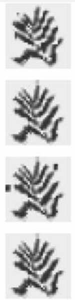
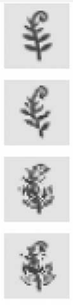
original foliage image database (unclassified). In Table XVII, we report the best classification results achieved with this database using the ACDE algorithm.

C. Discussion on Image Segmentation Results

From Tables XIII–XVI, one may see that our approach outperforms the state-of-the-art DCPSO and GCUK over a variety of image data sets in a statistically significant manner. Not only does the method find the optimal number of clusters, but it also manages to find better clustering of the data points in terms of the two major cluster validity indexes used in the literature. From Table XVII, it is visible that the cluster number of the proposed foliage image patterns is correctly determined by the ACDE, and the cluster center images can represent common and typical features of each class with respect to different types of foliage.

The remote sensing image of Mumbai (a mega city of India) in Fig. 8 bears special significance in this context. Usually, segmentation of such images helps in the land cover analysis of different areas in a country. The new method yielded six clusters for this image. A close inspection of Fig. 8(b) reveals that most of the land cover categories have been correctly distinguished in this image. For example, the *Santa Cruz* airport, the dockyard, the bridge connecting Mumbai to New Mumbai, and many other road structures have distinctly come out. In addition, the predominance of one category of pixels in the southern part of the image conforms to the ground truth; this part is known to be heavily industrialized, and hence, the majority of the pixels in this region should belong to the same class of concrete. The *Arabian Sea* has come out as a combination of pixels of two

TABLE XVII
CLUSTERING RESULT OVER THE FOLIAGE IMAGE PATTERNS BY THE ACDE ALGORITHM

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Cluster Centers							
Members of each class							

different classes. The seawater is found to be decomposed into two classes, i.e., turbid water 1 and turbid water 2, based on the difference of their reflectance properties.

From the experimental results, we note that the ACDE performs much better than the classical DE-based clustering scheme. Since both algorithms use the same chromosome representation scheme and start with the same initial population, the difference in their performance must be due to the difference in their internal operators and parameter values. From this, we may infer that the adaptation schemes suggested for parameters F and Cr of DE in (17) and (18) considerably improved the performance of the algorithm at least for the clustering problems covered here.

VI. CONCLUSION AND FUTURE DIRECTIONS

This paper has presented a new DE-based strategy for crisp clustering of real-world data sets. An important feature of the proposed technique is that it is able to automatically find the optimal number of clusters (i.e., the number of clusters does not have to be known in advance) even for very high dimensional data sets, where tracking of the number of clusters may be well nigh impossible. The proposed ACDE algorithm is able to outperform two other state-of-the-art clustering algorithms in a statistically meaningful way over a majority of the benchmark data sets discussed here. This certainly does not lead us to claim that ACDE may outperform DCPSO or GCUK over every data set since it is impossible to model all the possible complexities of real-life data with the limited test suit that we used for testing the algorithms. In addition, the performance of DCPSO and GCUK may also be enhanced with a judicious parameter tuning, which renders itself to further research with these algorithms. However, the only conclusion we can draw at this point is that DE with the suggested modifications can serve as an attractive alternative for dynamic clustering of completely unknown data sets.

To further reduce the computational burden, we feel that it will be more judicious to associate the automatic research of the clusters with the choice of the most relevant features compared to the process used. Often, we have a great number of features (particularly for a high-dimensional data set like the foliage images), which are not all relevant for a given operation. Hence, future research may focus on integrating the automatic feature-subset selection scheme with the ACDE algorithm. The combined algorithm is expected to automatically project the data to a low-dimensional feature subspace, determine the number of clusters, and find out the appropriate cluster centers with the most relevant features at a faster pace.

