Nat Comput (2017) 16:45–59
DOI 10.1007/s11047-016-9542-9

A population-based clustering technique using particle swarm optimization and *k*-means

Ben Niu[1,2] · **Qiqi Duan**[1] · **Jing Liu**[1] · **Lijing Tan**[3] · **Yanmin Liu**[4]

Published online: 9 February 2016

Abstract A population-based clustering technique, which attempts to integrate different particle swarm optimizers (PSOs) with the famous *k*-means algorithm, is proposed. More specifically, six existing extensively studied PSOs, which have shown promising performance for continuous optimization, are hybridized separately with Lloyd's *k*-means algorithm, leading to six PSO-based clustering methods. These PSO-based approaches use different social communications among neighbors to make some particles escape from local optima to enhance exploration, while *k*-means is utilized to refine the partitioning results for accelerating convergence. Comparative experiments on 12 synthetic and real-life datasets show that the proposed population-based clustering technique can obtain better and more stable solutions than five individual-based counterparts in most cases. Further, the effects of four different population topologies, three kinds of parameter settings, and two types of initialization methods on the clustering performance are empirically investigated. Moreover, seven boundary handling strategies for PSOs are firstly summarized. Finally, some unexpected conclusions are drawn from the experiments.

Keywords Population-based clustering technique · Particle swarm optimization (PSO) · Lloyd's *k*-means

✉ Ben Niu
drniuben@gmail.com

✉ Lijing Tan
mstlj@163.com

1 College of Management, Shenzhen University, Shenzhen 518060, China

2 Department of Mechanical Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong

3 Department of Business Management, Shenzhen Institute of Information Technology, Shenzhen 518172, China

4 School of Mathematics and Computer Science, Zunyi Normal College, Zunyi 563002, China

1 Introduction

The classical clustering problems as well as diverse clustering techniques have been extensively studied by different disciplines such as machine learning, pattern recognition, data mining, image processing, and so on (Flynn et al. 1999). Typically, the data clustering problem that partitions N data points into K mutually exclusive, nonempty clusters can be modeled as a *k-means-type* continuous optimization model where the centroids of K clusters and cluster indexes of N data points act as inputs and outputs, respectively. The most commonly used partitioning clustering methods (e.g., the most representative one is *k*-means Jain 2010 which are sensitive to the initial seeds), easily get trapped into local optima, in particular when solving large-scale datasets (Tzortzis and Likas 2014). More effective and robust partitioning algorithms are urgently required to uncover the hidden patterns and structures for complex large-sized datasets.

Generally speaking, achieving one exact solution of large-scale clustering problem is theoretically possible yet computationally infeasible (Murthy and Chowdhury 1996), since the total number of feasible solutions increases exponentially with N and K (Chioua and Lan 2001). Therefore, a variety of approximate clustering methods have been designed and investigated. Recently, population-based clustering techniques, which are regarded as an important branch of approximate methods, become popularity, due to their attractive performance on many datasets. The advantages of the population-based methods over the individual-based counterparts (e.g., k-means) may lie in their promising global exploration resulted in by the parallel search and the robustness of final results obtained via multiple agents. Almost all population-based stochastic optimization techniques [e.g., genetic algorithms (Murthy and Chowdhury 1996; Chioua and Lan 2001), evolution strategies (Lee and Antonsson 2000), ant colony optimization (Handl et al. 2006), differential evolution (Das et al. 2008), artificial bee colony algorithms (Karaboga and Ozturk 2011), and bacterial foraging optimization (Niu et al. 2013), just to name a few] have been successfully applied to some clustering problems.

As a powerful optimization tool, the particle swarm optimizer (PSO) have been widely studied and improved by different researchers, due to its simplicity of implementation, few parameter configuration and global exploration ability on some complex problems. More detailed descriptions of diverse PSO versions can be founded in Sect. 3, where six commonly used PSOs are highlighted. A variety of PSO-based clustering methods (e.g., Merwe and Engelbrecht 2003; Chen and Fun 2012; Omran et al. 2005; Kao et al. 2008; Alam et al. 2008; Abbas et al. 2010; Szabo et al. 2010; Niknam and Amiri 2010; Chuang et al. 2011; Radha et al. 2011; Yuwono et al. 2014; Tsai et al. 2014) have been developed, as shown in Sect. 2. Usually, they can show successes at least on a subset of datasets which are picked by their corresponding papers. For most of them, however, the influences of different parameter settings, population topologies, boundary handling strategies, and/or population initialization methods on the clustering performance have not been comprehensively studied. In addition, although different PSOs show significantly different performances on benchmark continuous functions, it is still not clear whether such significant differences can still hold for diverse clustering optimization problems. Note that the real-world clustering problems may include some unique landscape characteristics, which do not be represented by most continuous benchmark functions, which will be discussed in Sect. 4. If the significant differences on the clustering performance do exist among diverse PSOs, the following question is which of PSO variants can obtain the best partitioning solutions for most clustering problems. These unsolved yet important problems for PSO-based clustering leave an open and broad space for further research.

