Elastic Differential Evolution for Automatic Data Clustering

Jun-Xian Chen, Yue-Jiao Gong, *Senior Member, IEEE*, Wei-Neng Chen, *Senior Member, IEEE*, Mengting Li, and Jun Zhang, *Fellow, IEEE*

***Abstract*—In many practical applications, it is crucial to perform automatic data clustering without knowing the number of clusters in advance. The evolutionary computation paradigm is good at dealing with this task, but the existing algorithms encounter several deficiencies, such as the encoding redundancy and the cross-dimension learning error. In this article, we propose a novel elastic differential evolution algorithm to solve automatic data clustering. Unlike traditional methods, the proposed algorithm considers each clustering layout as a whole and adapts the cluster number and cluster centroids inherently through the variable-length encoding and the evolution operators. The encoding scheme contains no redundancy. To enable the individuals of different lengths to exchange information properly, we develop a subspace crossover and a two-phase mutation operator. The operators employ the basic method of differential evolution and, in addition, they consider the spatial information of cluster layouts to generate offspring solutions. Particularly, each dimension of the parameter vector interacts with its correlated dimensions, which not only adapts the cluster number but also avoids the cross-dimension learning error. The experimental results show that our algorithm outperforms the state-of-the-art algorithms that it is able to identify the correct number of clusters and obtain a good cluster validation value.**

***Index Terms*—Clustering, differential evolution, elastic encoding, subspace.**

I. Introduction

DATA clustering plays a crucial role in various fields, such as industrial informatics [1], [2]; bioinformatics [3]–[6]; and pattern recognition [7]–[9]. The task is to partition a dataset into different data subsets named clusters. Commonly, data clustering is performed in an unsupervised way using only the data distribution information. The principle is that we need to maximize the intracluster similarity while minimizing the intercluster similarity. Therefore, data clustering can be regarded as an optimization problem.

Different types of clustering algorithms were developed in the literature, such as the partitioning methods, the agglomerative or divisive hierarchical clustering, the prototype-based methods, and the density-based methods. In this article, we focus on partitioning the clustering which organizes the data objects into a number of exclusive clusters. The most well-known technique in this category is the K-means clustering algorithm [10]. Although it was proposed as early as 1979, the algorithm is still widely used in various fields in recent years [11], [12]. However, there are several shortcomings of the traditional partition-based clustering algorithms such as the K-means. First, they require *a priori* knowledge, namely, the predefined number of clusters. Given a dataset, one may need to use an analytical technique to run the algorithm repeatedly to identify the most suitable K value to improve the performance. Second, these algorithms are typically the local search algorithms which provide only locally optimal solutions. The results would be highly sensitive to the initial partitions of the data objects.

From the perspective of optimization, clustering is shown to be a nondeterministic polynomial NP-hard problem [13] and, therefore, the approximation algorithms, such as the evolutionary computation (EC) algorithms become the potential solvers. Many attempts have been devoted to this area. For example, Krishna and Murty [14] proposed a genetic K-means algorithm (GKA) with a specified cluster number for data clustering and replaced the crossover operator in the genetic algorithm with one step of K-means. In [15], a faster GKA (FGKA) was put forward by Lu *et al.* Merz and Zell [16] adopted some memetic algorithms for clustering gene expression data. In these algorithms, the cluster number is often regarded as an input, due to the fixed-length encoding scheme of EC algorithms. However, when dealing with high-dimensional and big-data problems, it becomes a tedious or even impossible task to predefine the optimal number of clusters. Therefore, we need an automatic way to adjust the number of clusters during the optimization process.

Recently, several techniques have been developed to address this issue. Das *et al.* [17] adopted a fixed-length structure to encode the coordinates of the cluster centroids, together

Manuscript received May 15, 2019; revised August 12, 2019 and September 6, 2019; accepted September 11, 2019. Date of publication October 11, 2019; date of current version August 4, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61873095, Grant 61873097, and Grant U1701267, in part by the Guangzhou Science and Technology Planning Project under Grant 201904010211, in part by the Guangdong Natural Science Foundation Research Team Project under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform Project under Grant 2018B050502006. This article was recommended by Associate Editor P. N. Suganthan. *(Corresponding author: Yue-Jiao Gong.)*

J.-X. Chen, Y.-J. Gong, and W.-N. Chen are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: gongyuejiao@gmail.com).

M. Li is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

J. Zhang is with the Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2019.2941707.

Digital Object Identifier 10.1109/TCYB.2019.2941707

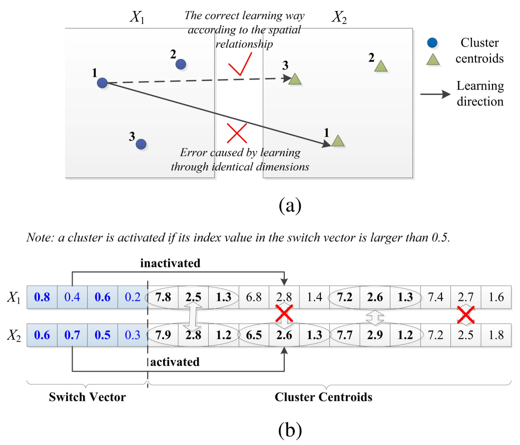
X_1
The correct learning way according to the spatial relationship
X_2
Cluster centroids
Learning direction
1
2
3
3
2
1
Error caused by learning through identical dimensions
(a)
Note: a cluster is activated if its index value in the switch vector is larger than 0.5.
inactivated
X_1 0.8 0.4 0.6 0.2 7.8 2.5 1.3 6.8 2.8 1.4 7.2 2.6 1.3 7.4 2.7 1.6
X_2 0.6 0.7 0.5 0.3 7.9 2.8 1.2 6.5 2.6 1.3 7.7 2.9 1.2 7.2 2.5 1.8
activated
Switch Vector
Cluster Centroids
(b)

Fig. 1. Illustrations of the cross-dimension learning error. (a) Individuals exchange information in the wrong way (the individual X_1 wrongly takes the index-based dimensions as learning direction). (b) Invalid learning due to the switch mechanism (the activated centroid learns invalid information from the inactivated one).

with a zero-one switch vector to represent the state of each cluster centroid. The cluster centroid is activated when its corresponding switch value is decoded into one. Since the switch vector is updated iteratively, the number of used clusters is adjusted during the optimization. The work of Liu *et al.* [18] follows this method, but it makes improvements in representing the activate state of the cluster centroids. The three adaptive encoding schemes reduce the number of bits in the chromosome to label the ON/OFF state of the cluster centroids. In [19], a genetic algorithm was developed to perform the clustering task, and the algorithm found the right number of clusters according to the average Silhouette width criterion. But the encoding scheme remained the constant-length parameter vector. Hruschka *et al.* [20] improved a genetic algorithm to estimate the number of clusters, and the algorithm performed well in gene expression data. Although it is able to adjust the cluster numbers, this kind of algorithm endures two deficiencies. First, the algorithms encode and optimize the maximum number of cluster centroids, regardless of whether they are activated or not. The search space is large and full of redundancy. The increase of problem dimensionality could lower the performance of the EC algorithms. Second, to the best of our knowledge, all existing EC-based clustering algorithms encounter the problem of "cross-dimension learning error." Cross-dimension learning error occurs when the individuals in the population interact with each other (e.g., the crossover of the GA and DE algorithms, or when a particle learns information from the others in the PSO). It has two situations, which are illustrated in Fig. 1 and described as follows.

First, the individuals in the traditional EC algorithms exchange information according to the dimension indices. When performing data clustering, this mechanism neglects the spatial relationship between the cluster centroids that a cluster centroid can exchange/learn the information from a totally unrelated cluster centroid. Fig. 1(a) shows a normal situation that two parameter vectors exchange the incorrect information: according to the dimension indices (cluster labels), the centroid 1 of X_1 interacts with the centroid 1 of X_2. However,

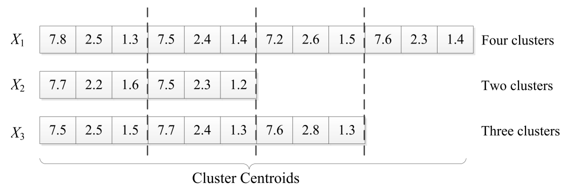
X_1 7.8 2.5 1.3 7.5 2.4 1.4 7.2 2.6 1.5 7.6 2.3 1.4 Four clusters
X_2 7.7 2.2 1.6 7.5 2.3 1.2 Two clusters
X_3 7.5 2.5 1.5 7.7 2.4 1.3 7.6 2.8 1.3 Three clusters
Cluster Centroids

Fig. 2. Elastic encoding scheme with variable lengths.

according to the spatial layout, the correct interaction should be between the centroid 1 of X_1 and the centroid 3 of X_2 because they correlate with each other.

Second, for automatic data clustering, many EC algorithms with a fixed-length structure encode the coordinates of all possible cluster centroids, while using a switch vector to represent the state (activated or not) of each cluster centroid. In these cases, the activated centroid may learn invalid information from the inactivated one, as shown in Fig. 1(b). However, the coordinates of inactivated centroids could be obsolete, which would provide wrong guidance. Because of the above cross-dimension learning error, the EC-based clustering algorithms become inefficient or vulnerable to premature convergence.

To summarize, the state-of-the-art clustering algorithms have two remaining problems to solve. First, the existing differential evolution algorithms use a fixed-length encoding scheme for automatic data clustering, but apply some auxiliary space to represent the activate state of the cluster centroids. The search space is redundant, which reduces clustering efficiency. Second, the existing evolutionary clustering algorithms commonly suffer cross-dimension learning error when performing crossover or individual learning. The existence of the cross-dimension learning error disorders the interactions between individuals and, hence, reduces the performance of a population-based optimization approach. To address the problems, we propose a novel elastic differential evolutionary (E-DE) algorithm in this article. The novelties and contributions of the E-DE algorithm are summarized as follows.

1) E-DE adopts an elastic encoding scheme where the population consists of variable-length parameter vectors, each denotes a different number of clusters. As illustrated in Fig. 2, the encoding scheme contains no redundancy: each vector possesses only the coordinates of the cluster centroids that take effect in the clustering of data. E-DE considers each clustering layout (number and positions of clusters) as a whole to perform optimization, and it allows the parameter vector to change its cluster number flexibly during the search process.
2) Based on our new elastic encoding scheme, the mutation and crossover operators of DE are sophisticatedly redefined to fit the requirements arising from the variation of parameter vector length. The new mutation adapts the cluster number and centroids according to the differential information of individuals in the current population. Meanwhile, we develop a subspace crossover that takes the spatial layout of clusters into consideration, letting each dimension of one vector learn from the correlated dimensions of the other vectors. To

the best of our knowledge, this article makes the first attempt to eliminate the cross-dimension learning error in evolutionary clustering algorithms.

3) Unlike some other peer algorithms that introduce many complex auxiliary techniques, we keep our algorithm conceptually simple. The framework of E-DE is identical to the classical DE algorithm, which iteratively performs the crossover, mutation, and selection steps to refine the population. The adjustment and optimization of cluster number and coordinates is realized through this process in an inherent way. Our algorithm is lightweight and computationally efficient.
4) We compare the performance of our algorithm with several state-of-the-art clustering algorithms on synthetic and real-life datasets of various characteristics. The experimental results indicate that the E-DE algorithm outperforms the others.

The remainder of this article is organized as follows. Section II discusses the traditional and state-of-the-art clustering algorithms and points out some drawbacks of the remaining methods. In Section III, we describe the elastic differential evolution algorithm in detail. Section IV shows the experiments and results. The conclusions are demonstrated in Section V.

II. Preliminaries

A. Related Work

The data clustering problem, which aims at partitioning the dataset into different clusters, is an NP-hard problem [13]. The evolutionary algorithms (EAs) are commonly known as the powerful optimizers for the NP-hard problems, so they have received wide attention for data clustering. There are some related survey papers on this topic [21]–[23]. In this section, we review the evolutionary clustering algorithms according to two categories: 1) algorithms with a fixed cluster number and 2) algorithms with a variable cluster number.

1) Clustering With Fixed Number of Clusters: In this category, the algorithms require a specified cluster number as *a priori* knowledge, in order to fit the fixed-length encoding scheme of EAs. Many studies made improvements on the evolution operators to enhance the performance of clustering.

