A Point Symmetry-based Automatic Clustering Approach Using Differential Evolution

Wenyin Gong[1], Zhihua Cai[1], Charles X. Ling[2], and Bo Huang[1]

[1] School of Computer Science,
China University of Geosciences, Wuhan 430074, P.R. China,
cug11100304@yahoo.com.cn, zhcai@cug.edu.cn
[2] Department of Computer Science,
The University of Western Ontario, London, Canada,
cling@csd.uwo.ca

Abstract. Clustering is a core problem in data mining and machine learning. It has innumerable applications in many fields. Recently, using the evolutionary algorithms for the clustering problem has become more and more popular. In this paper, we propose an automatic clustering differential evolution (DE) technique for the clustering problem. Our approach can be characterized by (i) proposing a modified point symmetry-based cluster validity index (CVI) as a measure of the validity of the corresponding partitioning, (ii) using the Kd-tree based nearest neighbor search to reduce the complexity of finding the closest symmetric point, and (iii) employing a new representation to represent an individual. Experiments have been conducted on 6 artificial data sets of diverse complexities. And the results indicate that our approach is suitable for both the symmetrical intra-clusters and the symmetrical inter-clusters. In addition, our approach is able to find the optimal number of clusters of the data. Furthermore, based on the comparison with the original point symmetry-based CVI, our proposed point symmetry-based CVI shows better performance in terms of the F-measure and the number of clusters found.

1 Introduction

Clustering is the unsupervised classification of objects (patterns) into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in the same clusters is similar and the data in different clusters is dissimilar according to some defined distance measure. Data clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis, and bioinformatics [1], [2].

Clustering techniques may broadly be divided into two categories: hierarchical and partitional clustering [3], [4]. Hierarchical clustering algorithms generate a cluster tree by using heuristic splitting or merging techniques. Algorithms that use splitting to generate the cluster tree are called divisive. On the other hand, the more popular algorithms that use merging to generate the cluster tree are called agglomerative [2]. There are two main advantages of the hierarchical clustering algorithms: i) the number of clusters need not to be specified a priori, and ii) they are independent of the initial conditions [2].

However, the main drawbacks of these algorithms are: i) they are static; that is, data points assigned to a cluster can not move to another cluster; ii) they may fail to separate overlapping clusters due to a lack of information about the global shape or size of the clusters; and iii) they are computationally expensive. On the other hand, partitional clustering algorithms try to decompose the data set directly into a set of disjoint clusters. They try to optimize certain criteria. The advantages of the hierarchical algorithms are the disadvantages of the partitional algorithms, and vice versa. Two extensive surveys of the clustering algorithms can be found in [1] and [2].

Since a priori knowledge is generally not available, it is difficult to estimate the exact number of clusters from the given data set. Recently, many automatic clustering algorithms based on evolutionary algorithms (EAs) have been introduced [5], [6], [7], [8], etc. Based on some clustering validity index (CVI) [9], these techniques are more efficient than the traditional method.

Differential evolution (DE) [10] algorithm is a novel evolutionary algorithm for faster optimization, which mutation operator is based on the distribution of solutions in the population. Among DE's advantages are its simple structure, ease of use, speed and robustness. Based on the successful applications of DE [11], [12] in many fields, some researchers adopted it to solve the clustering problems [13], [14], [7]. Experimental results have shown that the DE-based clustering algorithms can provide higher performance than GA-based clustering algorithms. However, the work of using DE for clustering problems is still preliminary. In addition, most of the previous work need to give the number of clusters in advance.

In this paper, in order to automatically determine the optimal number of clusters in the data set, we propose an automatic clustering DE technique based on the point symmetry-based CVI. Our approach is referred as ACDEPS. It is characterized by (i) proposing a modified point symmetry-based cluster validity index (CVI) as a measure of the validity of the corresponding partitioning, (ii) using the Kd-tree based nearest neighbor search to reduce the complexity of finding the closest symmetric point, and (iii) employing a new representation to represent an individual.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce the clustering problem definition and the DE algorithm. Our proposed approach is presented in detail in Section 3. In Section 4, we verify our approach through 6 artificial data sets of diverse complexities. The last section, Section 5, is devoted to conclusions and future work.

