2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Evolutionary Clustering with Differential Evolution

Gang Chen[1, 2] , Wenjian Luo[1, 2, *] and Tao Zhu[1, 2]
[1]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, Anhui, China
[2]Anhui Province key Laboratory of Software Engineering in Computing and Communication, University of Science and Technology of China, Hefei 230027, Anhui, China
jeken@mail.ustc.edu.cn, wjluo@ustc.edu.cn, zhuta@mail.ustc.edu.cn

***Abstract*—Evolutionary clustering is a hot research topic that clusters the time-stamped data and it is essential to some important applications such as data streams clustering and social network analysis. An evolutionary clustering should accurately reflect the current data at any time step while simultaneously not deviate too drastically from the recent past. In this paper, the differential evolution (DE) is applied to deal with the evolutionary clustering problem. Comparing with the typical k-means, evolutionary clustering based on DE (deEC) could perform a global search in the solution space. Experimental results over synthetic and real-world data sets demonstrate that the deEC provides robust and adaptive solutions.**

I. Introduction

Clustering is a task which aims to partition an unlabeled data set into groups according to the similarities among its objects [1], [2]. In the past decades, clustering has played an important role in many applications ranging from text mining, social web analysis, information discovery, bio-informatics and image segmentation. The traditional clustering algorithms, such as k-means [3] , are based on the assumption that all the samples are extracted from the same probability distribution and they keep static as the time goes by [4]. However, in typical large data mining problems, such as social web analysis, data are time-evolving [5]–[7] . In these cases, data not only change over a series of time steps, but their structures are various at different time steps. Usually, we expect the results obtained by the clustering algorithm should reflect the corresponding change of data in time. Moreover, the result should also not deviate too much from the recent past. This expectation is reasonable because a user is more willing to find a familiar clustering result at each time step rather than learn an entirely new way to segment data. Unfortunately, the traditional clustering algorithms in the literature fail to address these problems since they are inadequate to deal with time-evolving data.

One of the feasible clustering paradigms is evolutionary clustering which processes time-evolving data to produce a sequence of clusterings [8]. Here we should distinguish the concept of $evolutionary\ clustering$ from the evolutionary algorithms for clustering. The "evolutionary" means the evolutionary process of clusters in time-evolving data. In evolutionary clustering, a clustering result should achieve the two potentially conflicting criteria simultaneously: first, the clustering should accurately reflect the current data as much as possible; second, the clustering should not deviate drastically from one time step to the next. For example, in clustering of MicroBlog users for recommending topics, it is natural to expect that, the particular kinds of topics a user may be probably interested in change over time. However, it is important that the clustering result should not drift too much from one time step to the next. If the true relationships among data points change drastically, evolutionary clustering does not re-cluster the accumulated history data points, but integrate the history information to keep a smooth transition from the recent past. These two criteria also hold true to cluster large-scale evolving data because in many cases, it is time-consuming even computationally infeasible to re-cluster as the data evolves. Evolutionary clustering is essential to potential and existing applications like social network analysis to observe the change of community, blogs analysis to identify the new and hot topics.

In this paper, the differential evolution (DE) is applied to deal with the evolutionary clustering problem. Evolutionary algorithms (EAs) are wildly used to deal with the nonlinear optimization problems. Comparing to the classical techniques (gradient descent, deterministic hill climbing et al.), EAs are robust and adaptive search techniques which perform a global search in the solution space. In evolutionary clustering based on DE, two objective functions, the $snapshot\ cost$ and the $history\ cost$, are defined corresponding to the two criteria, respectively. Formally, let $\boldsymbol{X}_{t,i}$ be an individual at time step t. The $snapshot\ cost$ is used as the objective function to measure the quality of clustering of $\boldsymbol{X}_{t,i}$ at time step t. A higher $snapshot\ cost$ means the worse snapshot quality. The $history\ cost$ of the clustering is a measure of the $temporal\ smoothness$ of $\boldsymbol{X}_{t,i}$ in the old environment. A higher $history\ cost$ means the worse $temporal\ smoothness$ in the transition from time step $t-1$ to t. These two objective functions are combined into one with different weights to guide the population evolving toward the global optima. The weights is adaptive as the algorithm progresses. In addition, we consider evolutionary clustering as a particular kind of dynamic optimization problems [9]–[12]. The situation where there are new data arriving or old data disappearing is treated as a change of the environment. The algorithm is capable to respond to this change and find the global near-optimal at any time step.

Up to now, the work that clusters time-evolving data sets based on an evolutionary algorithm is little. To our best knowledge, there is only one paper that is related to evolutionary clustering and evolutionary algorithms. That is, Ma et al. [13] use a multi-objective evolutionary algorithm to optimize the two conflicting functions in evolutionary k-means algorithm (EKM). However, in this paper, we still consider the evolutionary clustering as an single-objective optimization

*Corresponding author.
$Tel.: +86-551-63602824$

problem. We believe that there are at least the following advantage to utilize an evolutionary algorithm. On one hand, a DE is easy to implement and requires a few amount of parameters tuning to achieve a promising search result. On the other hand, based on DE, the algorithm could perform a global search in the solution space and provides robust and adaptive solutions.