1) *Single-Objective Optimization Methods:* Krishna and Murty [14] proposed a GKA for data clustering, where the elementary step of K-means was taken to replace the crossover step in the evolution process. Lu *et al.* [15] improved the GKA in terms of the selection and mutation operators and made it run faster. Merz and Zell [16] adopted the memetic algorithms with two mutation and recombination operators for clustering gene expression data. These methods enable the algorithm to quickly converge to a near-optimal solution. Xiang *et al.* [24] employed the DE/best/1 mutation strategy to develop a dynamic shuffled differential EA (SDE) to improve the convergence performance of data clustering. The SDE algorithm divides the initial population into two subpopulations to evolve and combine the best results to form the best optimum at the end of optimization. More recently, a density-based NK hybrid genetic algorithm was developed in [25]. To determine the number of clusters, the algorithm first generates a large number of solutions and selects the solution with the best validation index. In 2019, Guan *et al.* [26] proposed a particle-swarm-optimized density-based clustering and classification (PODCC) algorithm, which successfully offset the drawbacks of the well-known DBSCAN algorithm for density clustering.
2) *Multiobjective Optimization Methods:* In recent years, the evolutionary multiobjective optimization (EMO) methods have also received attention for data clustering. Some studies focus on fuzzy clustering, such as the multiobjective genetic algorithm by Mukhopadhyay *et al.* [27] and the multiobjective differential EA by Saha *et al.* [28]. The algorithms optimize more than one measure for the clustering goodness and generate a set of Pareto optimal solutions for the subsequent operations.

2) Clustering With a Variable Number of Clusters: Because the number of clusters can hardly be known in advance, some efforts are made to estimate the number of clusters before performing clustering, such as the last leap and the last major leap methods proposed in [29]. The term "automatic clustering" commonly refers to the methods that are able to adapt the number of clusters automatically without external specification. To fulfill this goal, different kinds of heuristic or local search strategies are incorporated into EAs to enable the algorithms to change the cluster number during the clustering process.

1) *Single-Objective Optimization Methods:* Das *et al.* [17] put forward an improved differential EA called ACDE that adjusted the cluster number through a new individual encoding scheme. The ACDE algorithm embeds a zero-one switch vector in the individual representation for activating or inactivating the cluster centroids. An improved ACDE version was proposed by Tam *et al.* [30] using a new changing schema of threshold values. Further, Liu *et al.* [18] developed three adaptive encoding schemes that represent the activate state of the cluster centroids more efficiently. Arellano-Verdejo *et al.* [31] proposed a new hybrid differential EA (DELA), and Tvrdik and Křivý [32] combined differential evolution and K-means. They paid more attention to the center rearrangement. Maulik and Saha [33] proposed a modified differential evolution for automatic fuzzy clustering, which adopted the same individual encoding scheme like the ACDE. In addition, the global best and local best information are utilized to accelerate the searching process. Besides, Chang and Zhang [34] put forward a dynamic niching genetic algorithm for data clustering. Since the niching method is able to preserve the diversity of the population, the algorithm is more likely to find the correct cluster number and well-formed data structure. Sheng *et al.* [35] proposed a multilocal search and adaptive niching-based memetic algorithm (MANM) for data clustering. Two local search strategies merge and split clusters in the evolution process. Then, at the

end of each iteration, an adaptive niching strategy is applied to update the population information in the offspring. Furthermore, Sheng *et al.* [36] improved the performance of the MANM and designed a new memetic algorithm with adaptive multisubpopulation competition and multiniche crowding methods. In [37], the two-stage genetic clustering algorithm (TGCA) adopts a variable-length encoding like the proposed E-DE algorithm. Then, as the name indicated, the TGCA treats the number of clusters and the positions of clusters separately by a two-stage method during the evolution. The experiments showed good performance of TGCA. Besides, in [38], a weighted Gaussian means algorithm was developed, which adopts a feature weighting learning strategy for automatic data clustering.

2) *Multiobjective Optimization Methods:* The multiobjective optimization methods are also well suited in this category. Saha *et al.* [39] proposed a framework of multiobjective optimization to solve the semisupervised clustering. Four objective functions are applied: a) the total compactness of the partitioning; b) the total symmetry present in the clusters; c) the cluster connectedness; and d) the adjust rand index. Mukhopadhyay *et al.* [40] put forward a fuzzy C-medoid method for categorical data clustering using the multiobjective genetic algorithm. In the evolution process, the fuzzy cluster variance and cluster separation are taken as two objectives to be optimized simultaneously. Wan *et al.* [41] put forward an adaptive multiobjective memetic algorithm for spectral–spatial fuzzy clustering. The method contains two automatic layers and accomplishes the clustering task of remote sensing images very well. Garza-Fabre *et al.* [42] proposed an improved and more scalable evolutionary approach to multiobjective clustering. It utilizes two encoding schemes for better population representation, that is, the delta locus and the delta binary encoding schemes.

The existing clustering algorithms have the following limitations. First, the algorithms with a fixed number of clusters require prior knowledge that can hardly be known in practical applications. The EC-based automatic data clustering algorithms relieve this problem. However, as discussed in Section I, these algorithms endure the drawbacks of encoding redundancy and cross-dimension learning error. Besides, many algorithms rely on auxiliary techniques, such as niching [35] and population competition [36] to adjust the number of clusters, which increase computational complexity. To tackle the aforementioned limitations, we proposed an elastic differential EA for automatic data clustering. The algorithm is achieved by a new elastic encoding scheme and the corresponding new mutation and crossover operator under the concept of subspace swapping.

B. Validation Indices for Clustering

For the clustering problem, the ultimate purpose of the algorithm is to find a well-formed clustering structure for the dataset with maximum intercluster distance and minimum intracluster distance. Numerical measures are applied to judge the performance of data clustering, which include external indices and internal indices [43]. An external index measures the clustering results with a reference model, like the partition result provided by a domain expert. In contrast, an internal index directly measures the results based on the intrinsic characteristics of clusters, which does not refer to the external information. There exist a large number of internal indices for clustering validation, such as the Dunn index, CH index, Gamma index, C-index, SIL index, DB index, CS index, I index, and so on. Since there does not exist a "perfect" index so far, many works were devoted to make comparisons between different indices, such as the review work [44].

In this article, we focus on proposing an efficient automatic clustering algorithm. The developed E-DE is generic, which is suitable to adopt any of the above-mentioned indices in the fitness evaluation step. In the major body of this article, we use the DB [45] and I [46] indices that are commonly adopted in the literature. The two indices are simple to understand, lightweight to implement, and are validated to be efficient for various clustering datasets. Besides, we also implement and test the algorithms by adopting the other indices, which are reported in the supplementary file of this article.

1) DB Index: In the DB index, each cluster centroid corresponds to an index value R_i

$$R_i = \max_{j,j\neq i}\left(\frac{e_i + e_j}{D_{ij}}\right) \tag{1}$$

where $e_i = (1/N_i)\sum_{\boldsymbol{x}_i\in C_i}\|\boldsymbol{x}_i - \boldsymbol{m}_i\|^2$ is the average error within cluster i; $D_{ij} = \|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2$ is the distance between the cluster centroids; and N_i denotes the number of data points in cluster C_i. Then, the DB index is formulated as

$$\mathrm{DB}(K) = \frac{1}{K}\sum_{i=1}^{K} R_i. \tag{2}$$

Since we aim at maximizing the intercluster distance and minimizing the intracluster distance, the smaller the DB index value is, the better the clustering result will be.

2) I Index: The I index includes three factors that compete with each other, which can be defined as

$$\mathrm{I}(K) = \left(\frac{1}{K} \times \frac{P}{E_k} \times D_{\max}\right)^p \tag{3}$$

where K is the number of clusters; P is a constant related to the given dataset, which is set to the number of data samples (N); and E_k and $D_{\max}$ are calculated as

$$E_k = \sum_{k=1}^{K}\sum_{\boldsymbol{x}_i\in C_k}\|\boldsymbol{x}_i - \boldsymbol{m}_k\| \tag{4}$$

$$D_{\max} = \max_{i,j=1,2,\ldots,K}\left\|\boldsymbol{m}_i - \boldsymbol{m}_j\right\| \tag{5}$$

where $\boldsymbol{m}_k$ denotes the kth cluster centroid, $\boldsymbol{x}_i$ is the ith data sample, and C_k means the kth cluster set. Besides, the power $p > 1$ controls the contrast of the I index for different cluster configurations. The three factors in (3), namely, $1/K$, P/E_k, and $D_{\max}$, have a positive or negative correlation effect on the I index value. Generally, the larger the I index value is, the better the clustering result will be. The objective of the E-DE algorithm is to maximize the I index value in the evolution process.

3) Others: The average intracluster and intercluster distances are commonly adopted to give the details of the clustering results, which can be defined as

$$\text{Intra\_dist} = \sum_{k=1}^{K} \frac{\sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in C_k, j>i} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\frac{1}{2} \times N_{C_k} \cdot (N_{C_k} - 1)} \tag{6}$$

$$\text{Inter\_dist} = \frac{\sum_{i,j=1,2,\ldots,K, j>i} \|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2}{\frac{1}{2} \times K \cdot (K-1)} \tag{7}$$

where K is the number of clusters, C_k denotes the kth cluster, and N_{C_k} is the number of data assigned to the cluster C_k.

Besides, the adjusted rand index (ARI) [47] is a commonly used external index to check clustering accuracy. Given two partitions of a dataset of N samples, namely, $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$, the overlap between X and Y can be summarized in a contingency table $[n_{ij}]$, where each entry n_{ij} denotes the number of objects in common between X_i and Y_j: $n_{ij} = |X_i \bigcap Y_j|$. Then, the ARI is calculated as

$$\text{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{8}$$

where n_{ij}, a_i, and b_j are values from the contingency table. a_i represents the row sum of the contingency table while b_j denotes the column sum of the table.

III. Proposed Algorithm

In this article, we propose an E-DE algorithm to deal with the automatic data clustering problem. Generally, the DE framework is simple and flexible, which suits our proposed elastic encoding and reproduction method. In addition, the literature studies widely validate that DE exhibits powerful performance on continuous numerical optimization as well as real-world applications [48]–[51].

The clustering problem can be defined as follows. Suppose that a dataset $A = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ contains N unlabeled samples, while each sample is represented by a D-dimensional feature vector, namely, $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$. The clustering algorithm aims at partitioning the dataset A into K clusters that are disjoint with each other, which means that each sample in the dataset can only be assigned to one cluster (hard partitions). The partition result is denoted as $\{C_l \mid l = 1, 2, \ldots, K\}$, where the clusters have no intersection and the union of them is equal to the dataset A. We use $\lambda_j \in \{1, 2, \ldots, K\}$ to denote the cluster label for each sample $\boldsymbol{x}_j$, which means that $\boldsymbol{x}_j$ belongs to the cluster C_{λ_j}. Thus, the clustering result can be represented by a cluster label vector, that is, $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_N)$.

The proposed E-DE algorithm follows the framework of the basic differential EA, which evolves the candidate solutions (cluster configurations) under some criteria like the clustering validation indices. In this article, we use the DB index and the I index in the fitness evaluation of individuals. To automatically adapt the number of clusters, an elastic individual encoding

Algorithm 1 E-DE for Automatic Data Clustering

```
1: Randomly generate an initial population with $np$ individuals, each has a
   variable cluster number. (See SectionIII-B)
2: Evaluate the fitness of the individuals in the initial population.
3: Repeat the following steps for each individual $i = 1, 2, ..., np$ until the
   maximum evaluation times is met:
   3.1 Mutation (See Section III-C)
       3.1.1 Randomly select three individuals $X_{r1}$, $X_{r2}$ and $X_{r3}$;
       3.1.2 Determine the number of clusters $K_v$ in the mutant vector
             according to Eq. (8);
       3.1.3 Combine the three individuals to produce the initial mutant
             vector $V_i$;
       3.1.4 Fine-tune the cluster centroids in $V_i$ by the Gaussian location
             mutation in Eq. (9) to obtain the ultimate mutant vector $V_i$.
   3.2 Crossover (See Section III-D)
       3.2.1 Determine the length of crossover by the crossover rate $CR$
             according to Eq. (10);
       3.2.2 Retrieve the sequence of cluster centroids in $V_i$ to determine
             the subspace of crossover;
       3.2.3 Exchange partial cluster layouts between $V_i$ and the individual
             $X_i$ by Eq. (12) to generate the trial vector $U_i$.
   3.3 Selection (See Section III-E)
       3.3.1 Assign the data objects to the nearest centroids in the trial
             vector $U_i$;
       3.3.2 Evaluate the quality of clustering by the DB or I index;
       3.3.3 Preserve $U_i$ or $X_i$ to the next generation according to Eq. (13).
4: Output the partition represented by the best individual.
```

scheme is designed and employed. To meet the variable-length condition in our encoding scheme, we put forward the new crossover and mutation operators which are defined on the spatial layout represented by the individuals. These new reproduction operators also avoid the cross-dimension learning error. After a number of iterations, the algorithm terminates and outputs the global best individual. The overall process of the E-DE algorithm is shown in Algorithm 1. For public use, we provide the source code of E-DE online, which can be downloaded at [52]. The following sections describe different components of E-DE in detail.