2 Preliminary

2.1 Problem Definition

Generally, a clustering problem can be formally defined as follows [2]: Given a data set $\boldsymbol{X} = \{x_1, x_2, \cdots, x_n\}$, where n is the number of patterns in $\boldsymbol{X}$, x_i is a pattern in a d-dimensional feature space, then the clustering of $\boldsymbol{X}$ is the partitioning of $\boldsymbol{X}$ into k clusters $\{C_1, C_2, \cdots, C_k\}$ satisfying the following conditions:

- Each pattern should be assigned to a cluster, i.e. $\cup_{i=1}^{k} C_i = \boldsymbol{X}$.
- Each cluster has at least one pattern assigned to it, i.e. $C_i \neq \phi$, for $i = 1, 2, \cdots, k$.

– Each pattern is assigned to one and only one cluster (in case of hard clustering only), i.e. $C_i \cap C_j = \phi$, for $i = 1, 2, \cdots, k$, $j = 1, 2, \cdots, k$, and $i \neq j$.

2.2 Differential Evolution

The DE algorithm [10] is a simple EA that creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness. This is a rather greedy selection scheme that often outperforms traditional EAs. In addition, DE is a simple yet powerful population-based, direct search algorithm with the generation-and-test feature for globally optimizing functions using real-valued parameters. Among DE's advantages are its simple structure, ease of use, speed and robustness. Due to these advantages, it has many real-world applications, such as data mining [15], [7], pattern recognition, digital filter design, neural network training, etc. [11], [12].

The DE algorithm in pseudo-code is shown in Algorithm 1. d is the number of decision variables, NP is the size of the parent population P, F is the mutation scaling factor, CR is the probability of crossover operator, U^i is the offspring, rndint$(1, d)$ is a uniformly distributed random integer number between 1 and n, and $\mathrm{rnd}_j[0, 1)$ is a uniformly distributed random real number in $[0, 1)$. Many schemes of creation of a candidate are possible. We use the DE/rand/1/bin scheme (see lines 6 - 13 of Algorithm 1) described in Algorithm 1 (more details on DE/rand/1/bin and other DE schemes can be found in [16] and [11]).

Algorithm 1 DE algorithm with DE/rand/1/bin

```
 1: Generate the initial population
 2: Evaluate the fitness for each individual
 3: while The halting criterion is not satisfied do
 4:    for i = 1 to NP do
 5:       Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
 6:       j_rand = rndint(1, d)
 7:       for j = 1 to d do
 8:          if rnd_j[0, 1) > CR or j == j_rand then
 9:             U_j^i = X_j^{r1} + F × (X_j^{r2} − X_j^{r3})
10:          else
11:             U_j^i = X_j^i
12:          end if
13:       end for
14:       Evaluate the offspring U^i
15:       if U^i is better than P^i then
16:          P^i = U^i
17:       end if
18:    end for
19: end while
```

From Algorithm 1, we can see that there are only three control parameters in this algorithm. These are NP, F and CR. As for the terminal conditions, one can either

fix the maximum number of fitness function evaluations (NFFEs) *Max_NFFEs* or the precision of a desired solution *VTR* (value to reach).

2.3 Point Symmetry-based Distance Measures

In natural senses, symmetry is one of the basic feature of shapes and objects, and hence, it is reasonable to assume some kinds of symmetry exist in the structures of clusters. Based on this idea, some symmetry-based distance measures are proposed in literature recently [17], [18], [19]. Since in this work we only employ the point symmetry-based distance measure proposed in [19], we will briefly discuss this measure in the following.

Recently, Bandyopadhyay and Saha proposed a genetic clustering technique based on a new point symmetry-based distance measure [19]. In addition, they adopted the Kd-tree based nearest neighbor search method to reduce the complexity of finding the most symmetrical point. The proposed point symmetry-based distance measure is defined by

$$d_{ps}(x_i, c_t) = \frac{d_1 + d_2}{2} \times d_e(x_i, c_t) \tag{1}$$

where $d_e(x_i, c_t)$ is the Euclidean distance between the pattern x_i and the cluster centroid c_t, d_1 and d_2 are the first and the second unique nearest neighbors of the symmetrical point (i.e. $2 \times c_t - x_i$) of x_i with respect to a particular center c_t, respectively. To reduce the complexity of finding d_1 and d_2, an ANN search using the Kd-tree method is used. After applying the genetic clustering technique based on point symmetry distance measure (GAPS) to different types of data sets, they concluded that the GAPS method is able to detect any type of clusters as long as they possess the characteristic of symmetry.