REFERENCES

[1] I. E. Evangelou, D. G. Hadjimitsis, A. A. Lazakidou, and C. Clayton, "Data mining and knowledge discovery in complex image data using artificial neural networks," in *Proc. Workshop Complex Reason. Geogr. Data*, Paphos, Cyprus, 2001.
[2] T. Lillesand and R. Keifer, *Remote Sensing and Image Interpretation*. Hoboken, NJ: Wiley, 1994.
[3] H. C. Andrews, *Introduction to Mathematical Techniques in Pattern Recognition*. New York: Wiley, 1972.
[4] M. R. Rao, "Cluster analysis and mathematical programming," *J. Amer. Stat. Assoc.*, vol. 66, no. 335, pp. 622–626, Sep. 1971.
[5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. Hoboken, NJ: Wiley, 1973.
[6] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.
[7] B. S. Everitt, *Cluster Analysis*, 3rd ed. New York: Halsted, 1993.
[8] J. A. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.
[9] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 450–465, May 1999.
[10] Y. Leung, J. Zhang, and Z. Xu, "Clustering by scale-space filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1396–1410, Dec. 2000.
[11] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.
[12] E. W. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," *Biometrics*, vol. 21, no. 3, pp. 768–769, 1965.
[13] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. C-20, no. 1, pp. 68–86, Jan. 1971.
[14] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[15] J. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 296–317, Mar. 1995.
[16] N. R. Pal, J. C. Bezdek, and E. C.-K. Tsao, "Generalized clustering networks and Kohonen's self-organizing scheme," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 549–557, Jul. 1993.
[17] T. Kohonen, *Self-Organizing Maps*, vol. 30. Berlin, Germany: Springer-Verlag, 1995.
[18] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. Chichester, U.K.: Wiley, 1998.

[19] S. Paterlini and T. Minerva, "Evolutionary approaches for cluster analysis," in *Soft Computing Applications*, A. Bonarini, F. Masulli, and G. Pasi, Eds. Berlin, Germany: Springer-Verlag, 2003, pp. 167–178.

[20] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[21] S. Z. Selim and K. Alsultan, "A simulated annealing algorithm for the clustering problem," *Pattern Recognit.*, vol. 24, no. 10, pp. 1003–1008, 1991.

[22] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Probability*, 1967, pp. 281–297.

[23] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[24] S. Bandyopadhyay and U. Maulik, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern Recognit.*, vol. 35, no. 6, pp. 1197–1208, Jun. 2002.

[25] M. Omran, A. Salman, and A. Engelbrecht, "Dynamic clustering using particle swarm optimization with application in unsupervised image classification," in *Proc. 5th World Enformatika Conf. (ICCI)*, Prague, Czech Republic, 2005.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[27] A. Konar, *Computational Intelligence: Principles, Techniques and Applications*. Berlin, Germany: Springer-Verlag, 2005.

[28] P. Brucker, "On the complexity of clustering problems," in *Optimization and Operations Research*, vol. 157, M. Beckmenn and H. P. Kunzi, Eds. Berlin, Germany: Springer-Verlag, 1978, pp. 45–54.

[29] G. Hamerly and C. Elkan, "Learning the K in K-means," in *Proc. NIPS*, Dec. 8–13, 2003, pp. 281–288.

[30] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment: Finding the optimal partitioning of a data set," in *Proc. IEEE ICDM*, San Jose, CA, 2001, pp. 187–194.

[31] J. C. Dunn, "Well separated clusters and optimal fuzzy partitions," *J. Cybern.*, vol. 4, pp. 95–104, 1974.

[32] R. B. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Stat.*, vol. 3, no. 1, pp. 1–27, 1974.

[33] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 1, no. 2, pp. 224–227, Apr. 1979.

[34] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik, "Validity index for crisp and fuzzy clusters," *Pattern Recognit. Lett.*, vol. 37, no. 3, pp. 487–501, Mar. 2004.

[35] C. H. Chou, M. C. Su, and E. Lai, "A new cluster validity measure and its application to image compression," *Pattern Anal. Appl.*, vol. 7, no. 2, pp. 205–220, Jul. 2004.

[36] V. V. Raghavan and K. Birchand, "A clustering strategy based on a formalism of the reproductive process in a natural system," in *Proc. 2nd Int. Conf. Inf. Storage Retrieval*, 1979, pp. 10–22.