A. Elastic Encoding Scheme

The number of clusters influences the clustering performance significantly in considering various types of cluster validation measures. However, how to identify the optimal cluster number remains a challenging issue currently. To fulfill this goal, in the EC-based algorithms, it would be promising to let different individuals possess different cluster numbers. Then, with the evolution of population, because of the survival of fitness effect, the individual with the optimal cluster number would dominate the others and survive to the last. In this way, the optimal cluster number is found. The traditional fixed-length encoding scheme is rigid, while the method of embedding a cluster switch vector works but possesses deficiencies. We adopt an elastic encoding scheme: the population consists of variable-length parameter vectors, each denoting a cluster configuration with a different number of clusters. Illustrated in Fig. 2, unlike the switch mechanism that contains some redundant cluster centroids in the parameter vector [17], [18], our individual possesses only the coordinates of the cluster centroids that take effect in the clustering of data. During the evolution process, the cluster number in the individuals X_1, X_2, and X_3 can be automatically changed, along with the length of the parameter vectors.

B. Population Initialization

Initialization of the population is to generate *np* individuals, each with a specified number of clusters and the corresponding cluster centroids. For $i = 1, 2, \ldots, np$, we first use a random number generator to determine the cluster number K_i, where K_i should comply with the range $[2, 20]$. Afterward, K_i cluster centroids are randomly selected from the origin dataset, which constitutes the parameter vector of individual i. The length of each parameter vector is $D \times K_i$, where D represents the number of features in the dataset. For example, it can be observed from Fig. 2 that for the dataset of three features, the first individual has four clusters ($K_1 = 4$) and, hence, its parameter vector contains 12 dimensions. After randomly selecting the cluster centroids, the first individual becomes (7.8, 2.5, 1.3, 7.5, 2.4, 1.4, 7.2, 2.6, 1.5, 7.6, 2.3, 1.4).

C. Mutation

The traditional DE mutation randomly selects three individuals and then performs some arithmetic combination operation on them, in order to generate a mutant vector. The premise of the method is that the individuals, or parameter vectors, in the population possess the same number of dimensions. In the proposed E-DE, since we develop the variable-length encoding for different individuals, the traditional DE mutation is no longer suitable. Therefore, we design a new mutation operator, which consists of two phases described as follows.

1) Phase One: In the first phase, three individuals X_{r1}, X_{r2}, and X_{r3} are randomly selected from the current population. The number of clusters in the mutant vector V_i is calculated as

$$K_v = K_{r1} + F \times (K_{r2} - K_{r3}) \tag{9}$$

where K_i represents the cluster number of the ith individual, F is the scaling factor, and K_v should be a round number. Note that (8) is similar to the original DE/rand/1 mutation: the random selection of the base vector enhances the global exploration of the algorithm, while the differential length-based perturbation realizes the contour match effect when searching the optimum. This way, the cluster number would be optimized and converged eventually. Besides, we restrict the cluster number in the range of $[2, 20]$. If K_v is smaller than 2, we set $K_v = 2$; otherwise, if K_v is larger than 20, we set $K_v = 20$. After deciding the number of clusters in the mutant vector, the cluster centroids are determined according to the following three situations.

1) $K_v > K_{r1}$, which means that the mutant vector V_i has more clusters than the individual X_{r1}: the cluster centroids of individual X_{r1} are used as the fundamental structure to construct the mutant vector V_i. Then, we need to add $(K_v - K_{r1})$ more cluster centroids to V_i. They are randomly selected from the individuals X_{r2} and X_{r3}. It should be noted that the selected cluster centroids are distinct from each other.
2) $K_v < K_{r1}$, which means that the mutant vector V_i has fewer clusters than the individual X_{r1}: we also start with the cluster centroids of X_{r1} and then randomly deleted $(K_{r1} - K_v)$ cluster centroids from it. The resulting mutant vector V_i contains exactly K_v clusters.

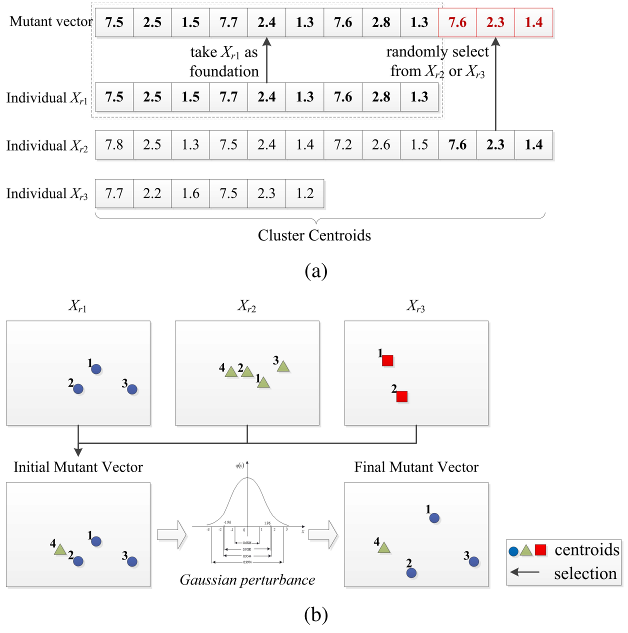

Fig. 3. Mutation process of the E-DE algorithm. (a) Parameter vectors (X_{r1}, X_{r2}, and X_{r3} are individuals randomly selected from the current population). (b) Spatial layouts.

3) $K_v = K_{r1}$, which means that the mutant vector V_i and the individual X_{r1} possess the same number of clusters: in this case, the cluster centroids of individual X_{r1} are directly copied to the mutant vector V_i.

2) Phase Two: By considering the above procedures, the increment or decrement of the cluster centroids comes from the pool of cluster centroids held by the previous population. To alter the centroids, in the second phase of our mutation, the initial mutant vectors further undergo a Gaussian mutation. The operation changes the geographic locations of centroids, which improves the exploration ability of the algorithm. For each cluster centroid that is contained in the mutant vector V_i, a random number $r \in [0, 1]$ is generated. If r is smaller than the mutation probability M_u, the cluster centroid in the mutant vector V_i is adjusted to

$$\boldsymbol{m}_i^k = \boldsymbol{m}_i^k + G(0, 0.1) \times (\boldsymbol{ub} - \boldsymbol{lb}) \tag{10}$$

where $\boldsymbol{m}_i^k$ represents the kth cluster centroid represented by the mutant vector, $G(0, 0.1)$ is a Gaussian random number generator with 0 mean and 0.1 standard deviation, and $\boldsymbol{ub}$ and $\boldsymbol{lb}$ represent the upper and lower bounds of the dataset. The utilization of the Gaussian perturbation considers the following factors: the Gaussian mutation is able to enrich the population diversity without destroying the formulated solution structures. In comparison, if the uniformly random mutation is applied instead, the locations of centroids would change to a large extent, which alters the cluster layout severely. So the Gaussian mutation helps to keep a more well-formed clustering structure and enables the DE algorithm to search the problem in a smooth way.

Fig. 3(a) and (b) shows a mutation example. Suppose that $K_{r1} = 3$, $K_{r2} = 4$, $K_{r3} = 2$, and $F = 0.5$, then the calculated K_v is 4. Since K_v is larger than K_{r1}, one cluster centroid should be selected from the individual X_{r2} or X_{r3} to fill the mutant vector. After that, the location perturbation is further performed on the mutant vector. The new operator not only adapts the cluster number in generating the mutant vector, but also incorporates the centroid distribution information from different individuals in the population. Next, the ultimate mutant vector will intersect with the target vector of individual i by the crossover operator.

D. Crossover

The existing DE crossover methods will lead to the cross-dimension learning error in data clustering. This is because, as illustrated in Fig. 1 and Section I, they exchange the information between the parameter vectors based on the dimension index abruptly, which ignores the spatial relationships of different cluster centroids. Besides, they are unavailable to be directly applied to variable-length parameter vectors. Therefore, we put forward a new crossover operator to solve the above two problems. The crossover can effectively exchange the information between the variable-length individuals without incurring the cross-dimension learning error.

Based on the exponential crossover of DE, we define and incorporate a subspace crossover in E-DE, which is described as follows and illustrated in Fig. 4.

Step a): First, determine the crossover segment for each individual: a random location n is chosen as the starting point and the following L genes will be used for crossover. The crossover length L is determined according to the crossover rate CR as

$$L = L + 1, \text{while } \mathrm{rand}(0, 1) < CR \tag{11}$$

where the initial value of L is zero. The L value determines the length of genes that are going to be swapped in the crossover process. This incremental way has been demonstrated in the original DE algorithm, which is called the exponential crossover [53].

Step b): Retrieve the cluster centroids that are contained in the mutant vector V_i from the selected gene segment in step a). Namely, the first cluster centroid comes from the randomly selected gene location n, while the last centroid is from the gene location $n+L-1$. Based on the retrieved sequence of cluster centroids, the subspace to exchange information in the crossover is defined as follows. First, the center of the subspace, denoted as $\boldsymbol{cs}_i$, is calculated as the average of all the retrieved centroids. Then, suppose that the first and last centroids in the sequence are $\boldsymbol{m}_i^{\text{first}}$ and $\boldsymbol{m}_i^{\text{last}}$, respectively; and the range of the subspace on each dimension is defined as the distance between $\boldsymbol{m}_i^{\text{first}}$ and $\boldsymbol{m}_i^{\text{last}}$ on each dimension

$$d_{i,j} = \left| m_{i,j}^{\text{first}} - m_{i,j}^{\text{last}} \right| \tag{12}$$

where $m_{i,j}^{\text{first}}$ and $m_{i,j}^{\text{last}}$ denote the jth attribute value of the first and last centroids. Particularly, assume that $c_{i,j}$ is the jth coordinate of the center $\boldsymbol{cs}_i$ and that $d_{i,j}$ is the distance between $\boldsymbol{m}_i^{\text{first}}$ and $\boldsymbol{m}_i^{\text{last}}$ on dimension j, and the jth dimension of the subspace ranges from $(c_{i,j} - d_{i,j}/2)$ to $(c_{i,j} + d_{i,j}/2)$.

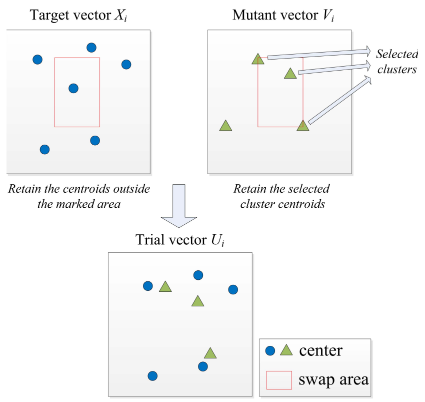

Fig. 4. Illustration of the subspace crossover process of the E-DE algorithm.

We use the distance between $\boldsymbol{m}_i^{\text{first}}$ and $\boldsymbol{m}_i^{\text{last}}$ to let the range of the subspace be more sensitive to the retrieved sequence of cluster centroids. It helps to increase the population diversity. In contrast, if we take the distance between the maximum and minimum value of all retrieved centroids for calculation, different selected gene sequences could correspond to an identical range of subspace. Besides, in the current stage, our E-DE concentrates on continuous data clustering. If the features are discrete, some discrete distance measures (Hamming distance, permutation distance, etc.) should be used according to the practical requirements.

Step c): For the other parameter vector that joins the crossover, that is, the target vector X_i, we use the identical area that is defined in step b) to retrieve the subset of cluster centroids in X_i. Then, the selected cluster centroids of V_i and X_i are exchanged to generate the trial vector U_i

$$U_i = V_i \otimes X_i. \tag{13}$$

The operator $\otimes$ corresponds to the information exchange process in step c), which is illustrated in Fig. 4. Note that the cluster numbers in the crossover area in X_i and V_i could be different. This is so that the cluster number in the generated trial vector could be distinct from the current individual X_i. Therefore, the proposed elastic length encoding scheme, together with the mutation and crossover, is well suited for the automatic data clustering problem that requires adjusting the number of clusters. Besides, if the cluster number in the trial vector U_i exceeds the maximum cluster number $K_{\max} = 20$ or lower than the minimum cluster number $K_{\min} = 2$, it will adopt the cluster number of target vector X_i, but each cluster centroid in the individual X_i will be replaced by the nearest cluster centroid in the mutant vector V_i.