The rest of this paper is organized as follow. We review the related works in section II. An evolutionary clustering algorithm based on DE is presented in details in section III. Extensive experimental results on both synthetic and real data sets are given in section IV. Finally, we conclude the whole paper in section V.

II. Related Work

In this section, we provide some significant background on evolutionary clustering, evolutionary algorithms (EAs) for clustering and review the related work.

A. Evolutionary Clustering

Evolutionary clustering is a hot topic and is first formulated by Chakrabarti and his colleagues [8]. In their work, a general framework is presented and two instantiations of the framework are described: an evolutionary version of the traditional k-means and an evolutionary version of the bottom-up agglomerative hierarchical clustering algorithm. In order to ensure the temporal smoothness, the framework adds a history cost to the original cost in the two clustering algorithms. The history cost penalizes the deviation of current cluster solution from the previous. In general, the evolutionary clustering will present the user a sequence of clusterings. Each clustering in the sequence is consistent with the clustering in the recent past. And if true clusters shift drastically over time, the algorithm will present a smooth view of the transition [8]. In addition, their work mainly focused on solving the online problem because they consider the online setting is arguably more important for real-world applications. Here, it is often said that an evolutionary clustering algorithm is online if it can only utilize the data during and before time step t. If the algorithm has access to all data beforehand, then it is offline.

Several research topics in the literature are close to evolutionary clustering, such as online document clustering, incremental clustering, clustering data streams and so on. However, they are essentially different. The task of online document clustering is to partition the documents as long as they arrive in a temporal sequence [14]. The temporal properties are also considered in online document clustering, however, which is to capture novelty in the form of some types of outliers, not to incorporate history cost into clustering objective. Typically, the issue of incremental clustering [15] is to update the cluster centers [16] or hierarchical trees [17] as new data arrive. Contrasted to evolutionary clustering, incremental clustering does not concern about the relationships between the current data and historical data, and the updating process is primarily to avoid storing all historical similarities. Clustering data stream [18] addresses the problem to cluster data that are generated continuously and frequently in a very wide range of fields. The early works have focused on analytically clustering data streams using K-median technique. The proposed algorithm makes a single pass over the data stream and uses small space. The recently works, however, turn to use a pyramidal time frame [5]. Clustering data streams ignores the smoothness of the clustering results in the series of time steps while it is an important objective in evolutionary clustering.

Although there is not enough literature about evolutionary clustering up to now, two categories could be roughly classified. Some researchers developed statistical models to describe the evolutionary data based on the assumption that the data are subject to stochastic process. These methods are the first category. In the second category, a smoothness property is incorporated into the clustering algorithms and their goal is to improve the quality of the result at each time step and the temporal smoothness between time steps. Some recent works on evolutionary clustering are presented as follows. Xu and Zhang et al. [19] built Hidden Markov Model combined with Hierarchical Dirichlet Process to describe the correlation of clustering results among the different time steps. Their method is non-parametric but the model is too complicated. Later, the same authors simplified their method. They directly utilized Dirichlet Process to build an infinite mixture model to describe the evolutionary change of the clusters over the time [20]. In their work, the cluster numbers could be automatically learnt during the evolution. The methods above are the first category, and the methods listed below are belong to the second one. Chi et al. [21] extended Chakrabarti's work and proposed two algorithms that incorporated temporal smoothness into the overall measure of clustering quality. Ma et al. [13] proposed to use a multi-objective evolutionary algorithm to optimize the two conflicting functions in evolutionary k-means algorithm (EKM). Wang et al. [22] proposed a general framework, which is based on low-rank kernel matrix factorization, to deal with evolutionary clustering problem.

B. Evolutionary Algorithms for Clustering

There are a number of literature on evolutionary algorithms for clustering [23]–[26]. However, the same as the traditional clustering algorithms, up to now the evolutionary algorithms for clustering only consider that the data are static. Noted that the techniques used in these algorithms are relevant to the proposed method, we would like to describe them concisely. To deal with the clustering problems, evolutionary algorithms basically evolve the initial candidates sampled from the search space through a sort of operators that use probabilistic rules [27]. Usually, the evolutionary algorithms are based on the optimizations of some objective functions. The solutions that more fitted the objective functions have higher probabilities to be sampled. The algorithms iterate the evolving processes and as a result, the more promising clustering solutions are survived. Generally speaking, the evolutionary algorithms could provide solutions of better quality than those found by traditional algorithms.