In [19], the authors pointed out that the complexity of assigning the points to the different clusters is $O(kn^2)$ when adopting the point symmetry-based distance measure. In order to reduce the complexity, the Kd-tree based nearest neighbor search technique, which reduces the complexity from $O(kn^2)$ to $O(kn \log(n))$, is adopted. ANN is a library written in C++ [20], which supports data structures and algorithms for both exact and approximate nearest neighbor searching in arbitrarily high dimensions.

2.4 Point Symmetry-based CVI

For automatic clustering techniques, there are two fundamental questions that need to be addressed: i) how many clusters are actually present in the data, and ii) how real or good is the clustering itself [9]. To measure the goodness of the clustering result, the cluster validity index (CVI) is used to evaluate the results of a clustering algorithm on a quantitative basis. In [8], they proposed a point symmetry-based CVI and defined as:

$$Sym(k) = \left(\frac{1}{k} \times \frac{1}{\xi_k} \times D_k\right) \tag{2}$$

Where,

$$\xi_k = \sum_{i=1}^{k} E_i \tag{3}$$

such that

$$E_i = \sum_{j=1}^{n_i} d_{ps}^*(x_j^i, c_i) \tag{4}$$

and

$$D_k = max_{i,j=1}^{k} \parallel c_i - c_j \parallel \tag{5}$$

k is the number of clusters. D_k is the maximum Euclidean distance between two cluster centers among all pairs of centers. $d_{ps}^*(x_j^i, c_i)$ is computed by Equation 1 with some constraint. Here, the first $knear$ nearest neighbors of $2 \times c_i - x_j$ will be searched among only those points with are in cluster i. The objective is to maximize this index in order to obtain the actual number of clusters. More details about this index can be found in [8].

3 Our approach: ACDEPS

As above-mentioned, DE is a simple and versatile global optimizer. The DE-based clustering algorithms can provide higher performance than GA-based clustering algorithms [13]. Motivated by this idea, in this work, we propose an automatic clustering DE approach for the clustering problems. Our approach is referred as ACDEPS, i.e., Automatic Clustering DE based on Point Symmetry-based measure. It is explained in detail in the following subsections.

3.1 Individual Representation

In our approach, the individual representation proposed in [7] is used. For n data points, each d dimensional, and for a maximum number of clusters $k_{max} = \sqrt{n}$ [6], an individual is a vector of real numbers of dimension $k_{max} + k_{max} \times d$. It is defined as:

$$X_i = \left(T_{i,1}, T_{i,2}, \cdots, T_{i,k_{max}}, c_{i,1}, c_{i,2}, \cdots, c_{i,k_{max}}\right)^T \tag{6}$$

Where the first k_{max} entries are positive floating-point numbers in $[0, 1]$, each of which controls whether the corresponding cluster is to be used or not. Only if $T_{i,j} > 0.5$, then the j-th cluster center in the i-th individual is active. The rest entries are the d-dimensional cluster centers.

3.2 Modified Point Symmetry-based CVI

From Equation 2 we can see that Sym-index is a composition of three factors, $1/k$, $1/\xi_k$, and D_k. The first factor increases as k decreases. The second and the third factors decrease as k decreases. Since the Sym-index needs to be maximized for optimal clustering, only the first factor prefers to decrease k. While the other two factors prefer to increase k. Thus, in the beginning of the evolutionary process, the individuals prefer to find more clusters. In order to make the algorithm find the optimal cluster centers faster

and then obtain the optimal partitioning, in this work, we propose a modified point symmetry-base CVI, where the k is penalized dynamically. It is described as follows.

$$Sym'(k) = \left(\frac{1}{k'} \times \frac{1}{\xi_k} \times D_k\right) \tag{7}$$

Here, $1/\xi_k$ and D_k are defined by Equations 3 and 5, respectively. k' is defined as:

$$k' = k^{2^t} \tag{8}$$

and

$$t = 1.0 - \alpha \times \frac{gen}{G_{max}} \tag{9}$$

where gen is the current generation number. G_{max} is the maximum generations. $\alpha \geq 1$ is a constant; it controls the dynamic penalty of k. When $\alpha = 1$, it means that k is penalized in the entire evolution, except for the last generation. When $\alpha = 2$, it indicates that k is penalized if $gen < 0.5 \times G_{max}$; while when $gen \geq 0.5 \times G_{max}$, k is not penalized, i.e. $k' = k$. Based on the penalized dynamic Sym'-index, the algorithm is able to avoid finding too more clusters in the beginning of the evolutionary process.