In addition to being capable of adjusting the cluster numbers, the proposed subspace crossover inherently avoids the cross-dimension learning error since it considers the spatial relationship between clusters. Note that the target and mutant vectors represent different spatial layouts of cluster centroids, as depicted in Fig. 4. In the crossover process, first, a subspace is selected as the swap area and, then, the target and mutant vectors exchange their subsets of cluster centroids that are located within the subspace. This mechanism ensures that the exchanged clusters have a spatial relationship, which

TABLE I
PROPERTIES OF 19 DATASETS USED IN THE CLUSTERING EXPERIMENTS

Datasets	# Samples	# Features	# Clusters (Ground truth)
Iris	50	3	3
Cancer	683	9	2
Glass	214	9	6
Vowel	871	3	6
Zoo	101	7	7
Vote	435	16	2
Ecoli	336	7	8
Seed	210	7	3
Compound	339	2	6
R15	600	2	15
Dim2	1351	2	9
G2	2048	2	2
Dim32	1024	32	16
Customer	440	6	6
Unbalance	6500	2	8
Spiral	312	2	3
Mockdata1	1000	2	4
Mockdata2	1000	2	2
Mockdata3	600	2	4

are different from the traditional DE algorithms that perform crossover through the cluster indices. Thus, the generated trial vector inherits different cluster centroids from the target or mutant vectors from a spatial point of view. To summarize, the subspace crossover not only fits our elastic encoding scheme but also avoids the cross-dimension learning error of the traditional DE algorithms.

E. Selection

To perform selection first, the fitness of the trial vector U_i is calculated. A data assignment step is required when the cluster centroids are determined. For each data point in the dataset, calculate its distance to all cluster centroids in U_i. Then, each data object is assigned to its nearest cluster centroid.

Note that sometimes there may exist some clusters which contain less than 2 data points, which are considered empty clusters. The empty clusters will affect the well-formed cluster structure. However, if they are simply removed, the population would have a hidden property to reduce the total number of clusters during optimization, which goes against our intention. Therefore, we use a random reassignment strategy [17] instead. Particularly, for the individual that contains an empty cluster, it is rearranged according to the following procedure. Select N/K data points from the dataset, use the mean of the data points as the new cluster centroid, and redo the data assignment. Repeat this step until no empty cluster comes up. Then, update the individual accordingly.

After the data assignment step, the dataset is partitioned into clusters. The validation indices can be applied to evaluate the quality of the partition. Since how to design a good validation index for clustering is beyond the scope of this article, here, we simply adopt the well-known DB index and I index described in Section II-B. The selection of E-DE is identical to the classical DE selection, namely, the algorithm compares the fitness of the current individual X_i and the trial vector U_i, and it selects the better one to the next generation

$$X_{i,G+1} = \begin{cases} X_{i,G}, & \text{if } f(X_{i,G}) \text{ is better than } f(U_{i,G}) \\ U_{i,G}, & \text{otherwise.} \end{cases} \tag{14}$$

Note that if the DB index is applied, the smaller the f value is, the better the clustering results are. Differently, for the I index, the higher f value is preferred.

IV. EXPERIMENTS

A. Experimental Setup

In this section, we conduct several experiments to validate the performance of the proposed E-DE algorithm. The experiments are based on 19 datasets (summarized in Table I) that are widely adopted to examine the state-of-the-art clustering algorithms [29], [54]–[61]. These datasets possess various characteristics. The Iris, Cancer, Glass, Zoo, Vote, Ecoli, Seed, and Customer datasets are standard datasets coming from the UCI machinery repository [54]. In addition, the Compound [55] is a graphical-shaped dataset with six clusters and 339 data objects in two dimensions. The R15 [56] contains Gaussian clusters in a ringed region, with 600 data objects in total. The Dim2 dataset [57] is a synthetic clump including nine clusters, with 1351 data objects in two dimensions. The G2 dataset [58] is a typical 2-class collection, with 2048 data in two dimensions. The Dim32 [59] is a high-dimensional dataset that contains 32 dimensions, with 1024 data objects in total. The Unbalance dataset [60] has eight clusters of quite different sizes and densities, with 6500 data objects in total. The Spiral [61] is a nonconvex dataset composed of three different trajectories of the spiral, with 312 data objects in two dimensions. Further, we test three mock datasets, that is, Mockdata1, Mockdata2, and Mockdata3. They are produced according to three functions that are written by Jeroen Kools to generate different pathological clustering datasets. Commonly, the centroid-based clustering algorithms can hardly handle nonconvex datasets like the spiral. The nonconvex datasets are tested in this article for the purpose of investigations.

We compare the E-DE algorithm with six representative clustering algorithms, namely, the K-means [10], BF-K-means [44], S-DE [24], ACDE [17], MANM [35], and TGCA [37]. The K-means algorithm requires a predefined cluster number as input to partition the dataset into K clusters. The brute-force K-means (BF-K-means) is to run K-means several times varying the number of clusters from the $K_{\min}$ to $K_{\max}$ (i.e., 2 to 20), and select the best clustering solution. The S-DE algorithm is a global optimization algorithm, while it also requires the cluster number as the prior condition. The ACDE algorithm adopts a relatively automatic parameter vector encoding scheme, adjusting the cluster number through the zero-one switch. The MANM algorithm takes the memetic algorithm as a framework, and it utilizes the multilocal search and adaptive niching technique for automatic data clustering. The TGCA algorithm adopts the two-stage selection and mutation operations to adjust the cluster number and centroids. The individuals that have the same cluster number will be partitioned into the same subpopulation to perform crossover. In the experiments, we adopt the parameter settings that are suggested in their original references. Besides, all EC algorithms are assigned a maximum number of fitness evaluation times, 10^6. Both the K-means and BF-K-means algorithms are terminated when the cluster centroids are not updated.

For the E-DE algorithm, the size of population np is set to $10 \times D$, where D is the data dimensions. The minimum and maximum cluster numbers are set to 2 and 20. The scalar factor F and the crossover rate CR are set to 0.5 and 0.4, while

TABLE II
COMPARING THE CLUSTERING RESULTS (MEAN AND STANDARD DEVIATION) OF THE E-DE ALGORITHM WITH THE OTHER SIX CLUSTERING ALGORITHMS USING THE DB INDEX. NOTE: THE CORRECT CLUSTER NUMBER IS PROVIDED UNDER THE NAME OF EACH DATASET

Problem		*K*-means	S-DE	BF-*K*-means	ACDE	MANM	TGCA	E-DE
	#Counts	6	6	3	3	2	3	**3**
Iris	Avg. Fitness	0.875±0.036	0.824±0.023	0.699±0.028	0.655±0.128	0.429±0.021	0.663±0.175	**0.612±0.086**
(3)	Intra Distance	0.753±0.037	0.709±0.013	1.228±0.017	0.855±0.198	1.083±0.021	1.073±0.029	**1.029±0.227**
	Inter Distance	2.607±0.386	2.754±0.177	3.374±0.016	4.094±0.616	4.647±0.714	4.293±0.528	**4.556±1.038**
	Avg. ARI	0.393	0.312	0.877	**0.926**	0.521	0.924	0.923
	#Counts	6	6	4(27),3(3)	2(28),4(2)	2	2	**2**
Cancer	Avg. Fitness	1.386±0.227	1.703±0.043	1.640±0.033	1.766±0.041	1.285±0.017	0.932±0.018	**0.905±0.024**
(2)	Intra Distance	8.905±0.078	7.936±0.227	7.199±0.275	8.249±0.895	9.885±0.284	9.048±0.757	**8.633±0.589**
	Inter Distance	11.983±1.542	11.776±0.994	11.190±0.364	13.459±1.918	14.206±0.897	14.337±0.268	**21.721±1.564**
	Avg. ARI	0.328	0.215	0.253	0.847	**0.953**	0.908	0.931
	#Counts	6	6	2(25),4(3),5(2)	6(25),5(5)	4(28),6(2)	6(27),5(3)	**6**
Glass	Avg. Fitness	1.083±0.147	1.222±0.279	1.257±0.037	0.884±0.226	0.965±0.095	0.802±0.093	**0.728±0.081**
(6)	Intra Distance	3.363±0.389	2.059±0.336	1.026±0.045	2.841±0.849	3.831±0.190	3.928±0.132	**2.310±0.650**
	Inter Distance	4.801±0.397	4.826±0.215	3.668±0.113	7.924±1.201	5.931±0.249	6.423±0.670	**6.840±1.945**
	Avg. ARI	0.928	**0.938**	0.409	0.838	0.321	0.792	0.906
	#Counts	6	6	5(26),2(4)	6(24),5(4),4(1),2(1)	**6**	6(22),5(8)	6(27),3(3)
Vowel	Avg. Fitness	1.843±0.329	1.261±0.127	0.823±0.097	0.793±0.028	**0.832±0.118**	0.740±0.013	0.704±0.068
(6)	Intra Distance	223.498±34.667	223.724±3.598	152.772±2.086	294.238±66.832	**232.934±32.384**	295.430±55.321	342.241±78.759
	Inter Distance	792.036±325.327	651.392±149.922	729.311±12.334	1429.426±442.988	**945.664±183.294**	1163.427±356.298	1345.429±302.392
	Avg. ARI	0.969	**0.984**	0.348	0.904	0.853	0.812	0.843
	#Counts	6	6	2(15),7(8),3(7)	**7**	4	4(22),7(5),3(3)	7(28),4(2)
Zoo	Avg. Fitness	1.218±0.170	0.929±0.084	0.492±0.060	**0.888±0.173**	0.851±0.090	0.820±0.046	0.625±0.023
(7)	Intra Distance	2.910±0.387	1.929±0.322	0.635±0.053	**1.732±0.478**	1.322±0.136	1.664±0.181	1.392±0.067
	Inter Distance	5.398±0.896	5.927±1.019	4.628±0.224	**6.327±0.921**	5.553±0.421	5.663±0.389	5.945±0.455
	Avg. ARI	0.184	0.256	0.503	0.863	0.124	0.464	**0.913**
	#Counts	6	6	4(25),2(5)	6(24),8(6)	2	2	**2**
Vote	Avg. Fitness	2.326±0.220	1.321±0.037	1.769±0.030	1.322±0.145	1.540±0.177	1.138±0.129	**0.936±0.037**
(2)	Intra Distance	1.773±0.013	1.425±0.032	1.552±0.028	1.782±0.056	1.922±0.050	1.957±0.086	**2.122±0.018**
	Inter Distance	1.913±0.330	2.643±0.063	2.008±0.036	2.864±0.328	3.325±0.423	2.821±0.279	**3.354±0.061**
	Avg. ARI	0.190	0.015	0.138	0.119	0.881	0.824	**0.969**
	#Counts	6	6	7(23),8(7)	3(17),7(8),5(3),6(2)	4(25),2(3),3(2)	7(25),11(5)	**8(25),9(4),11(1)**
Ecoli	Avg. Fitness	1.322±0.072	0.951±0.039	0.933±0.033	0.932±0.067	0.920±0.087	0.935±0.076	**0.891±0.024**
(8)	Intra Distance	0.352±0.053	0.325±0.060	0.257±0.053	0.333±0.031	0.334±0.032	0.333±0.042	**0.324±0.053**
	Inter Distance	0.589±0.132	0.823±0.056	0.7941±0.051	1.324±0.223	1.133±0.132	0.854±0.052	**0.818±0.026**
	Avg. ARI	0.313	0.338	0.422	0.379	0.386	0.219	**0.796**
	#Counts	6	6	4	6(24),5(6)	3	**3**	2
Seed	Avg. Fitness	1.329±0.024	0.878±0.031	0.729±0.019	0.831±0.058	0.721±0.089	**0.712±0.020**	0.643±0.056
(3)	Intra Distance	1.633±0.032	1.643±0.010	0.824±0.067	2.012±0.196	2.518±0.225	**2.039±0.038**	2.737±0.323
	Inter Distance	5.097±0.323	5.325±0.120	0.535±0.128	6.931±0.631	6.939±0.332	**7.321±0.432**	9.621±1.495
	Avg. ARI	0.446	0.272	0.563	0.537	**0.934**	0.877	0.752
	#Counts	6	6	6(28),2(2)	6	2(26),5(2),6(2)	6	**6**
Compound	Avg. Fitness	0.725±0.054	0.889±0.035	0.633±0.053	0.733±0.078	0.565±0.042	0.634±0.03	**0.622±0.043**
(6)	Intra Distance	3.222±0.311	4.425±0.178	2.560±0.232	3.952±0.639	4.132±0.432	4.924±1.343	**4.329±1.520**
	Inter Distance	13.062±2.521	16.518±1.832	12.447±0.713	17.997±2.394	16.187±1.982	18.432±1.132	**20.292±7.936**
	Avg. ARI	0.823	**0.893**	0.826	0.847	0.831	0.853	0.673
	#Counts	6	6	15	15	15	15	**15**
R15	Avg. Fitness	0.633±0.036	0.923±0.030	0.606±0.038	0.422±0.116	0.493±0.091	0.443±0.018	**0.419±0.023**
(15)	Intra Distance	1.202±0.016	2.532±0.000	0.441±0.020	0.673±0.234	0.530±0.021	0.591±0.095	**0.540±0.000**
	Inter Distance	8.321±0.311	8.087±0.010	5.321±0.179	6.123±1.209	5.327±0.394	5.319±0.319	**5.367±0.032**
	Avg. ARI	0.578	0.592	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	#Counts	6	6	14(22),3(8)	8(20),9(10)	8	8	**9**
Dim2	Avg. Fitness	0.333±0.081	0.351±0.005	0.227±0.006	0.224±0.012	0.261±0.014	0.231±0.019	**0.105±0.012**
(9)	Intra Distance	60131.285±2032.591	63194.524±133.360	6254.236±50.350	24293.329±8319.015	31422.125±2032.458	26931.431±2432.266	**12994.333±10.030**
	Inter Distance	615825.672±2592.429	603412.329±5931.429	585041.899±18.732	532011.394±8429.425	579915.434±2520.325	590915.429±7320.429	**625253.304±3420.588**
	Avg. ARI	0.682	0.661	0.537	0.674	0.303	0.269	**1.000**
	#Counts	6	6	2(27),4(3)	4(15),2(12),3(3)	2	**2**	2
G2	Avg. Fitness	1.149±0.072	0.759±0.000	0.507±0.252	0.278±0.029	0.218±0.049	**0.202±0.033**	0.887±0.018
(2)	Intra Distance	13.279±0.593	13.265±0.000	6.766±0.180	14.933±2.825	16.210±0.359	**15.186±0.452**	2.500±0.053
	Inter Distance	88.745±5.893	118.156±0.000	80.428±2.664	115.443±24.52	132.344±8.429	**133.389±12.556**	158.303±11.483
	Avg. ARI	0.210	0.422	0.931	0.864	**1.000**	**1.000**	**1.000**
	#Counts	6	6	17(23),16(7)	12	16	12	**16**
Dim32	Avg. Fitness	1.397±0.055	1.274±0.031	0.195±0.054	0.545±0.270	0.143±0.049	0.329±0.032	**0.139±0.026**
(16)	Intra Distance	190.411±7.183	268.838±0.000	41.006±1.574	59.136±35.227	17.429±9.233	56.894±45.429	**18.997±0.000**
	Inter Distance	366.887±21.253	520.513±0.836	355.759±11.510	448.532±51.679	430.300±15.235	512.427±44.832	**423.329±0.996**
	Avg. ARI	0.353	0.489	0.570	0.234	**1.000**	0.393	**1.000**
	#Counts	**6**	6	2(21),9(9)	9(24),7(6)	4	6(18),9(7),7(5)	6(26),7(4)
Customer	Avg. Fitness	**1.105±0.085**	1.407±0.126	0.866±0.027	0.933±0.147	0.872±0.094	0.843±0.062	0.771±0.130
(6)	Intra Distance	**20770.128±1253.367**	13977.385±1444.357	14527.325±1529.562	23083.718±3792.921	24295.530±1324.526	31457.299±3632.592	33952.631±3041.882
	Inter Distance	**42935.507±3460.251**	24403.229±4488.728	43242.930±2308.265	65299.427±9234.252	73259.319±3391.425	83295.053±2325.530	83266.342±13212.389
	Avg. ARI	0.857	0.904	0.479	0.172	0.313	**0.921**	0.904
	#Counts	6	6	3	4	4	4	**4**
Unbalance	Avg. Fitness	0.790±0.070	0.583±0.022	0.459±0.022	0.304±0.042	0.329±0.019	0.394±0.019	**0.276±0.028**
(8)	Intra Distance	19142.155±2192.667	17203.035±332.292	6249.331±813.769	24951.925±10422.391	23503.259±2135.447	22942.429±3809.427	**24610.960±0.000**
	Inter Distance	122007.214±4542.056	134251.439±2425.120	179398.534±12766.332	132522.432±28423.399	183247.242±13925.240	183294.593±14305.503	**203826.326±10329.566**
	Avg. ARI	0.340	0.472	**0.557**	0.393	0.552	0.484	0.397
	#Counts	6	6	2	6	**3**	3(22),4(8)	3(25),4(3),2(2)
Spiral	Avg. Fitness	0.958±0.027	0.936±0.024	0.841±0.023	0.837±0.042	**0.803±0.520**	0.829±0.030	0.827±0.032
(3)	Intra Distance	4.936±0.218	5.751±0.167	5.799±0.264	6.473±1.253	**7.543±0.399**	6.133±0.328	4.456±1.321
	Inter Distance	12.570±0.639	14.034±2.179	12.705±0.181	20.426±1.325	**20.473±0.638**	19.325±0.443	18.328±2.030
	Avg. ARI	0.153	0.150	0.113	0.115	0.682	**0.730**	0.663
	#Counts	6	6	**4**	3	2	3	4
Mockdata1	Avg. Fitness	0.953±0.034	0.733±0.033	**0.560±0.018**	0.683±0.050	0.754±0.015	0.733±0.033	0.629±0.026
(4)	Intra Distance	4.298±0.056	3.326±0.024	**1.682±0.084**	6.432±0.066	8.135±0.089	6.832±0.032	1.738±0.022
	Inter Distance	13.379±1.448	12.426±0.557	**12.974±0.173**	14.428±1.045	23.324±1.081	17.214±1.036	12.323±0.568
	Avg. ARI	0.389	0.534	**0.917**	0.888	0.542	0.563	0.778
	#Counts	6	6	6	4	**2**	4	6
Mockdata2	Avg. Fitness	0.928±0.024	0.747±0.019	0.878±0.041	0.593±0.033	**0.776±0.038**	0.643±0.055	0.628±0.023
(2)	Intra Distance	3.373±0.041	3.325±0.079	3.792±0.061	4.342±0.067	**7.465±0.088**	3.537±0.045	3.463±0.070
	Inter Distance	13.561±0.068	13.325±1.457	14.097±1.067	17.426±1.766	**18.336±1.537**	14.838±1.427	15.942±0.642
	Avg. ARI	0.447	0.513	0.463	0.581	**0.938**	0.533	0.229
	#Counts	6	6	4	4	4	4	**4**
Mockdata3	Avg. Fitness	0.753±0.039	0.843±0.016	0.741±0.067	0.833±0.056	0.732±0.022	0.753±0.070	**0.657±0.042**
(4)	Intra Distance	7.103±0.368	8.154±0.069	4.208±0.160	8.537±0.043	15.427±1.296	6.232±0.563	**5.778±0.084**
	Inter Distance	32.515±0.334	32.538±1.023	32.836±2.839	43.549±1.326	43.532±1.759	32.545±1.633	**33.359±1.255**
	Avg. ARI	0.513	0.664	0.848	0.844	**0.959**	0.853	0.838
Total CSR		N/A	N/A	40.702%	43.684%	48.070%	58.596%	**80.877%**
Total ARI		0.479	0.507	0.550	0.622	0.689	0.711	**0.813**