In evolutionary algorithms for clustering, the cluster number k is usually fixed in advance by the user. However, in some cases, it could be determined automatically by the clustering algorithm. If the cluster number k is fixed before running, we call the algorithm is parametric method; otherwise it is non-parametric method. Krishna and Murty [23] used a genetic algorithm to address the clustering problem. In their work, the cluster number k is set before running. While in Das's work

[24], an improved DE is proposed to automatically cluster the large-scale unlabeled data set. The proposed algorithm tries to determine the optimal number of partitions of the data during the running of the algorithm. The non-parametric method is more important to the real-world applications because in many cases, we do not have the prior knowledge on the data sets. In some applications such as image processing, evolutionary algorithms for clustering are mainly used to divide images into disjoint homogeneous regions which usually contain similar objects of particular interest. And the algorithms are required to automatically determine the cluster numbers. Bandyopadhyay et al. applied evolutionary algorithms for clustering [25], [26] to distinguish landscape regions like mountains, rivers, vegetation areas and habitations in satellite images.

III. Evolutionary Clustering Based on DE

In this section, an evolutionary clustering based on DE (we called deEC in the rest of the paper for convenience) is presented and the implementation details will be discussed as well.

A. Chromosome Representation

In the deEC, a chromosome is represented as a vector of real numbers of $K*d$ dimensions. K is a user specified cluster number and d is the dimension of the data points. The first d positions represent the d dimensions of the first cluster center, the next d positions represent those of the second cluster center, and so forth. For example, a probable chromosome in the two dimension space can be represented as the following vector.

$$\boldsymbol{X} = \boxed{1.5 \mid 2.0 \mid 5.5 \mid 5.0 \mid 2.5 \mid 3.0}$$

It indicates that there are 3 cluster centers encoded in the chromosome, (1.5 , 2.0), (5.5 , 5.0) and (2.5 , 3.0), respectively. Given a chromosome like this, a cluster solution can be recovered according to the nearest prototype rule. That is, assigning each data object to its nearest cluster. It is noted that such an chromosome representation has been wildly used in evolutionary algorithms for static clustering [24], [26].

B. Fitness Function

Chakrabarti et al. [8] first addressed the evolutionary clustering problem and proposed a general framework. The framework focused on the online setting and tried to optimize the incremental quality of cluster at each time step t:

$$sq(C_t, M_t) - cp * hc(C_t, C_{t-1}) \tag{1}$$

where the first term is the $snapshot\ cost$ which measures the quality of the clustering at time step t. C_t is the partitioning for the data at time step t and M_t are the input matrix at time step t. The second term is the $history\ cost$ which measures the distance between C_{t-1} and C_t.

Many clustering validity indexes can be used in deEC to assess the snapshot quality. For the parametric setting, the cluster number K is set beforehand at each time step, we can use the validity index J [28] outlined as:

$$J = \sum_{j=1}^{K} \sum_{\boldsymbol{s}_{t,i} \in \boldsymbol{c}_{t,j}} ||\boldsymbol{s}_{t,i} - \boldsymbol{z}_{t,j}|| \tag{2}$$

where $\boldsymbol{C}_t = \{\boldsymbol{c}_{t,1}, \boldsymbol{c}_{t,2}, \boldsymbol{c}_{t,3}, ..., \boldsymbol{c}_{t,K}\}$ is the set of K clusters encoded into a genotype (also called chromosome), $\boldsymbol{s}_{t,i}$ is a data object, and $\boldsymbol{z}_{t,j}$ is the mean vector of cluster $\boldsymbol{c}_{t,j}$. There are many other clustering validity indexes. For crisp clustering, the well-known indexes like the Dunn's index (DI) [29], the DB index [30], can also be used in our work. Usually, different clustering validity indexes lead to different cluster quality. For example, they could lead to different cluster numbers and the corresponding best partition. We do not discuss this issue in this paper.

The $temporal\ smoothness$ is expressed as the adaptability of an individual (genotype or chromosome) in the old environment. Suppose there are two individuals $\boldsymbol{X}_{t,i}$ and $\boldsymbol{X}_{t,j}$, performing equally well in the environment at time step t (having equal snapshot quality). However, $\boldsymbol{X}_{t,i}$ performs better than $\boldsymbol{X}_{t,j}$ (the fitness value of $\boldsymbol{X}_{t,i}$ is better) when they are put into old environment at time step $t-1$. Then we consider $\boldsymbol{X}_{t,i}$ is better than $\boldsymbol{X}_{t,j}$ because cluster solution carried by $\boldsymbol{X}_{t,i}$ is more consistent with historical data. In the deEC, each of the individuals in the population at time step t is put into the old environment at time step $t-1$ and the so-called history cost is obtained. The higher history cost (fitness value) of a individual has worse temporal smoothness. The history cost is given as follows by using the validity index in equation (2):

$$HC = \sum_{j=1}^{K} \sum_{\boldsymbol{s}_{t-1,i} \in \boldsymbol{c}_{t,j}} ||\boldsymbol{s}_{t-1,i} - \boldsymbol{z}_{t,j}|| \tag{3}$$

where $\boldsymbol{C}_t = \{\boldsymbol{c}_{t,1}, \boldsymbol{c}_{t,2}, \boldsymbol{c}_{t,3}, ..., \boldsymbol{c}_{t,K}\}$ is the set of K clusters encoded into a genotype (also called chromosome), $\boldsymbol{s}_{t,i}$ is a data object, and $\boldsymbol{z}_{t,j}$ is the mean vector of cluster $\boldsymbol{c}_{t,j}$.