3.3 Avoiding Erroneous Individuals

In our approach, calculation of the Sym'-index needs to find the first and the second symmetrical points. In this work, the Kd-tree based nearest neighbor search method is employed to find the two points, and hence, there are at least two data points for each cluster. For an individual if any cluster has fewer than two dada points in it, the individual is reinitialized to k randomly selected points from the data set. After the special individual is reinitialized, all data points are reassigned to this individual.

Furthermore, when a new offspring is created according to DE/rand/1/bin strategy as shown in Algorithm 1, if some $T_{i,j}$ in the offspring is greater than 1 or less than 0, it is forcefully fixed to 1 or 0, respectively. If the number of $T_{i,j} \geq 0.5$ is less than 2, we randomly select two $T_{i,j}$ and reinitialize them to a random value in $[0.5, 1.0]$. Thus, the minimum number of clusters is 2.

4 Experimental results and analysis

To evaluate the performance of our approach, we test the ACDEPS approach for both Sym'-index and Sym-index, then the two methods are referred to as ACDEPS1 and ACDEPS2, respectively. Moreover, we compare the two ACDEPS methods with GCUK proposed in [5][3]. To make a fair comparison, in GCUK, the Sym'-index and Sym-index are also used; they are referred to as GCUK1 and GCUK2, respectively.

[3] Since we can not obtain the VGAPS method [8] code, we don't make a comparison with VGAPS.

4.1 Experimental Setup

In our proposed approach, there are four parameters to be specified: i) the population size NP, ii) the crossover probability CR, iii) the maximum number of generations G_{max}, and iv) the dynamic control factor α. In the original DE, the scaling factor F requires to be specified in advance. However, in this work, the dither technique is used to avoid tuning this parameter, where F is uniformly distributed random number generated from $[0.0, 1.0]$. The reason is that the dither technique can improve the performance of DE [11], [12]. Moreover, it can avoid tuning this parameter for the user. For all experiments, we use the following parameters unless a change is mentioned. For GCUK, the parameter settings are used as mentioned in [5].

- Population size: $NP = 100$;
- Crossover probability: $CR = 0.3$;
- DE scheme: DE/rand/1/bin;
- Dynamic control factor: $\alpha = 2.0$;
- Maximum generations: $G_{max} = 30$.

In our experiments, each data set is optimized over 10 independent runs. We also use the same set of initial random populations to evaluate different algorithms. All the algorithms are implemented in standard C++ and the experiments are done on a P-IV (Core 2) 2.1 GHz laptop with 1.0 GB RAM under WIN-XP platform.

4.2 Data Sets

In order to validate the performance of ACDEPS, we have carried out different experiments using a test suite, which consists of 6 artificial data sets of diverse complexities chosen from literature. They are artificial data sets and are briefly described as follows.

- **data1**: This data set has been used in [8]. It consists of two crossed ellipsoid shells, where each ellipsoid shell contains 200 data points.
- **data2**: This data set, used in [17], [19], [8], is a combination of ring-shaped, compact and linear clusters. It contains 350 data points.
- **data3**: This data set, used in [19], contains 400 data points and three clusters, which consists of a ring-shaped cluster, a rectangular cluster and a linear cluster.
- **data4**: This data set, used in [19], [8], is in 3-d space, and has 4 hyper-spherical disjoint clusters. The total number of points is 400.
- **data5**: This data set has 6 different clusters in 2-d space. It contains 300 data points used in [19] and [8].
- **data6**: This data set, used in [8], contains 850 data points distributed on five clusters.

4.3 Performance Criteria

To compare the performance of the four algorithms, three performance criteria are selected to evaluate the performance of the algorithms. These criteria are described as follows.

Table 1. Comparison of the F-measure value for the four algorithms after $3,000$ NFFEs. Where "Mean" indicates the mean f-measure values found in the last generation; "Std Dev" stands for the standard deviation. The best results are highlighted in **Bold** face.