[37] C. A. Murthy and N. Chowdhury, "In search of optimal clusters using genetic algorithm," *Pattern Recognit. Lett.*, vol. 17, no. 8, pp. 825–832, Jul. 1996.

[38] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "Pattern classification with genetic algorithms," *Pattern Recognit. Lett.*, vol. 16, no. 8, pp. 801–808, Aug. 1995.

[39] R. Srikanth, R. George, N. Warsi, D. Prabhu, F. E. Petri, and B. P. Buckles, "A variable-length genetic algorithm for clustering and classification," *Pattern Recognit. Lett.*, vol. 16, no. 8, pp. 789–800, Aug. 1995.

[40] Y. C. Chiou and L. W. Lan, "Theory and methodology genetic clustering algorithms," *Eur. J. Oper. Res.*, vol. 135, no. 2, pp. 413–427, 2001.

[41] K. Krishna and M. N. Murty, "Genetic K-means algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, no. 3, pp. 433–439, Jun. 1999.

[42] R. J. Kuo, J. L. Liao, and C. Tu, "Integration of ART2 neural network and genetic K-means algorithm for analyzing web browsing paths in electronic commerce," *Decis. Support Syst.*, vol. 40, no. 2, pp. 355–374, Aug. 2005.

[43] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2/3, pp. 107–145, Dec. 2001.

[44] S. Theodoridis and K. Koutroubas, *Pattern Recognition*. New York: Academic, 1999.

[45] C. Rosenberger and K. Chehdi, "Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation," in *Proc. IEEE ICPR*, Barcelona, Spain, 2000, vol. 1, pp. 656–659.

[46] C.-Y. Lee and E. K. Antonsson, "Self-adapting vertices for mask-layout synthesis," in *Proc. Model. Simul. Microsyst. Conf.,* M. Laudon and B. Romanowicz, Eds., San Diego, CA, Mar. 27–29, 2000, pp. 83–86.

[47] H.-P. Schwefel, *Evolution and Optimum Seeking*, 1st ed. New York: Wiley, 1995.

[48] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[49] M. Sarkar, B. Yegnanarayana, and D. Khemani, "A clustering algorithm using an evolutionary programming-based approach," *Pattern Recognit. Lett.*, vol. 18, no. 10, pp. 975–986, Oct. 1997.

[50] S. Paterlinia and T. Krink, "Differential evolution and particle swarm optimisation in partitional clustering," *Comput. Stat. Data Anal.*, vol. 50, no. 5, pp. 1220–1247, Mar. 2006.

[51] M. Omran, A. Engelbrecht, and A. Salman, "Particle swarm optimization method for image clustering," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 3, pp. 297–322, 2005.

[52] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Conf. Syst., Man, Cybern.*, 1997, pp. 4104–4108.

[53] M. Omran, A. P. Engelbrecht, and A. Salman, "Differential evolution methods for unsupervised image classification," in *Proc. 7th CEC*, 2005, pp. 966–973.

[54] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. ACM-SIGEVO GECCO*, Washington, DC, 2005, pp. 991–998.

[55] W. H. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *J. Classif.*, vol. 1, no. 1, pp. 1–24, Dec. 1984.

[56] C. Blake, E. Keough, and C. J. Merz, *UCI Repository of Machine Learning Database*, 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLrepository.html

[57] S. K. Pal and D. D. Majumder, "Fuzzy sets and decision making approaches in vowel and speaker recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, no. 8, pp. 625–629, Aug. 1977.

[58] C. Olson, "Parallel algorithms for hierarchical clustering," *Parallel Comput.*, vol. 21, no. 8, pp. 1313–1325, Aug. 1995.

[59] B. Flury, *A First Course in Multivariate Statistics*, vol. 28. Berlin, Germany: Springer-Verlag, 1997.

[60] J. C. Bezdek and N. R. Pal, "Some new indexes of cluster validity," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 3, pp. 301–315, Jun. 1998.

[61] R. Kothari and D. Pitts, "On finding the number of clusters," *Pattern Recognit. Lett.*, vol. 20, no. 4, pp. 405–416, Apr. 1999.

[62] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. London, U.K.: Addison-Wesley, 1974.