In deEC, the objective function (also called fitness function) is defined as the overall cost of the clustering at each time step, which consists of the snapshot cost and the history cost. The overall cost by using the equations (1)-(3) can be outlined as:

$$\begin{aligned} F(t) = \alpha * \sum_{j=1}^{k} \sum_{\boldsymbol{s}_{t,i} \in \boldsymbol{c}_{t,j}} \parallel \boldsymbol{s}_{t,i} - \boldsymbol{z}_{t,j} \parallel + \\ \beta * \sum_{j=1}^{k} \sum_{\boldsymbol{s}_{t-1,i} \in \boldsymbol{c}_{t,j}} \parallel \boldsymbol{s}_{t-1,i} - \boldsymbol{z}_{t,j} \parallel \end{aligned} \tag{4}$$

where the first term is snapshot cost and the second term is history cost. The parameters α and β are the weights and $\beta = 1 - \alpha$. In this paper, the weight α is adaptive as the algorithm progresses. At the beginning, the value of α is small (the value is 0.3) and therefore more the weight is assigned to the temporal smoothness. Then the value of α is linearly increased to the maximum value (the maximum value is set 0.9). The adaptive weight α can be expressed in the following equation:

$$\alpha = 0.3 + \frac{0.6 * g}{G} \tag{5}$$

where G is the maximum number of generations and g is the current generation number. By this way, the algorithm at the beginning focuses on the search in the solution space near the historical solutions but gradually adjusts the movements of the solutions during the later stages of search, so that they can reflect the current data as well as possible.

Alternatively, if the algorithm focuses on the snapshot quality at the beginning but gradually increases the weight of the history cost to ensure the temporal smoothness, we can use the equation outlined as:

$$\alpha = 1 - \frac{0.1 * g}{G} \tag{6}$$

where the G and g are the same as before.

Mathematically, the expression of total cost in equation (4) is essentially equivalent to the expression of PCQ in [21]. The history cost is produced by using the cluster solution at time step t to cluster data points at time step $t-1$, the same as PCQ. However, we still use the mean vectors Z_t encoded in the individual as the mean vectors at time step $t-1$. Because it seems more conform to the definition of the history cost and we believe it embodies the concept that an optimal individual is the result of combined influence between the current environment and the past ones. Additionally, we used the adaptive parameters. The algorithm with different definitions of α outlined in equations (5) and (6) is named deEC1 and deEC2, respectively.

C. Algorithm Description

Algorithm 1 deEC

Input:
data set $\boldsymbol{S}_{t-1}$, $\boldsymbol{S}_t$, cluster number $k^{(t-1)}$, $k^{(t)}$
the size of the population NP
the maximum number of generation G

Output:
the best individual $\boldsymbol{X}_{t,best}$

Randomly initialize a population of NP individuals, each individual contains $k^{(t)}$ number of randomly selected cluster centers;
for $g = 1$ *to* G **do**
  for each individual $\boldsymbol{X}_{t,i}$ **do**
    calculate the distance of each data vector $\boldsymbol{s}_{t,i}$ from all cluster centers of $\boldsymbol{X}_{t,i}$ by the $Euclidean\ distance$:

$$d(\boldsymbol{s}_{t,i}, \boldsymbol{z}_{t,i}) = \sqrt{\sum_{p}^{d} (s_{t,i,p} - z_{t,i,p})^2}$$

    assign each data vector $\boldsymbol{s}_{t,i}$ to its nearest cluster center
    process a one step k-means algorithm as a local search operation;
    process the DE operations according to the DE algorithm outlined in equations (8)-(12). Use the fitness function outlined in equations (4)-(6) to guide the evolution.
  end for
end for
report the best individual $\boldsymbol{X}_{t,best}$ and the cluster centers as well as the partition obtained by $\boldsymbol{X}_{t,best}$.

1) Classical DE: The classical DE [31] is an effective meta-heuristic, global optimization algorithm which uses real-coded representation. After the initialization, it works by iterating several operators including mutation, crossover and selection. In the initialization process, an initial population is randomly generated. The ith individual vector (genotype or chromosome) of the population at time step t and generation g has $K \cdot d$ components (dimensions),i.e.,

$$\boldsymbol{X}_{t,i}(g) = [x_{t,i,1}(g), x_{t,i,2}(g), x_{t,i,3}(g), ..., x_{t,i,Kd}(g)] \tag{7}$$

In the course of mutation, DE creates a donor vector $\boldsymbol{V}_{t,i}(g)$ corresponding to each target vector $\boldsymbol{X}_{t,i}(g)$. The classical strategy is to randomly samples three other individuals, i.e., $\boldsymbol{X}_{t,k}(g), \boldsymbol{X}_{t,m}(g), \boldsymbol{X}_{t,n}(g)$, from the same generation (for distinct k, m, n, and i). The difference of any two of these three vectors is scaled by a scale factor F (typically lies in [0.4,1]). Then the donor vector $\boldsymbol{V}_{t,i}(g)$ is obtained by adding the third sample to the scaled difference:

$$\boldsymbol{V}_{t,i}(g) = \boldsymbol{X}_{t,k}(g) + F(\boldsymbol{X}_{t,m}(g) - \boldsymbol{X}_{t,n}(g)) \tag{8}$$