Dataset	ACDEPS1			GCUK1			ACDEPS2			GCUK2		
	Mean	Std Dev	SR	Mean	Std Dev	SR	Mean	Std Dev	SR	Mean	Std Dev	SR
data1	**0.982**	0.015	**1.0**	0.084	0.266	0.1	0.492	0.519	0.5	0.000	0.000	0.0
data2	**0.942**	0.021	**1.0**	0.645	0.453	0.7	0.762	0.402	0.8	0.278	0.448	0.3
data3	0.999	0.001	**1.0**	**1.000**	0.000	**1.0**	0.300	0.482	0.3	0.100	0.316	0.1
data4	**1.000**	0.000	**1.0**	0.300	0.483	0.3	**1.000**	0.000	**1.0**	0.899	0.316	0.9
data5	**1.000**	0.000	**1.0**	0.800	0.242	0.0	**1.000**	0.000	**1.0**	0.900	0.316	0.9
data6	0.365	0.472	0.4	**0.699**	0.482	**0.7**	0.197	0.415	0.2	0.690	0.477	0.6

– **F-measure**: F-measure [21] is associated to the information retrieval field, recall and precision are measures that give us some idea of how well a clustering algorithm is identifying the classes present in the data set. In the context of classification, recall is define as $r(i,j) = n_{ij}/n_i$ where n_{ij} is the number if items of class i in cluster j and n_i is the number of elements of class i. Precision is defined as $p(i,j) = n_{ij}/n_j$ where n_j is the number of elements in cluster j. For a class i and cluster j the F-measure is define by

$$F(i,j) = \frac{2p(i,j)r(i,j)}{p(i,j) + r(i,j)} \tag{10}$$

The overall F-measure for the classification generated by the clustering algorithm is give by

$$F = \sum_{i=1}^{k} \left(\frac{n_i}{n} \max_j F(i,j) \right) \tag{11}$$

where n is the size of the data set. F is limited to the interval $[0,1]$ with a value of 1 with a perfect clustering.

– **Number of clusters**: The number of clusters found of each algorithm in the final generations. The average value and the standard deviation are calculated over 10 runs.
– **Successful rate (SR)**: If the algorithm can find the actual number of clusters of a given data set in one run, it means that the algorithm obtains a successful run. The SR value is the ratio of the successful runs over the total runs.

4.4 Experimental Results

In this section, we compare the performance of the four algorithms. The parameters used for ACDEPS1, ACDEPS2, GCUK1, and GCUK2 are described above. All data sets are conducted for 10 independent runs. The experimental results for the four algorithms are shown in Table 1 and Table 2, respectively. And some representative clustering results are illustrated in Fig. 1.

Table 2. Comparison of the number of clusters found by the four algorithms after $3,000$ NFFEs. Where "Mean" indicates the mean number of clusters found in the last generation; "Std Dev" stands for the standard deviation. The best results are highlighted in **Bold** face.

Dataset	ACDEPS1			GCUK1			ACDEPS2			GCUK2		
	Mean	Std Dev	SR	Mean	Std Dev	SR	Mean	Std Dev	SR	Mean	Std Dev	SR
data1	**2.000**	0.000	**1.0**	4.200	2.201	0.1	3.100	1.663	0.5	7.200	1.229	0.0
data2	**3.000**	0.000	**1.0**	2.700	0.483	0.7	3.200	0.422	0.8	5.600	1.897	0.3
data3	**3.000**	0.000	**1.0**	**3.000**	0.000	**1.0**	4.400	0.966	0.3	4.800	0.632	0.1
data4	**4.000**	0.000	**1.0**	2.600	0.966	0.3	**4.000**	0.000	**1.0**	4.100	0.316	0.9
data5	**6.000**	0.000	**1.0**	5.800	0.323	0.0	**6.000**	0.000	**1.0**	6.100	0.316	0.9
data6	5.800	0.789	0.4	**5.200**	0.422	**0.7**	7.000	1.333	0.2	4.100	1.449	0.6

From Table 1, it can be seen that ACDEPS1 can find the actual number of clusters for five out of six data sets over all 10 runs. And the average F-measure values for these five data sets (data1 - data5) are very close to 1.0. It indicates that ACDEPS1 can obtain the near-optimal partitioning of these data sets. When compared with ACDEPS2, we can see that ACDEPS1 is superior to ACDEPS2 in terms of both the F-measure and the success rate. This phenomenon means that our proposed Sym'-index is better than the original Sym-index when used in the ACDEPS method. Compared the results of ACDEPS with those of GCUK (ACDEPS1 vs GCUK1, and ACDEPS2 vs GCUK2), the results show that ACDEPS is better than GCUK on the majority of the data sets. Except for data6, GCUK performs better than ACDEPS.