the mutation rate M_u varies with the iteration times

$$M_u = 0.1 - 0.06 \times \frac{G}{G_{\max}} \tag{15}$$

where $G_{\max}$ means the maximum generations, that is, $G_{\max} = 10^6/np$, and G stands for the present generation in the evolution process. All of the simulation results presented were obtained through a PC with an Intel Core CPU at 3.30 GHz running the Ubuntu operating system. In case of the incidental results, we repeat each algorithm 30 times on all 19 datasets.

B. Experimental Results

1) Validation by the DB Index: First, we use the DB index to examine seven clustering algorithms. Table II shows the mean clustering results with the standard deviations, including the cluster number counted in each runtime (#counts, where the number outside the parenthesis means the cluster number and the number inside the parenthesis indicates the frequency of this number), the average DB index value, the intercluster distance, the intracluster distance, and the ARI. In addition, at the bottom of Table II, we summarize the total clustering successful ratio (CSR), that is, the ratio of the run times that the algorithm finds the correct cluster number to the total run times, as well as the total ARI. For *K*-means and S-DE, they use a fixed number of clusters, which is set to 6. This is because the practical number of clusters cannot be used in automatic data clustering, while 6 is the most common and the median value of the cluster number according

TABLE III
COMPARING THE CLUSTERING RESULTS (MEAN AND STANDARD DEVIATION) OF THE E-DE ALGORITHM WITH THE OTHER SIX CLUSTERING ALGORITHMS USING THE I INDEX. NOTE: THE CORRECT CLUSTER NUMBER IS PROVIDED UNDER THE NAME OF EACH DATASET