The trial vector $\boldsymbol{U}_{t,i}(g)$ is created by crossing the components of target vector and donor vector whenever a randomly generated number (in [0,1]) is less than or equal to the Cr value:

$$U_{t,i,n}(g+1) = \begin{cases} V_{t,i,n}(g), & \text{if rand}(0,1) < Cr \\ X_{t,i,n}(g). & otherwise \end{cases} \tag{9}$$

The selection operator determines whether the trial vector or the target vector survives to the next generation. If the trial vector yields a better value of the objective function, it replaces the target vector in the next generation; otherwise the target vector is retained in the population. That is to say,

$$\boldsymbol{X}_i(g+1) = \begin{cases} \boldsymbol{U}_i(g+1), & \text{if F}(\boldsymbol{U}_i(g+1)) < \text{F}(\boldsymbol{X}_i(g)) \\ \boldsymbol{X}_i(g). & otherwise \end{cases} \tag{10}$$

2) Parameters F and Cr: To improve the convergence rate of DE, we adopt the methods proposed in [24] to modify the parameters F and Cr. We vary the scale factor F in a random manner ranged 0.5 to 1 outlined as:

$$F = 0.5 * (1 + \text{rand}(0,1)) \tag{11}$$

where $rand(0,1)$ is a function generating uniformly distributed number within the range [0,1]. The mean value of F is 0.75 and it is believed that this helps the algorithm to retain the population diversity as the search progresses.

In addition, we decrease the crossover rate Cr linearly [24]:

$$Cr = \frac{0.5 * (\text{G-g})}{\text{G}} \tag{12}$$

where G is the maximum number of generations and g is the current generation number. By this way, it is believed that it can help thoroughly explore the search space at the beginning but convergent into a small space where the global optimum lies during the latter stages of search.

3) Local Search by K-means: Many hybrid or memetic evolutionary algorithms for clustering are usually endowed with mechanisms to globally explore and locally exploit the search space. In general, a local exploitation of the search space can sometimes avoid expensive computation and help the algorithm quickly converge to the global optimum [23]. In deEC, the global search may take a long time to iterate the DE operators described in equations (8)-(10). Hence a one step k-means algorithm, named as the k-means operator, is used in deEC. Let $\boldsymbol{X}_{t,i}$ be a genotype. As the work in [23], we use the following two steps:

1) recalculate each cluster center by the equation:

$$z_{t,i,j} = \frac{\sum_{i=1}^{n_i}(x_{t,i,j})}{n_i} \tag{13}$$

where $\boldsymbol{x}_{t,i}$ represents a data point and n_i is the number of data points in the ith cluster.

2) reassign each data point to the cluster with the nearest cluster center.

Actually, the k-means operator locally exploits the search space for fine-tuning partitions found by deEC. And we believed it can improve the convergence properties.

The pseudo-code for deEC is given in Algorithm1.

IV. Experiments

In this section, we report the experimental results of deEC on synthetic and real-world data sets. The evolutionary version of traditional k-means [8] is implemented as comparison with deEC1 and deEC2. And a baseline called INST was designed, which employs k-means algorithm to independently cluster the data at only time step t and ignoring all historical data before t. We mainly observe these three methods and compare their snapshot cost, history cost and overall cost at each time step. For all the costs, the lower value is better. Besides, we utilize the NMI metric [32] to evaluate the cluster performance. NMI ranges in [0,1]. A high NMI value indicates that the cluster solution matches the ground truth well.

A. Synthetic Data

We use two experiments on synthetic data to illustrate the good performance of the deEC algorithm in different situation. In the first experiment, We add a Gaussian noise following N(0, 0.25) to the initial data set to simulate a situation where the data evolve smoothly. In the second experiment, besides adding a Gaussian noise following N(0, 0.5), we rotate all the data points by a small random angle to simulate another situation where the relationship among data points drifts significantly. Through these two experiments, we demonstrate the deEC is capable to adapt itself in different situation.

We modify the Iris data sets to meet the need of the experiments. The Iris data sets consists of three different species of iris flower (3 classes). For each species, 50 samples with 4 features were collected (each cluster contains 50 objects). In the first experiment, for each time step, we perturb the initial data by adding a Gaussian noise with different random seeds to each data point. In the second experiment, besides adding a Gaussian noise, we rotate all the data points by a small random angle, so that the relative positions among data points drift. For parameter setting, the population size is set 40, namely 10 times the number of features of the data set. The maximum number of generation is set 15. The scale factor F and mutation rate Cr has been described in equations (11) and (12), respectively. Unless stated otherwise, all experiments are run 30 times independently and the average performances are given.