[63] M. M. Trivedi and J. C. Bezdek, "Low-level segmentation of aerial images with fuzzy clustering," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 4, pp. 589–598, Jul. 1986.

[64] G. Ball and D. Hall, "A clustering technique for summarizing multivariate data," *Behav. Sci.*, vol. 12, no. 2, pp. 153–155, Mar. 1967.

[65] C. S. Wallace and D. M. Boulton, "An information measure for classification," *Comput. J.*, vol. 11, no. 2, pp. 185–194, Aug. 1968.

Swagatam Das was born in Kolkata, India, in 1980. He received the B.E.Tel.E. and M.E.Tel.E. degrees in control engineering in 2003 and 2005, respectively, from Jadavpur University, Kolkata, where he is currently working toward the Ph.D. degree.

He is currently a Lecturer with the Department of Electronics and Telecommunication Engineering, Jadavpur University. He is the author or coauthor of more than 25 papers published in international journals and conference proceedings. His research interests include evolutionary computing, swarm intelligence, pattern recognition, data mining, chaos theory, and bioinformatics.

Mr. Das has been a reviewer for several journals such as *Pattern Recognition* and IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS. He was the recipient of the Best Paper Award at the Sixth International Conference on Intelligent Systems Design and Applications (ISDA2006), Jinan, China.

Ajith Abraham (M'96–SM'07) received the Ph.D. degree from Monash University, Melbourne, Australia, in 2001.

He is currently a Visiting Professor with the Centre for Quantifiable Quality of Service in Communication Systems (Q2S), Centre of Excellence, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. Before joining NTNU, he was working under the South Korean Government's Institute of Information Technology Advancement (IITA) Professorship Program at Yonsei University, Seoul, Korea, and Chung-Ang University, Seoul. He was a Visiting Researcher with Rovira i Virgili University, Tarragona, Spain, during 2005–2006 and is currently an Adjunct Professor with Jinan University, Jinan, China, and Dalian Maritime University, Dalian, China. He has authored or coauthored more than 300 research publications in peer-reviewed reputed journals, book chapters, and conference proceedings. His primary research interests are in computational intelligence, with a focus on using global optimization techniques for designing intelligent systems. His application areas include Web services, information security, Web intelligence, financial modeling, multicriteria decision making, data mining, etc. He is a regular reviewer of IEEE Intelligent Systems, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON NEURAL NETWORKS, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, and IEEE TRANSACTIONS ON POWER SYSTEMS.

Dr. Abraham serves on the Editorial Boards of more than a dozen international journals and has also guest edited 23 special issues on various topics for international journals. Since 2001, he has been actively involved in the Hybrid Intelligent Systems and the Intelligent Systems Design and Applications series of annual international conferences. He was the recipient of five Best Paper Awards.

Amit Konar (M'97) received the B.E. degree from the Bengal Engineering and Science University, Shibpur, India, in 1983 and the M.E.Tel.E. and Ph.D. degrees from Jadavpur University, Kolkata, India, in 1985 and 1994, respectively.

He is currently a Professor with the Department of Electronics and Telecommunication Engineering, Jadavpur University. He was a Visiting Professor for the summer courses with the University of Missouri, St. Louis, in 2006. His research areas include the study of computational intelligence algorithms and their applications to the entire domain of electrical engineering and computer science. He has specifically worked on fuzzy sets and logic, neurocomputing, evolutionary algorithms, Dempster–Shafer theory, and Kalman filtering and has applied the principles of computational intelligence in image understanding, VLSI design, mobile robotics, and pattern recognition. He has supervised ten Ph.D. theses. He currently serves on the Editorial Board of the *International Journal of Hybrid Intelligent Systems* and the *International Journal of Neurocomputing*. He is the author or coauthor of more than 130 papers published in international journals and conference proceedings and is the author of five books, four of which have been published by Springer-Verlag, Germany, and one has been published by CRC Press, Florida.

Dr. Konar was a recipient of the All India Council of Technical Education (AICTE)-accredited 1997–2000 Career Award for Young Teachers for his significant contribution in teaching and research.