Problem		*K*-means	S-DE	BF-*K*-means	ACDE	MANM	TGCA	E-DE
	#Counts	6	6	3	**3**	3	3	3
Iris	Avg. Fitness	3.238±0.164	5.143±0.032	6.640±0.277	**7.239±1.012**	4.449±0.330	5.310±0.533	5.329±0.433
(3)	Intra Distance	0.734±0.029	0.633±0.031	0.933±0.053	**0.842±0.067**	1.032±0.019	0.854±0.066	0.833±0.054
	Inter Distance	2.323±0.325	3.537±0.032	3.587±0.187	**4.642±0.547**	2.039±0.420	4.033±0.027	3.777±0.529
	Avg. ARI	0.347	0.372	0.906	0.922	0.914	0.929	**0.953**
	#Counts	6	6	2	2	2	2	**2**
Cancer	Avg. Fitness	0.417±0.215	0.724±0.143	1.187±0.050	2.053±1.449	1.277±0.287	1.783±0.550	**2.338±0.640**
(2)	Intra Distance	8.220±0.045	7.239±0.053	8.785±0.042	9.323±0.405	10.662±0.153	9.326±0.523	**9.796±0.380**
	Inter Distance	10.723±1.169	13.436±0.564	12.032±0.270	18.325±3.733	15.829±1.435	20.067±2.325	**21.721±3.659**
	Avg. ARI	0.249	0.313	0.827	0.924	0.892	0.875	**0.929**
	#Counts	6	**6**	7(28),6(2)	2(22),4(8)	3(24),4(6)	4(24),6(6)	6(28),5(2)
Glass	Avg. Fitness	2.321±0.442	**4.248±1.321**	2.132±0.029	6.223±1.420	4.126±0.420	6.517±0.220	3.325±0.367
(6)	Intra Distance	2.122±0.432	**2.324±0.560**	1.933±0.012	2.121±0.750	1.546±0.223	2.333±0.341	2.313±0.420
	Inter Distance	5.320±0.442	**6.620±0.320**	5.432±0.325	9.323±0.630	8.329±0.524	9.424±1.534	7.132±0.557
	Avg. ARI	0.848	**0.906**	0.550	0.215	0.238	0.360	0.860
	#Counts	6	**6**	2	5(22),6(8)	4	5	5(22),4(8)
Vowel	Avg. Fitness	1.703±0.132	**3.450±0.227**	2.929±0.103	4.032±0.323	3.222±0.324	3.332±0.145	3.814±0.547
(6)	Intra Distance	232.325±3.120	**246.425±12.430**	387.453±59.450	291.545±52.391	312.324±24.030	259.314±22.830	253.526±12.556
	Inter Distance	738.524±32.670	**1149.625±42.535**	942.374±70.265	1193.472±128.566	1489.125±32.240	1207.643±41.250	1232.332±66.430
	Avg. ARI	0.887	**0.938**	0.270	0.563	0.297	0.263	0.385
	#Counts	6	6	3(24),6(6)	3(12),5(10),6(8)	4(23),2(7)	**5(21),7(9)**	4(22),3(8)
Zoo	Avg. Fitness	0.846±0.067	1.643±0.140	1.162±0.069	1.514±0.261	1.222±0.082	**1.862±0.009**	1.761±0.062
(7)	Intra Distance	1.756±0.162	1.988±0.106	2.513±0.380	2.414±0.494	3.654±0.166	**2.278±0.156**	2.360±0.293
	Inter Distance	4.367±0.339	5.768±0.293	5.430±0.417	6.646±0.699	5.700±0.259	**6.574±0.724**	6.728±0.733
	Avg. ARI	0.2400	0.234	0.313	0.287	**0.526**	0.369	0.246
	#Counts	6	6	2	2	2	2	**2**
Vote	Avg. Fitness	0.188±0.078	0.124±0.142	0.718±0.030	0.584±0.236	0.473±0.040	0.601±0.019	**1.328±0.025**
(2)	Intra Distance	1.921±0.014	1.552±0.012	2.017±0.042	1.584±0.042	2.048±0.023	1.270±0.088	**2.025±0.006**
	Inter Distance	1.732±0.106	2.433±0.120	2.394±0.044	2.929±0.390	2.108±0.093	2.489±0.220	**3.953±0.048**
	Avg. ARI	0.129	0.163	0.891	0.860	0.906	0.891	**0.992**
	#Counts	6	6	2(27),3(3)	**3(17),4(9),7(4)**	5(22),2(8)	5(20),4(10)	4(21),5(9)
Ecoli	Avg. Fitness	0.374±0.157	1.443±0.064	0.976±0.032	**2.317±0.527**	2.045±0.035	1.547±0.067	1.354±0.230
(8)	Intra Distance	0.302±0.025	0.323±0.020	0.364±0.058	**0.365±0.094**	0.374±0.018	0.368±0.054	0.324±0.022
	Inter Distance	0.523±0.024	0.742±0.024	0.555±0.032	**1.202±0.328**	0.942±0.029	0.933±0.051	0.923±0.087
	Avg. ARI	0.408	**0.594**	0.156	0.146	0.112	0.231	0.262
	#Counts	6	6	**3**	3(20),4(6),6(4)	3	3(24),6(6)	3
Seed	Avg. Fitness	1.438±0.042	2.435±0.080	**2.946±0.023**	3.379±0.321	2.329±0.064	2.901±0.072	2.845±0.462
(3)	Intra Distance	1.626±0.044	1.622±0.007	**2.234±0.019**	1.976±0.230	2.424±0.024	1.921±0.052	1.953±0.244
	Inter Distance	5.073±0.124	5.832±0.015	**5.334±0.112**	7.533±0.244	7.325±0.045	7.243±0.235	6.329±0.576
	Avg. ARI	0.154	0.187	0.919	0.694	**0.942**	0.932	0.855
	#Counts	6	6	2	6	2	**6**	6
Compound	Avg. Fitness	2.772±0.173	3.823±0.443	3.592±0.033	4.189±0.232	4.551±0.018	**4.223±0.322**	3.825±0.630
(6)	Intra Distance	3.235±0.127	3.796±0.162	6.722±0.053	4.422±0.740	5.603±0.223	**4.672±0.529**	4.282±0.813
	Inter Distance	15.233±0.344	18.257±0.653	19.191±0.688	21.324±0.931	24.465±0.331	**23.627±0.240**	22.514±3.253
	Avg. ARI	0.784	0.828	0.228	**0.912**	0.674	0.836	0.540
	#Counts	6	6	16(27),12(3)	15(27),16(3)	**15**	15(26),16(3),17(1)	15(28),17(2)
R15	Avg. Fitness	1.382±0.353	1.923±0.029	3.824±0.775	3.726±1.325	**3.438±0.425**	3.925±0.132	4.133±0.268
(15)	Intra Distance	1.694±0.134	1.773±0.038	0.446±0.032	1.325±0.932	**0.539±0.074**	0.573±0.064	0.533±0.012
	Inter Distance	8.176±0.420	9.328±0.016	5.854±0.035	8.226±2.453	**5.923±0.161**	5.374±0.123	6.268±0.132
	Avg. ARI	0.149	0.125	0.688	0.924	**1.000**	0.734	0.982
	#Counts	6	6	12(20),14(10)	9	9	9	**9**
Dim2	Avg. Fitness	3.824±0.363	3.148±0.426	62.133±11.560	127.536±56.743	7.226±0.266	115.433±8.327	**128.747±7.337**
(9)	Intra Distance	64322.933±171.383	74171.311±0.000	8548.503±230.599	13297.671±0.000	13297.671±0.000	13297.671±0.000	**13297.671±0.000**
	Inter Distance	632561.326±1215.367	627468.325±1513.632	612923.753±256.326	632648.468±5235.278	572125.333±524.685	612566.126±2215.332	**617739.822±1292.223**
	Avg. ARI	0.035	0.203	0.469	**1.000**	**1.000**	**1.000**	**1.000**
	#Counts	6	6	2	2	**2**	2	2
G2	Avg. Fitness	8.367±1.624	14.377±0.000	31.137±3.520	28.359±5.052	**31.166±1.231**	29.574±2.481	22.855±3.971
(2)	Intra Distance	11.919±0.323	17.631±0.000	17.947±0.424	17.509±0.566	**17.947±0.339**	17.654±0.427	17.020±1.258
	Inter Distance	84.314±3.212	110.382±0.000	141.804±9.737	145.787±2.046	**141.980±4.162**	148.656±12.676	146.253±15.799
	Avg. ARI	0.106	0.143	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	#Counts	6	6	2(18),5(8),6(4)	3(22),4(5),16(3)	16	3	**16**
Dim32	Avg. Fitness	0.149±0.071	0.366±0.057	0.195±0.090	1.464±0.173	3.171±0.582	2.681±0.761	**5.366±1.936**
(16)	Intra Distance	174.279±21.999	225.128±0.000	154.321±13.892	186.777±12.219	18.997±0.000	182.634±16.754	**18.997±0.000**
	Inter Distance	357.748±34.157	505.580±0.469	369.233±81.479	689.055±181.976	425.219±9.228	626.308±154.728	**438.504±7.786**
	Avg. ARI	0.177	0.313	0.444	0.131	**1.000**	0.141	**1.000**
	#Counts	6	6	7(24),6(6)	3(25),6(4),7(1)	5(26),6(4)	6(27),7(3)	**6(27),5(3)**
Customer	Avg. Fitness	2.857±1.174	8.683±2.584	2.356±0.049	13.457±3.660	20.599±0.830	14.536±0.440	**10.478±2.640**
(6)	Intra Distance	20740.059±2465.931	22543.934±3215.112	23649.710±3060.238	31937.862±7486.158	61684.642±2429.622	33259.212±6538.291	**31125.805±8602.349**
	Inter Distance	42848.348±8671.446	68462.293±13843.452	44800.441±2459.566	999559.105±24283.764	129627.096±8596.391	126137.721±2334.650	**97819.753±25557.605**
	Avg. ARI	**0.935**	0.915	0.529	0.238	0.332	0.834	0.845
	#Counts	6	6	4	5(24),6(5),8(1)	4	4	**8(26),7(4)**
Unbalance	Avg. Fitness	54.810±7.132	57.242±0.674	107.451±29.880	96.515±34.284	68.115±2.772	91.328±2.336	**108.676±5.449**
(8)	Intra Distance	17480.289±1136.566	17502.468±113.471	24610.960±669.354	18479.130±2864.790	24681.605±1007.485	20573.820±3146.750	**17912.095±3443.909**
	Inter Distance	131277.974±10129.727	137691.055±1356.423	179398.534±966.687	204641.800±4844.924	176973.351±3714.590	213659.393±5748.629	**230571.694±11320.226**
	Avg. ARI	0.244	0.672	0.234	0.481	0.234	0.234	**0.898**
	#Counts	6	6	2(25),4(5)	5(22),4(4),3(4)	**3**	5(25),4(4),3(1)	5(15),4(8),3(7)
Spiral	Avg. Fitness	0.616±0.108	1.886±0.329	0.733±0.078	2.118±0.348	**1.977±0.043**	1.863±0.082	1.617±0.253
(3)	Intra Distance	5.649±0.341	5.329±0.009	9.932±0.091	6.578±0.998	**7.794±0.272**	7.525±0.385	7.251±0.897
	Inter Distance	13.845±1.610	18.819±1.227	12.227±0.133	22.592±3.051	**23.750±0.786**	23.032±1.534	20.918±2.560
	Avg. ARI	0.114	0.093	0.379	0.267	**0.683**	0.442	0.109
	#Counts	6	6	4(28),6(2)	3	2	3	**4**
Mockdata1	Avg. Fitness	0.874±0.023	2.439±0.037	1.081±0.034	2,318±0.066	2.079±0.044	2.158±0.048	**2.010±0.018**
(4)	Intra Distance	3.999±0.001	3.324±0.061	3.910±0.074	7.110±0.082	8.045±0.093	7.481±0.059	**5.515±0.042**
	Inter Distance	11.164±0.187	15.449±0.302	11.368±0.175	22.578±0.139	22.585±0.370	22.280±0.340	**17.695±0.203**
	Avg. ARI	0.524	0.375	0.929	0.447	0.529	0.473	**0.938**
	#Counts	6	6	5	**4**	3	5	5(28),4(2)
Mockdata2	Avg. Fitness	2.497±0.015	2.682±0.072	2.505±0.049	**3.112±0.017**	2.491±0.065	2.495±0.029	2.569±0.020
(2)	Intra Distance	3.452±0.028	3.647±0.049	4.036±0.067	**4.796±0.088**	8.793±0.056	4.286±0.055	4.058±0.036
	Inter Distance	14.127±0.139	16.864±0.415	14.174±0.233	**17.280±0.257**	27.135±0.228	17.083±0.728	17.769±0.467
	Avg. ARI	0.239	0.313	0.326	0.494	0.369	0.487	**0.524**
	#Counts	6	6	2	2	2	4	**4**
Mockdata3	Avg. Fitness	2.316±0.025	2.949±0.048	2.672±0.060	3.831±0.062	3.826±0.073	2.582±0.045	**3.033±0.079**
(4)	Intra Distance	7.198±0.026	7.886±0.036	16.894±0.365	17.038±0.074	17.354±0.031	7.953±0.029	**13.056±0.084**
	Inter Distance	37.796±1.314	44.854±1.784	39.899±7.822	55.081±1.936	60.184±1.273	48.309±1.458	**46.747±1.789**
	Avg. ARI	0.472	0.502	0.389	0.624	0.704	0.968	**0.971**
Total CSR		N/A	N/A	32.632%	44.386%	48.070%	53.158%	**72.982%**
Total ARI		0.370	0.431	0.550	0.586	0.650	0.632	**0.752**

to our datasets in Table I. Unlike the traditional *K*-means algorithm, the BF-*K*-means is conducted with different cluster numbers and chooses the best result as the solution. The other four algorithms, namely, ACDE, MANM, TGCA, and the proposed E-DE, automatically adapt the cluster number during optimization. The best result of each dataset is selected and marked in boldface according to the correctness of #counts and the average DB index value. Meanwhile, the best result according to the ARI is also marked in boldface.

It can be observed that our E-DE algorithm exhibits better performance when comparing other data clustering algorithms on most datasets. E-DE can always find the correct number of clusters where it reaches approximately 81% clustering accuracy on the 19 datasets. For the compared automatic data clustering algorithms, we can observe that the ACDE algorithm fails to find the correct cluster number in many cases. The ACDE is sometimes unstable that its clustering results in different runs vary widely, as can be seen from the #counts for the Vowel, Vote, Ecoli, Dim2, G2, and Customer datasets. For the MANM column in Table II, the memetic algorithm has similar clustering accuracy with the ACDE. Its local search strategies consume many fitness evaluations, which leads the algorithm to exploit too much at the beginning. For the two-stage TGCA algorithm, its performance is slightly better than the MANM. But the crossover way that was used still has the cross-dimension learning error that the clustering accuracy

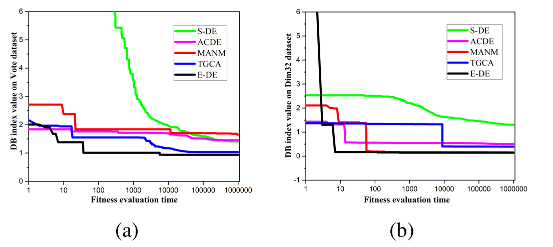

Fig. 5. DB index convergence curves of S-DE, ACDE, MANM, TGCA, and E-DE algorithms. (a) Vote dataset. (b) Dim32 dataset.

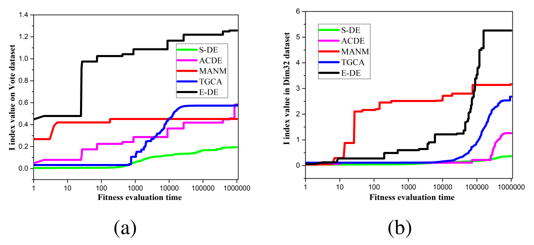

Fig. 6. I index convergence curves of S-DE, ACDE, MANM, TGCA, and E-DE algorithms. (a) Vote dataset. (b) Dim32 dataset.

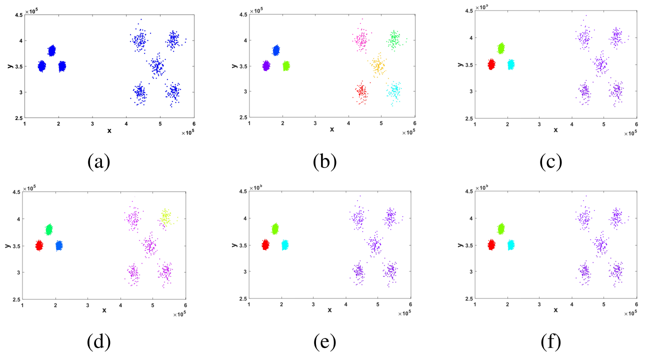

Fig. 7. Clustering results of Unbalance dataset using the I index: (a) is the original dataset while (b)–(f) are the clustering results of the E-DE, BF-K-means, ACDE, MANM, and TGCA algorithms.

cannot improve so much. Besides, although enumerating all possible cluster numbers, the general performance of the BF-K-means is lower than all evolutionary clustering algorithms. The reasons are: 1) the K-means algorithm only minimizes the intracluster distances, whereas the EC algorithms optimize the cluster validation index that considers both intracluster and intercluster distances and 2) the K-means is a local search algorithm that can find a local optimum only, whereas the EC algorithms try to find the global optimum of the problem.