In the first experiment, A Gaussian noise following N(0, 0.25) is added to the initial data set for each time step. By this way, we simulate a stationary situation in which data evolve smoothly and do not drift too much. Table I to Table III show the performance of the three methods. For all the costs, a lower value is better. As shown in these tables, the proposed methods deEC1 and deEC2 are better than the other two. On the whole, INST has a lower snapshot cost than kmEC while its history cost is higher than kmEC. This is because INST only considers the current data. However, their difference is unremarkable for the data drift smoothly. It surprises us that both the deEC1 and deEC2 have lower snapshot cost than INST. Since deEC1 and deEC2 take the smoothness property into account and they must trade off the accuracy of clustering the current data with consistence over the time. We try to make an explanation from the global optimization perspectives. The deEC1 and deEC2 can thoroughly explore the search space and avoid stuck at the local optima, so they can stably provide a global optimum. This may be one of the reason why the snapshot quality of deEC1 and deEC2 are better than INST on the average. It is interesting that deEC1 has lower history cost than deEC2, while deEC2 has lower snapshot cost than deEC1. This is because deEC1 focuses more on smoothness property, while deEC2 focuses more on snapshot quality. From Table IV, we can observe that deEC1 and deEC2 have the better NMI values, which means they have the better cluster performance.

In the seconde experiment, from each time step, besides adding a Gaussian noise following N(0, 0.5), we rotate each data point by a $\pi/6$ degree angle. By this way, we simulate a non-stationary situation in which the relationships among data points drift over time. The results given in Table V to Table VII illustrate both deEC1 and deEC2 are still better than the other two. It can be observed that, due to the data drift significantly, the temporal smoothness of INST becomes poor. This is because the true cluster may have been deviate from the recent past significantly. Table VI reports that both deEC1 and deEC2 can provide low history cost at all the time steps. The NMI values reported in Table VIII demonstrate the cluster performance of both deEC1 and deEC2 are better than the other two. The second experimental results demonstrate that, if the data set drift significantly over time, incorporating the smoothness property in the algorithm can produce a smooth result from one time step to the next.

B. Real Data

To demonstrate the performance of the proposed method on real data set, we choose pendigits data set which is publicly available in UCI repository. Pendigits data set is created to classify human being's handwriting. The data set consists of ten classes. Each samples contains 16 features. There are totally 7500 samples including 6 ones are originally used for testing. We divide the data set into 10 groups to simulate the corresponding 10 time steps, each of which has 750 samples randomly selected from each cluster. By this way, each group still consists of ten classes. For parameter setting, the population size is set 160, namely 10 times the number of features. The generation number is set 20. The α and F as well as Cr remain the same as before.

All the performance comparisons among the three methods are reported in Table IX - Table XII. Contrary to the experiments on synthetic data, the difference between deEC and the other two methods is remarkable. Clearly, the proposed methods deEC1 and deEC2 outperform the other two. Figure IX shows that the snapshot cost of both deEC1 and deEC2 are

TABLE I. SNAPSHOT COST IN STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.25)$

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	117.530	117.529	117.682	123.580	119.731	121.214	119.391	120.997	117.639	126.346
deEC2	**117.530**	**117.519**	**117.656**	**123.236**	**119.669**	**120.748**	**119.359**	**120.899**	**117.608**	**125.241**
INST	122.869	119.399	123.940	128.469	124.743	126.810	124.715	125.380	121.395	129.231
kmEC	122.863	122.803	122.264	127.746	126.158	128.243	123.251	121.121	121.440	128.522

TABLE II. HISTORY COST IN STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.25)$

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	0	**118.642**	**118.313**	**118.039**	**124.282**	**120.288**	**122.427**	119.519	**121.838**	**118.423**
deEC2	0	118.645	118.476	118.490	124.382	122.121	122.633	**119.509**	121.974	120.093
INST	0	120.845	124.880	123.785	130.083	126.037	128.665	124.244	125.483	124.059
kmEC	0	124.393	122.863	122.894	131.448	127.548	127.147	119.785	125.551	123.426

TABLE III. OVERALL COST IN STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.25)$

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**117.530**	117.682	117.785	**119.756**	120.291	**120.596**	119.768	**119.993**	118.155	**120.862**
deEC2	117.530	**117.534**	**117.678**	122.785	**119.700**	120.768	**119.388**	120.770	**117.631**	124.751
INST	122.869	119.543	124.034	128	125.277	126.733	125.110	125.266	121.804	128.713
kmEC	110.577	122.962	122.324	127.260	126.687	128.174	123.640	120.988	121.851	128.012

TABLE IV. THE NMI IN STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.25)$

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**0.729**	0.766	**0.722**	0.721	**0.733**	0.716	**0.787**	0.770	**0.721**	0.811
deEC2	0.729	**0.767**	0.722	**0.729**	0.732	**0.719**	0.763	**0.773**	0.72	**0.823**
INST	0.698	0.727	0.699	0.699	0.704	0.666	0.716	0.729	0.719	0.745
kmEC	0.705	0.742	0.696	0.697	0.698	0.708	0.678	0.747	0.720	0.695