From Table 2 we can see that for five data sets (data1 - data5) ACDEPS1 can automatically determine the optimal number of clusters over all 10 runs. It can also obtain the best results compared with ACDEPS2, GCUK1, and GCUK2.

In addition, from Fig. 1 it is apparent to see that for data1 to data5, ACDEPS1 can obtain both the optimal number of clusters and their corresponding optimal partitioning. However, for data6, ACDEPS1 is failed to cluster this data set.

In summary, for the majority of the data sets used in this work, our proposed ACDEPS1 approach is able to automatically determine the optimal number clusters, also it can obtain the optimal partitioning as long as the data sets possess the characteristic of symmetry. And the proposed Sym'-index can make the ACDEPS method more robust than the original Sym-index.

5 Conclusions and future work

The DE algorithm is a simple yet powerful evolutionary algorithm for global optimization. In this paper, we adopt the DE algorithm for the automatic clustering problem. To find the optimal number of clusters, we propose a modified Sym'-index, which can avoid finding too more number of clusters in the beginning of the evolutionary process. A new individual representation is employed to make DE suitable for the automatic clustering. In addition, the Kd-tree based nearest neighbor search is used to reduce the complexity of finding the closest symmetric point.

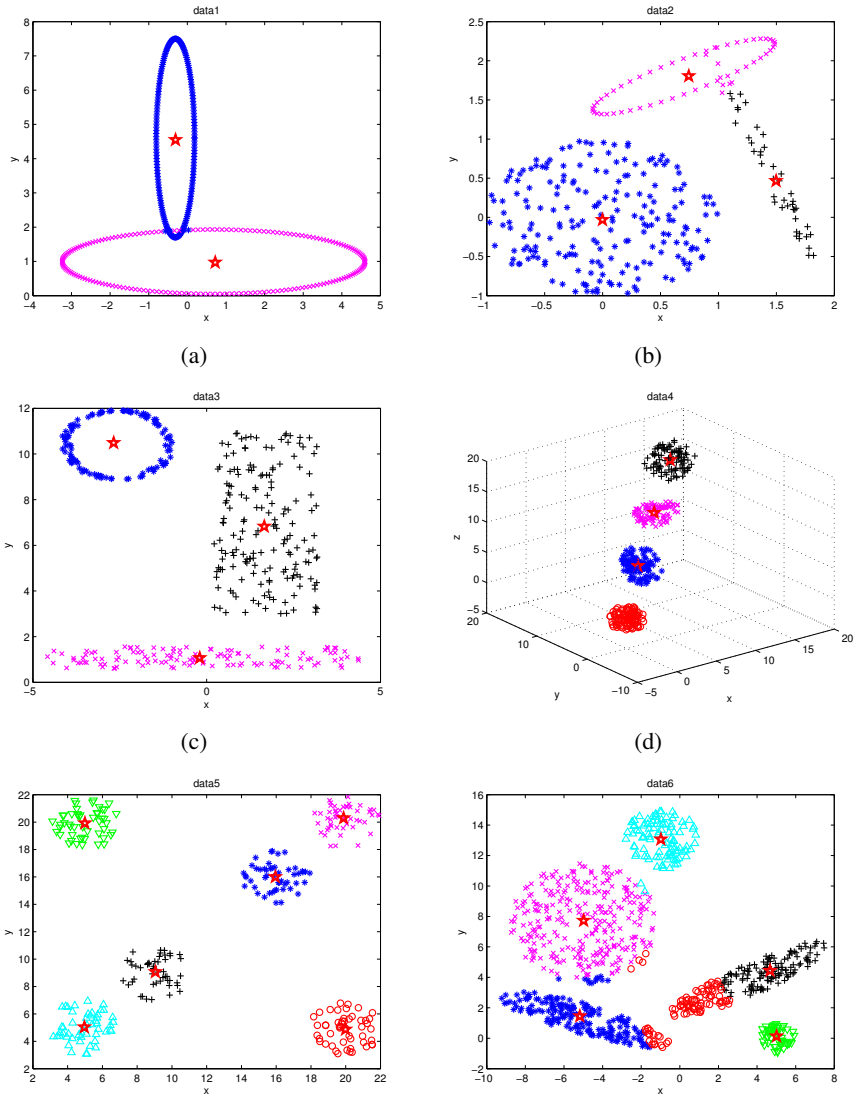

Fig. 1. Clustered results of ACDEPS1 for all data sets. (a) data1. (b) data2. (c) data3. (d) data4. (e) data5. (f) data6. The $\star$ indicates the cluster center.