For an evolutionary clustering algorithm, the dimension of the problem space is at least the number of features times the number of clusters. Commonly, in the field of EC, the problem possessing higher than 500 dimensions is considered as a high-dimensional problem. The Dim32 dataset contains 32 dimensions and 16 clusters, whose search space has at least $32 \times 16 = 512$ dimensions. It can be considered a high-dimensional problem in the EC field. From Table II, we can see that the E-DE algorithm performs well for the Dim32 dataset and surpasses the other six methods a lot. This is because the traditional evolutionary clustering algorithms endure the problems of encoding redundancy and/or cross-dimension learning error, whereas our elastic encoding and subspace crossover prevents the E-DE algorithm from these deficiencies. Nevertheless, for some extremely high-dimensional clustering situations like deep feature-based clustering, the dimension reduction or the cooperative evolution framework [62], [63] is required for the evolutionary clustering methods.

Further, we compare the convergence curves of the E-DE algorithm with those of the S-DE, ACDE, MANM, and TGCA algorithms. (We do not consider the K-means and BF-K-means algorithms because such methods terminate in a short time.) Taking the Vote and Dim32 datasets as examples, Fig. 5 compares the convergence curves of the DB index fitness values. It can be observed that the E-DE algorithm converges faster than S-DE, ACDE, MANM, and TGCA.

2) Validation by the I Index: Next, we use the I index to examine the seven clustering algorithms, for which the results are presented in Table III. It can be observed that our E-DE algorithm has an outstanding performance compared with other clustering algorithms for most of the datasets. E-DE tends to find the correct number of clusters so that it reaches an approximately 73% CSR on the 19 datasets. Meanwhile, the total ARI obtained by E-DE is much higher than the others. Taking the Vote and Dim32 datasets as examples to examine the convergence of the compared algorithms evaluated by the I index value, the curves are presented in Fig. 6. From Fig. 6(a), we can see that the E-DE converges much faster than the S-DE, ACDE, MANM, and TGCA. In Fig. 6(b), the E-DE algorithm eventually obtains better fitness than the other four clustering algorithms.

Further, the clustering results of the 2-D Unbalance are displayed in Fig. 7. Fig. 7(a) is the original dataset without adopting the clustering algorithm, while the remaining subfigures show the results of the E-DE, BF-K-means, ACDE, MANM, and TGCA algorithms in turn. The visual results also validate the powerfulness of our E-DE algorithm. Particularly, it can be seen from Fig. 7(a) that, for the Unbalance dataset, the groups of points possess different densities. Then, as shown in Fig. 7(b), E-DE correctly partitions the original dataset into eight clusters, which proves the effectiveness of our elastic encoding strategy and the corresponding evolution operators. In comparison, in Fig. 7(d), the ACDE cannot divide the five clusters in the right part thoroughly, and, in Fig. 7(c)–(f), the BF-K-means, MANM, and TGCA wrongly regard the five different clusters as one cluster.

3) Significance Tests: We conduct statistical tests based on the fitness evaluation indices to ensure that our improvement of performance is statistically significant. Particularly, the Wilcoxon's rank-sum test is performed for one-on-one comparisons, while the Kruskal–Wallis (KW) test is conducted for multiple comparisons. The significance test results using DB and I indices are reported in Tables IV and V, respectively. Meanwhile, in the tables, the symbol (+) indicates that the performance of E-DE is significantly better than the compared algorithm(s), whereas the symbol (−) indicates the opposite. Considering Wilcoxon's rank-sum test results, we can observe

TABLE IV
P-VALUES OF WILCOXON RANK-SUM TESTS AND KW TESTS BETWEEN OUR METHOD AND THE OTHER SIX CLUSTERING ALGORITHMS OF THE DB INDEX ON 19 DATASETS. (+) INDICATES THAT THE PERFORMANCE OF E-DE IS SIGNIFICANTLY BETTER THAN THE COMPARED ALGORITHM WITH A CONFIDENCE LEVEL OF 95%, WHILE (−) REPRESENTS THE OPPOSITE

Method		Iris	Cancer	Glass	Vowel	Zoo	Vote	Ecoli	Seed	Compound
Wilcoxon	E-DE to K-means	0.008(+)	0.010(+)	0.002(+)	0.002(-)	0.002(+)	0.037(+)	0.160	0.045(+)	0.010(+)
	E-DE to S-DE	0.021(+)	0.010(+)	0.002(+)	0.002(-)	0.002(+)	0.002(+)	0.027(+)	0.037(+)	0.019(+)
	E-DE to BF-K-means	0.038(+)	0.002(+)	0.042(+)	0.013(+)	0.002(+)	0.002(+)	0.026(+)	0.010(+)	0.037(+)
	E-DE to ACDE	0.002(+)	0.002(+)	0.002(+)	0.084	0.019(-)	0.027(+)	0.003(+)	0.007(+)	0.023(+)
	E-DE to MANM	0.049(+)	0.028(+)	0.008(+)	0.017(-)	0.002(+)	0.002(+)	0.206	0.007(-)	0.010(+)
	E-DE to TGCA	0.016(+)	0.002(+)	0.034(+)	0.063	0.002(+)	0.002(+)	0.008(+)	0.002(-)	0.016(+)
KW	E-DE to Others	0.010(+)	0.007(+)	0.008(+)	0.178	0.002(+)	0.027(+)	0.010(+)	0.010(+)	0.003(+)

Method		R15	Dim2	G2	Dim32	Customer	Unbalance	Spiral	Mockdata1	Mockdata2	Mockdata3
Wilcoxon	E-DE to K-means	0.010(+)	0.010(+)	0.002(+)	0.002(+)	0.084(-)	0.002(+)	0.084	0.042(+)	0.002(+)	0.026(+)
	E-DE to S-DE	0.007(+)	0.010(+)	0.014(+)	0.008(+)	0.028(-)	0.009(+)	0.032(+)	0.016(+)	0.002(+)	0.002(+)
	E-DE to BF-K-means	0.002(+)	0.005(+)	0.004(+)	0.063	0.002(+)	0.002(+)	0.002(+)	0.028(-)	0.010(+)	0.042(+)
	E-DE to ACDE	0.002(+)	0.002(+)	0.002(+)	0.014(+)	0.047(+)	0.004(+)	0.005(+)	0.024(+)	0.017(-)	0.002(+)
	E-DE to MANM	0.002(+)	0.002(+)	0.013(-)	0.002(+)	0.008(+)	0.044(+)	0.004(-)	0.010(+)	0.002(-)	0.010(+)
	E-DE to TGCA	0.010(+)	0.014(+)	0.002(-)	0.013(+)	0.019(+)	0.002(+)	0.002(+)	0.032(+)	0.037(+)	0.002(+)
KW	E-DE to Others	0.012(+)	0.002(+)	0.006(+)	0.013(+)	0.002(+)	0.368	0.008(+)	0.002(+)	0.004(+)	0.025(+)

TABLE V
P-VALUES OF THE WILCOXON RANK-SUM TESTS AND KW TESTS BETWEEN OUR METHOD AND THE OTHER SIX CLUSTERING ALGORITHMS OF THE I INDEX ON 19 DATASETS. (+) INDICATES THAT THE PERFORMANCE OF E-DE IS SIGNIFICANTLY BETTER THAN THE COMPARED ALGORITHM WITH A CONFIDENCE LEVEL OF 95%, WHILE (−) REPRESENTS THE OPPOSITE

Method		Iris	Cancer	Glass	Vowel	Zoo	Vote	Ecoli	Seed	Compound
Wilcoxon	E-DE to K-means	0.002(+)	0.002(+)	0.002(-)	0.005(-)	0.002(+)	0.002(+)	0.010(+)	0.010(+)	0.002(+)
	E-DE to S-DE	0.010(+)	0.010(+)	0.010(-)	0.010(-)	0.010(+)	0.009(+)	0.030(-)	0.010(+)	0.008(+)
	E-DE to BF-K-means	0.005(-)	0.019(+)	0.044(+)	0.026(+)	0.002(+)	0.002(+)	0.002(+)	0.008(-)	0.020(+)
	E-DE to ACDE	0.002(-)	0.018(+)	0.010(+)	0.012(-)	0.010(+)	0.002(+)	0.005(-)	0.067	0.008(-)
	E-DE to MANM	0.028(+)	0.023(+)	0.037(+)	0.010(+)	0.010(+)	0.010(+)	0.006(-)	0.006(+)	0.007(+)
	E-DE to TGCA	0.002(+)	0.002(+)	0.005(+)	0.002(+)	0.004(-)	0.004(+)	0.001(-)	0.002(+)	0.002(-)
KW	E-DE to Others	0.013	0.037(+)	0.036(+)	0.028	0.033(+)	0.030(+)	0.003(-)	0.004(+)	0.018(+)

Method		R15	Dim2	G2	Dim32	Customer	Unbalance	Spiral	Mockdata1	Mockdata2	Mockdata3
Wilcoxon	E-DE to K-means	0.010(+)	0.002(+)	0.002(+)	0.010(+)	0.010(+)	0.002(+)	0.010(+)	0.033(+)	0.002(+)	0.002(+)
	E-DE to S-DE	0.010(+)	0.002(+)	0.010(+)	0.010(+)	0.067	0.010(+)	0.010(+)	0.010(+)	0.065	0.003(+)
	E-DE to BF-K-means	0.007(+)	0.033(+)	0.037(-)	0.002(+)	0.010(+)	0.002(+)	0.003(+)	0.001(+)	0.003(+)	0.010(+)
	E-DE to ACDE	0.003(+)	0.004(+)	0.005(-)	0.009(+)	0.001(+)	0.003(+)	0.003(+)	0.019(+)	0.010(-)	0.010(+)
	E-DE to MANM	0.013(-)	0.006(+)	0.002(-)	0.002(+)	0.013(+)	0.002(+)	0.002(-)	0.007(+)	0.005(+)	0.010(+)
	E-DE to TGCA	0.002(+)	0.010(+)	0.002(-)	0.002(+)	0.010(+)	0.005(+)	0.008(+)	0.009(+)	0.002(+)	0.084(+)
KW	E-DE to Others	0.010(+)	0.044(+)	0.004	0.002(+)	0.030(+)	0.008(+)	0.001(+)	0.003(+)	0.034(+)	0.042(+)

from Table IV that, in terms of the DB index, the differences between the results of E-DE and the other algorithms are significant (the p-values are smaller than 0.05) on almost all datasets. Meanwhile, in most cases, E-DE outperforms the others. Similar results are also observed from Table V, showing significantly better performance of the proposed method in the comparisons using the I index. In addition, the above KW test results prove that, on most datasets, our E-DE algorithm is able to achieve statistically better results than the other six algorithms.

C. Analysis of Time Complexity

For the evolutionary clustering algorithms, the computational costs involve two parts: 1) the fitness evaluation of the clustering index and 2) the evolution operation. The complexity of fitness evaluation is identical for different evolutionary clustering algorithms. Particularly, the data assignment consumes $O(N \times K)$ time in calculating the distance between data points and cluster centroids, and the fitness evaluation of DB and I indices both takes an additional $O(K^2)$ time to calculate the distance between cluster centroids. Therefore, the fitness evaluation step consumes $O((N + K) \times K \times np)$ time in each iteration. The E-DE algorithm does not bring additional distance calculations when adopting the new mutation and crossover operators. The complexity of the evolution operation is the same as the classical DE algorithm, which is $O(K \times np)$ in each iteration. The evolution operations of ACDE and TGCA have the equal time consumption with our E-DE. Besides, MANM has an additional $O((K + np) \times np)$ time cost, which is caused by the auxiliary merging and niching procedures. From the time complexity analysis, we can see that the evolution costs of the evolutionary clustering algorithms are far less than the evaluation costs, which can be ignored, especially when the scale of datasets is large. Besides, as indicated in the experimental comparisons, the proposed E-DE converges faster than the other algorithms, which consumes fewer fitness evaluation times to obtain promising results. From this point of view, E-DE is more computationally efficient than the others.

V. CONCLUSION

In this article, we proposed an E-DE to tackle the automatic data clustering problem. First, an elastic individual encoding scheme was introduced to adapt the number of clusters without external guidance. Unlike the zero-one switch encoding scheme used in traditional algorithms, this elastic scheme enables the population of DE to contain individuals that have different dimensions and evolves this parameter (i.e., the number of dimensions) inherently during the evolution. Second, a two-phase mutation operator was designed, which subtly utilizes the information of three individuals with random selection, deletion, and permutation operations to generate mutant vectors. Through this mutation, the individual learns sufficient information about the data clustering structure from the current population in order to adapt the cluster numbers and centroids. Third, we developed an exponential subspace crossover strategy that exchanges partial cluster layouts between parameter vectors. This exchange method not only avoids the cross-dimension learning error encountered by the previous algorithms, but it also adjusts the number of clusters in a flexible way. The experimental results proved that our E-DE algorithm outperformed the other clustering algorithms in terms of both internal and external clustering indices.