TABLE V. SNAPSHOT COST IN NON-STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.5)$ AND ROTATED BY A $\pi/6$ DEGREE ANGLE

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**144.541**	147.677	143.273	152.847	149.910	145.292	143.659	150.385	142.51	155.670
deEC2	**144.541**	**146.816**	**143.174**	**152.440**	**149.800**	**145.236**	**143.581**	**149.758**	**142.398**	**154.432**
INST	147.917	148.897	146.601	154.423	150.487	149.823	146.646	152.733	144.930	155.079
kmEC	148.699	149.776	146.537	154.676	151.466	149.018	145.514	150.969	144.467	155.647

TABLE VI. HISTORY COST IN NON-STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.5)$ AND ROTATED BY A $\pi/6$ DEGREE ANGLE

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	0	**146.109**	**148.720**	**144.440**	**154.254**	**152.591**	**149.868**	**144.193**	**152.377**	**144.178**
deEC2	0	148.177	149.489	145.127	156.202	152.951	150.487	145.217	152.834	145.855
INST	0	151.137	150.990	147.947	156.589	155.665	153.475	150.089	153.646	146.683
kmEC	0	152.665	150.712	148.693	157.136	155.069	152.142	147.556	153.139	147.002

TABLE VII. OVERALL COST IN NON-STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.5)$ AND ROTATED BY A $\pi/6$ DEGREE ANGLE

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**144.541**	**146.676**	143.926	**147.146**	150.432	146.168	144.404	**146.236**	143.694	**147.866**
deEC2	**144.541**	146.838	**143.203**	151.748	**149.883**	**145.295**	**143.630**	149.325	**142.468**	153.602
INST	147.917	149.121	147.040	153.775	151.097	150.407	147.329	152.469	145.802	154.239
kmEC	133.829	150.065	146.954	154.078	152.033	149.623	146.177	150.627	145.335	154.782

TABLE VIII. THE NMI IN NON-STATIONARY SITUATION WITH GAUSSIAN NOISE $(0, 0.5)$ AND ROTATED BY A $\pi/6$ DEGREE ANGLE

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**0.694**	**0.692**	**0.665**	0.684	**0.669**	0.667	**0.616**	0.613	**0.708**	**0.706**
deEC2	0.693	0.691	0.660	**0.720**	0.661	**0.669**	0.602	**0.625**	0.658	0.658
INST	0.645	0.678	0.647	0.650	0.656	0.621	0.599	0.601	0.635	0.663
kmEC	0.643	0.655	0.644	0.641	0.629	0.640	0.599	0.600	0.662	0.618

significantly lower than INST and kmEC. It can be observe that, the NMI values of deEC1 and deEC2 are relatively high at almost all the time steps, while the history cost of deEC1 and deEC2 are lower than the other two. That means, to cluster the real world time-evolving data set, both the deEC1 and deEC2 can provide a faithful cluster solution at each time step, meanwhile keep stable and consistent with the recent past.

V. Conclusions and Future Works

In this paper, we deal with evolutionary clustering problem. An evolutionary clustering algorithm based on DE (deEC) is developed. The deEC, which includes deEC1 and deEC2, could perform a global search in the solution space. The deEC1 focuses on the search in the solution space near the historical solutions at the beginning but gradually adjusts the movements of the solutions during the later stages of search, so that they can reflect the current data as well as possible. Contrary to deEC1, the deEC2 focuses on the snapshot quality at the beginning, but is gradually biased toward searching in the history solution space to ensure the temporal smoothness. The experimental results illustrate both the deEC1 and deEC2 can provide robust and adaptive solutions. The deEC1 has lower history cost than deEC2, while the deEC2 has better snapshot quality than deEC1.

Considering that this is the tentative work, we assume the cluster number K is known in advance at each time step. Actually, based on DE, the evolutionary clustering algorithm could automatically determines the cluster number at each time step with the special modification of the chromosome representation schemes. This will enables the algorithm to respond to the more complicated situation in which the cluster number is unknown in advance and it can be various at different time steps. However, more issues have to be considered. For example, the clustering validity index should be non-monotonic with the number of clusters. And a new method to measure the temporal smoothness is needed when the cluster numbers varies. These will left for our future work.

Acknowledgement

This work is partly supported by the 2006-2007 Excellent Young and Middle-aged Academic Leader Development Program of Anhui Province Research Experiment Bases.

References

[1] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. Wiley Series in Probability and Mathematical Statistics, 1990, vol. 344.

[2] B. S. Everitt, S. Landau, and M. Leese, "Cluster analysis," *Arnold Publishers*, 2001.

[3] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[4] V. Vapnik, *The nature of statistical learning theory*. springer, 1995.

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases*. VLDB Endowment, 2003, pp. 81–92.

[6] M. M. Gaber and P. S. Yu, "Detection and classification of changes in evolving data streams," *International Journal of Information Technology and Decision Making*, vol. 5, no. 04, pp. 659–670, 2006.