In order to test the performance of our approach, 6 artificial data sets are chosen from literature. Experimental results indicate that our proposed ACDEPS1 approach is able to automatically determine the optimal number clusters, also it can obtain the optimal partitioning as long as the data sets possess the characteristic of symmetry. Furthermore, based on the comparison with the original Sym-index, our proposed Sym'-index shows better performance in terms of the F-measure and the number of clusters found.

In our proposed Sym'-index, an additional parameter α is used to control the dynamic penalty of k. It may be problem-dependent. Our future work will conduct further experiments, both for the artificial data sets and the real world data sets, to test the influence of this parameter.

Acknowledgments

The first author would like to thank Dr. S. Saha for providing the data sets and her suggestions on this work. This work was supported by the Fund for Outstanding Doctoral Dissertation of CUG, China Scholarship Council under Grant No. 2008641008, and the National High Technology Research and Development Program of China under Grand No. 2009AA12Z117.

References

1. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3) (1999) 264–323
2. Omran, M.G.H., Engelbrecht, A.P., Salman, A.: An overview of clustering methods. Intell. Data Anal. **11**(6) (2007) 583–605
3. Frigui, H., Krishnapuram, R.: A robust competitive clustering algorithm with applications in computer vision. IEEE Trans. Pattern Anal. Mach. Intell. **21**(5) (1999) 450–465
4. Leung, Y., Zhang, J.S., Xu, Z.B.: Clustering by scale-space filtering. IEEE Trans. Pattern Anal. Mach. Intell. **22**(12) (2000) 1396–1410
5. Bandyopadhyay, S., Maulik, U.: Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition **35**(6) (2002) 1197–1208
6. Sheng, W., Swift, S., Zhang, L., Liu, X.: A weighted sum validity function for clustering with a hybrid niching genetic algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B **35**(6) (2005) 1156–1167
7. Das, S., Abraham, A., Konar, A.: Automatic clustering using an improved differential evolution algorithm. IEEE Transaction on Systems Man and Cybernetics: Part A **38**(1) (2008) 218–237
8. Bandyopadhyay, S., Saha, S.: A point symmetry-based clustering technique for automatic evolution of clusters. IEEE Trans. on Knowl. and Data Eng. **20**(11) (2008) 1441–1457
9. Pakhira, M.K., Bandyopadhyay, S., Maulik, U.: Validity index for crisp and fuzzy clusters. Pattern Recognition **37**(3) (2004) 487–501
10. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4) (1997) 341–359
11. Price, K., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag, Berlin (2005)
12. Chakraborty, U.: Advances in Differential Evolution. Springer-Verlag, Berlin (2008)

13. Paterlini, S., Krink, T.: High performance clustering with differential evolution. In: In Proceedings of 2004 Congress on Evolutionary Computation, IEEE Press (2004) 2004–2011
14. Paterlini, S., Krink, T.: Differential evolution and particle swarm optimisation in partitional clustering. Comput. Stat. Data Anal **50** (2006) 1220–1247
15. Alatas, B., Akin, E., Karci, A.: Modenar: Multi-objective differential evolution algorithm for mining numeric association rules. Applied Soft Computing **8**(1) (2008) 646–656
16. Storn, R., Price, K.: Home page of differential evolution. avaiable online at: http://www.icsi.berkeley.edu/˜storn/code.html (2008)
17. Su, M.C., Chou, C.H.: A modified version of the k-means algorithm with a distance based on cluster symmetry. IEEE Trans. Pattern Anal. Mach. Intell. **23**(6) (2001) 674–680
18. Chung, K.L., Lin, J.S.: Faster and more robust point symmetry-based k-means algorithm. Pattern Recogn. **40**(2) (2007) 410–422
19. Bandyopadhyay, S., Saha, S.: Gaps: A clustering method using a new point symmetry-based distance measure. Pattern Recogn. **40**(12) (2007) 3430–3451
20. Mount, D., Arya, S.: Ann: A library for approximate nearest neighbor searching. avaiable online at: http://www.cs.umd.edu/˜mount/ann (2008)
21. van Rijsbergen, C.: Information Retrieval, (2nd Edition). Butterworths, London, UK (1979)