The E-DE algorithm showed significant improvements over the existing methods, but some limitations still remain. The proposed algorithm belongs to centroid-based (partitioning) clustering. Although such methods have been widely applied in practice, they endure the performance degradation in handling the nonconvex datasets. So one of our future works is to extend the E-DE framework by incorporating some density or connectivity-based clustering methods for nonconvex datasets. In addition, the evolutionary clustering methods may encounter deficiency when handling the extremely high-dimensional datasets like the deep feature-based image database. Another possible extension is to make some adjustments, such as incorporating a cooperative coevolution framework, to further enhance the scalability of the E-DE algorithm. Besides, due to the good performance of E-DE, it would be appealing to apply the algorithm to some practical applications, such as image analysis, document clustering, and gene detection. To summarize, we believe that our E-DE clustering method is flexible and robust to solve the automatic data clustering problem and there remains lots of possibilities for it to extend to other subjects or reality fields.

References

[1] Q. Zhang, C. Zhu, L. T. Yang, Z. Chen, L. Zhao, and P. Li, "An incremental CFS algorithm for clustering large data in Industrial Internet of Things," *IEEE Trans. Ind. Inf.*, vol. 13, no. 3, pp. 1193–1201, Jun. 2017.

[2] W. Bi, M. Cai, M. Liu, and G. Li, "A big data clustering algorithm for mitigating the risk of customer churn," *IEEE Trans. Ind. Inf.*, vol. 12, no. 3, pp. 1270–1281, Jun. 2016.

[3] P. A. Jaskowiak, R. J. G. B. Campello, and I. G. Costa, "Proximity measures for clustering gene expression microarray data: A validation methodology and a comparative analysis," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 10, no. 4, pp. 845–857, Jul./Aug. 2013.

[4] P. Maji and S. Paul, "Rough-fuzzy clustering for grouping functionally similar genes from microarray data," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 10, no. 2, pp. 286–299, Mar./Apr. 2013.

[5] P. C. H. Ma, K. C. C. Chan, X. Yao, and D. K. Y. Chiu, "An evolutionary clustering algorithm for gene expression microarray data analysis," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 296–314, Jun. 2006.

[6] R. C. Romero-Zaliz, C. Rubio-Escudero, J. P. Cobb, F. Herrera, Ó. Cordón, and I. Zwir, "A multiobjective evolutionary conceptual clustering methodology for gene annotation within structural databases: A case of study on the gene ontology database," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 679–701, Dec. 2008.

[7] I. D. Gebru, X. Alameda-Pineda, F. Forbes, and R. Horaud, "EM algorithms for weighted-data clustering with application to audio-visual scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 12, pp. 2402–2415, Dec. 2016.

[8] N. Lu and H. Miao, "Clustering tree-structured data on manifold," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1956–1968, Oct. 2016.

[9] G. Folino, C. Pizzuti, and G. Spezzano, "Training distributed GP ensemble with a selective algorithm based on clustering and pruning for pattern classification," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 458–468, Aug. 2008.

[10] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A *k*-means clustering algorithm," *J. Roy. Stat. Soc. C (Appl. Stat.)*, vol. 28, no. 1, pp. 100–108, 1979.

[11] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[12] J. Qin, W. Fu, H. Gao, and W. X. Zheng, "Distributed *k*-means algorithm and fuzzy *c*-means algorithm for sensor networks based on multiagent consensus theory," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 772–783, Mar. 2017.

[13] D. S. Hochbaum, "Approximation algorithms for NP-hard problems," *ACM SIGACT News*, vol. 28, no. 2, pp. 40–52, 1997.

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

[15] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown, "FGKA: A fast genetic *K*-means clustering algorithm," in *Proc. ACM Symp. Appl. Comput.*, 2004, pp. 622–623.

[16] P. Merz and A. Zell, "Clustering gene expression profiles with memetic algorithms," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2002, pp. 811–820.

[17] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 218–237, Jan. 2008.

[18] C. Liu, A. Zhou, and G. Zhang, "Automatic clustering method based on evolutionary optimisation," *IET Comput. Vis.*, vol. 7, no. 4, pp. 258–271, Aug. 2013.

[19] E. R. Hruschka and N. F. F. Ebecken, "A genetic algorithm for cluster analysis," *Intell. Data Anal.*, vol. 7, no. 1, pp. 15–25, 2003.

[20] E. R. Hruschka, R. J. G. B. Campello, and L. N. De Castro, "Evolving clusters in gene-expression data," *Inf. Sci.*, vol. 176, no. 13, pp. 1898–1927, 2006.

[21] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm Evol. Comput.*, vol. 16, pp. 1–18, Jun. 2014.

[22] A. Jos-Garca and W. Gmez-Flores, "Automatic clustering using nature-inspired metaheuristics: A survey," *Appl. Soft Comput.*, vol. 41, pp. 192–213, Apr. 2016.

[23] E. Hancer and D. Karaboga, "A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number," *Swarm Evol. Comput.*, vol. 32, pp. 49–67, Feb. 2017.

[24] W.-L. Xiang, N. Zhu, S.-F. Ma, X.-L. Meng, and M.-Q. An, "A dynamic shuffled differential evolution algorithm for data clustering," *Neurocomputing*, vol. 158, pp. 144–154, Jun. 2015.

[25] R. Tinós, L. Zhao, F. Chicano, and D. Whitley, "NK hybrid genetic algorithm for clustering," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 748–761, Oct. 2018.

[26] C. Guan, K. K. F. Yuen, and F. Coenen, "Particle swarm optimized density-based clustering and classification: Supervised and unsupervised learning approaches," *Swarm Evol. Comput.*, vol. 44, pp. 876–896, Feb. 2019.

[27] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay, "Multiobjective genetic algorithm-based fuzzy clustering of categorical attributes," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 991–1005, Oct. 2009.

[28] I. Saha, D. Maity, and U. Maulik, "Categorical data analysis using multiobjective differential evolution based fuzzy clustering," in *Proc. IEEE Int. Conf. Adv. Comput. Commun. Informat. (ICACCI)*, 2013, pp. 2013–2017.

[29] A. Gupta, S. Datta, and S. Das, "Fast automatic estimation of the number of clusters from the minimum inter-center distance for *k*-means clustering," *Pattern Recognit. Lett.*, vol. 116, pp. 72–79, Dec. 2018.

[30] H.-H. Tam, S.-C. Ng, A. K. Lui, and M.-F. Leung, "Improved activation schema on automatic clustering using differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2017, pp. 1749–1756.

[31] J. Arellano-Verdejo, E. Alba, and S. Godoy-Calderon, "Efficiently finding the optimum number of clusters in a dataset with a new hybrid differential evolution algorithm: DELA," *Soft Comput.*, vol. 20, no. 3, pp. 895–905, 2016.

[32] J. Tvrdik and I. Křivỳ, "Hybrid differential evolution algorithm for optimal clustering," *Appl. Soft Comput.*, vol. 35, pp. 502–512, Oct. 2015.

[33] U. Maulik and I. Saha, "Automatic fuzzy clustering using modified differential evolution for image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 9, pp. 3503–3510, Sep. 2010.

[34] D. Chang and X. Zhang, "Dynamic niching genetic algorithm with data attraction for automatic clustering," *Tsinghua Sci. Technol.*, vol. 14, no. 6, pp. 718–724, Dec. 2009.

[35] W. Sheng, S. Chen, M. Fairhurst, G. Xiao, and J. Mao, "Multilocal search and adaptive niching based memetic algorithm with a consensus criterion for data clustering," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 721–741, Oct. 2014.

[36] W. Sheng, S. Chen, M. Sheng, G. Xiao, J. Mao, and Y. Zheng, "Adaptive multisubpopulation competition and multiniche crowding-based memetic algorithm for automatic data clustering," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 838–858, Dec. 2016.

[37] H. He and Y. Tan, "A two-stage genetic algorithm for automatic clustering," *Neurocomputing*, vol. 81, pp. 49–59, Apr. 2012.

[38] S. Chakraborty and S. Das, "Simultaneous variable weighting and determining the number of clusters—A weighted Gaussian means algorithm," *Stat. Probab. Lett.*, vol. 137, pp. 148–156, Jun. 2018.

[39] S. Saha, A. Ekbal, and A. K. Alok, "Semi-supervised clustering using multiobjective optimization," in *Proc. IEEE 12th Int. Conf. Hybrid Intell. Syst. (HIS)*, 2012, pp. 360–365.
[40] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay, "Hybrid evolutionary multiobjective fuzzy c-medoids clustering of categorical data," in *Proc. IEEE Workshop Hybrid Intell. Models Appl. (HIMA)*, 2013, pp. 7–12.
[41] Y. Wan, Y. Zhong, and A. Ma, "Fully automatic spectral–spatial fuzzy clustering using an adaptive multiobjective memetic algorithm for multispectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2324–2340, Apr. 2019.
[42] M. Garza-Fabre, J. Handl, and J. Knowles, "An improved and more scalable evolutionary approach to multiobjective clustering," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 515–535, Aug. 2018.
[43] R. Xu, J. Xu, and D. C. Wunsch, "A comparison study of validity indices on swarm-intelligence-based clustering," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 4, pp. 1243–1256, Aug. 2012.
[44] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognit.*, vol. 46, no. 1, pp. 243–256, 2013.
[45] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.
[46] S. Bandyopadhyay and U. Maulik, "Nonparametric genetic clustering: Comparison of validity indices," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 1, pp. 120–125, Feb. 2001.
[47] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985.
[48] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, and P. N. Suganthan, "Ensemble of differential evolution variants," *Inf. Sci.*, vol. 423, pp. 172–186, Jan. 2018.
[49] X. Qiu, J.-X. Xu, Y. Xu, and K. C. Tan, "A new differential evolution algorithm for minimax optimization in robust design," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1355–1368, May 2018.
[50] P. Ochoa, O. Castillo, J. Soria, and P. Cortes-Antonio, "Differential evolution algorithm using a dynamic crossover parameter with high-speed interval type 2 fuzzy system," in *Advances in Soft Computing*. Cham, Switzerland: Springer Int., 2018, pp. 369–378.
[51] O. Castillo, F. Valdez, J. Soria, L. Amador-Angulo, P. Ochoa, and C. Peraza, "Comparative study in fuzzy controller optimization using bee colony, differential evolution, and harmony search algorithms," *Algorithms*, vol. 12, no. 1, p. 9, 2018.
[52] (Aug. 31, 2019). *E-DE Source Code*. [Online]. Available: https://github.com/rich-lavender/E-DE_v1
[53] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[54] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[55] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. C-100, no. 1, pp. 68–86, Jan. 1971.
[56] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1273–1280, Sep. 2002.
[57] I. Kärkkäinen and P. Fränti, "Gradual model generator for single-pass clustering," *Pattern Recognit.*, vol. 40, no. 3, pp. 784–795, 2007.
[58] P. Fränti, R. Mariescu-Istodor, and C. Zhong, "XNN graph," in *Proc. Joint IAPR Int. Workshops Stat. Techn. Pattern Recognit. Struct. Syn. Pattern Recognit.*, 2016, pp. 207–217.
[59] P. Franti, O. Virmajoki, and V. Hautamaki, "Fast agglomerative clustering using a *k*-nearest neighbor graph," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1875–1881, Nov. 2006.
[60] M. Rezaei and P. Fränti, "Set matching measures for external cluster validity," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2173–2186, Aug. 2016.
[61] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, 2008.
[62] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.
[63] X. Y. Zhang, Y. J. Gong, Y. Lin, J. Zhang, S. Kwong, and J. Zhang, "Dynamic cooperative coevolution for large scale optimization," *IEEE Trans. Evol. Comput.*, to be published.

Jun-Xian Chen received the bachelor's degree from Jinan University, Guangzhou, China, in 2017. He is currently pursuing the master's degree with the School of Computer Science and Engineering, South China University of Technology, Guangzhou.

His current research interests include evolutionary computation, swarm intelligence, and data mining.

Yue-Jiao Gong (S'10–M'15–SM'19) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

She is currently a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her current research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation and smart city scheduling. She has published over 80 papers, including over 30 IEEE TRANSACTIONS papers in the above areas.

Wei-Neng Chen (S'07–M'12–SM'17) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering.

Prof. Chen was a recipient of the IEEE Computational Intelligence Society Outstanding Dissertation Award in 2016 for his Ph.D. thesis and the National Science Fund for Excellent Young Scholars in 2016.

Mengting Li received the M.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China.

She is currently a Postdoctoral Research Fellow with the School of Data and Computer Science, Sun Yat-sen University. Her current research interests include data mining and machine learning.

Jun Zhang (M'02–SM'08–F'17) received the Ph.D. in electrical engineering degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high-performance computing, and operations research.

Dr. Zhang is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.