[7] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM International Conference on Data Mining*, 2006, pp. 328–339.

[8] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 554–560.

[9] A. J. Owens, M. J. Walsh, and L. J. Fogel, *Artificial intelligence through simulated evolution*, 1966.

[10] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[11] J. Branke, *Evolutionary optimization in dynamic environments*. Kluwer Academic Publishers, 2001.

[12] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[13] J. Ma, Y. Wang, M. Gong, L. Jiao, and Q. Zhang, "Spatio-temporal data evolutionary clustering based on moea/d," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*. ACM, 2011, pp. 85–86.

[14] J. Zhang, Z. Ghahramani, and Y. Yang, "A probabilistic model for online document clustering with application to novelty detection," in *Advances in Neural Information Processing Systems*, 2004, pp. 1617–1624.

[15] W. Pedrycz and K.-C. Kwak, "The development of incremental models," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 3, pp. 507–518, 2007.

[16] C. Gupta and R. Grossman, "Genic: A single pass generalized incremental algorithm for clustering," in *Proceedings of the SIAM International Conference on Data Mining*, 2004, pp. 137–153.

[17] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997, pp. 626–635.

[18] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.

[19] T. Xu, Z. Zhang, P. S. Yu, and B. Long, "Evolutionary clustering by hierarchical dirichlet process with hidden markov state," in *ICDM'08. Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 658–667.

[20] ——, "Dirichlet process based evolutionary clustering," in *ICDM'08. Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 648–657.

[21] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 153–162.

[22] L. Wang, M. Rege, M. Dong, and Y. Ding, "Low-rank kernel matrix factorization for large-scale evolutionary clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 6, pp. 1036–1050, 2012.

[23] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 29, no. 3, pp. 433–439, 1999.

[24] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 38, no. 1, pp. 218–237, 2008.

[25] S. Bandyopadhyay and U. Maulik, "An evolutionary technique based on k-means algorithm for optimal clustering in rn," *Information Sciences*, vol. 146, no. 1, pp. 221–237, 2002.

[26] ——, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern Recognition*, vol. 35, no. 6, pp. 1197–1208, 2002.

[27] E. Falkenauer, *Genetic algorithms and grouping problems*. John Wiley and Sons, Inc., 1998.

[28] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern recognition*, vol. 33, no. 9, pp. 1455–1465, 2000.

[29] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

TABLE IX. SNAPSHOT COST ON PENDIGITS DATA SET

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**47103.80**	**46660.60**	47197.72	**46394.40**	**46149.60**	48114.36	**47193.30**	47644.54	47833.86	**46634.00**
deEC2	47105.60	46673.50	**47195.60**	46398.86	46151.90	**47950.10**	47202.14	**47644.20**	**47827.60**	46641.17
INST	48262.23	47936.24	48869.25	47694.33	47040.36	48942.78	48160.98	48596.15	49040.65	48220.90
kmEC	48230.76	47556.87	48452.69	48104.57	47149.14	48829.76	48385.52	48638.13	48991.31	47890.41

TABLE X. HISTORY COST ON PENDIGITS DATA SET

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	0	**48393.1**	**47632.5**	**48363.7**	**47365.2**	**47265.8**	48849.17	48381.77	**49023.1**	**48865.1**
deEC2	0	48397.48	47639.97	48370.02	47449.45	47458.35	**48828.50**	**48377.10**	49070.01	48949.04
INST	0	49627.23	48890.99	49483.25	48719.98	48579.12	49898.38	49176.27	49632.86	50401.01
kmEC	0	49137.06	48661.13	49789.27	49002.57	48232.92	50074.1	49154.53	49511.93	50000.19

TABLE XI. OVERALL COST ON PENDIGITS DATA SET

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**47103.80**	46868.47	47252.37	46630.73	46296.63	**47645.50**	47392.00	47733.01	47976.57	46901.76
deEC2	47105.60	**46719.50**	**47217.30**	**46469.30**	**46197.10**	47903.96	**47252.80**	**47674.30**	**47870.80**	**46698.80**
INST	48262.23	48105.34	48871.43	47873.22	47208.33	48906.41	48334.72	48654.17	49099.87	48438.91
kmEC	48230.76	47714.89	48473.53	48273.04	47334.48	48770.08	48554.37	48689.77	49043.38	48101.38

TABLE XII. THE NMI ON PENDIGITS DATA SET

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
deEC1	**0.675**	**0.730**	0.705	**0.743**	**0.720**	0.689	**0.710**	0.666	0.717	**0.722**
deEC2	**0.675**	0.729	0.704	0.737	0.719	0.691	0.709	0.666	**0.720**	0.713
INST	0.673	0.691	0.688	0.703	0.695	0.672	0.697	0.672	0.687	0.683
kmEC	0.674	0.668	**0.713**	0.686	0.691	**0.706**	0.692	**0.684**	0.673	0.692

[30] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224–227, 1979.

[31] R. Storn and K. Price, "Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[32] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.