Following this research line, in this paper, six PSO-based clustering techniques are designed in one unified algorithm framework and compared with each other on 12 synthetic and real-life datasets. Six existing PSO versions with different population structures and different parameter settings are taken into account (see Sect. 3 for more information). In the unified algorithm framework, six PSOs are hybridized separately with Lloyd's k-means (Lloyd 1982), leading to the six PSO-based clustering algorithms. The rationale of combining PSOs with k-means is simple and natural: (1). the potential global search ability of PSOs with slow convergence can complement the fast convergence of k-means with local search ability; and (2). early studies (e.g., Merwe and Engelbrecht 2003; Omran et al. 2005; Kao et al. 2008; Radha et al. 2011 which provide a base for this paper) have shown the superiority of such combination strategy. By comparative experiments, the effects of four different population topologies, three kinds of parameter settings, and two types of initialization methods on the clustering performance are empirically investigated. In addition, seven boundary handling strategies are firstly summarized in Sect. 3.

The remainder of the paper is organized as following. Section 2 reviews the applications of EAs (especially PSOs) to data partitioning problems. In Sect. 3, a population-based clustering algorithm framework is designed, which incorporates six existing PSOs with Lloyd's k-means separately. Section 4 conducts comparative experiments. Finally, conclusions and further research are drawn in Sect. 5.

2 Literature overview

Many research efforts have been devoted to the applications of evolutionary algorithms (EAs) on a variety of clustering problems. In this section, we provide a brief literature overview of EA-based clustering algorithms, but focus mainly on PSO-based techniques. Initially, genetic algorithms (GAs) were investigated to improve the performance of classic clustering algorithms and/or directly solve the clustering optimization problems. For instance, Murthy and Chowdhury (1996) firstly developed a GA-based clustering algorithm that encoded each partitioning as a real-valued string of length N. This encoding scheme needs expensive storage requirements and computational costs for medium-to-large scale datasets. The corresponding experiments were conducted only on three small datasets with $N < 100$, which seemingly cannot confirm its effectiveness on medium- and large-sized datasets. Another similar GA-based clustering approach was proposed by Chioua and Lan (2001), which adopted a binary-encoding representation way. The binary-encoding strategy leads to a large search space and increases computational complexity. Therefore, a proper way of representing the partitioning solution is very crucial to any EA-based clustering technique.

The particle swarm optimizer (PSO) has attracted increasing attention from the scientific and engineering communities, since it was originally proposed in Eberhart and Kennedy (1995). Due to its simplicity, PSO has been widely studied and applied to clustering problems by many researchers from both the data mining and evolutionary computation community. The hybridization strategy among different kinds of algorithms is an important research direction for the clustering optimization field. Usually, hybridization can unify the advantageous characteristics of different techniques while weakening their own drawbacks, and thus achieve good performance (Radha et al. 2011). Based on the above merits, many researchers have attempted to hybrid different PSO versions with diverse clustering methods and/or other optimization techniques. For example, (Merwe and Engelbrecht (2003) designed two simple methods for PSO-based clustering. The first one uses the partitioning result obtained by *k*-means to seed the population, while the second refines the solution of *k*-means by PSO. Both the methods will be further discussed and validated in this paper. A hybrid version, which incorporated EPSO with *k*-means, was proposed by Alam et al. (2008). However, the experiments could not effectively validate the performance of EPSO, since only a small dataset (i.e., the well-known *Ruspini* dataset that has 75 data points and 2 attributes) was chosen as the benchmark dataset. More datasets with more dimensions and more data points are expected to be tested. Kao et al. (2008) designed a hybridized clustering method called K–NM–PSO, which integrated *k*-means with PSO and the Nelder–Mead simplex search technique. Niknam and Amiri (2010) incorporated PSO with ACO and *k*-means, which resulted in a more complicated implementation procedure. Alam et al. (1995) integrated PSO with the hierarchical agglomerative clustering method which, however, is not suitable for large-scale clustering problems. The readers are encouraged to read the literature survey on hybrid PSO-based clustering algorithms provided in Radha et al. (2011), and Mohamed and Sivakumar (2011) for more details. Most hybridization strategies can improve the performance but at the expense of complex algorithmic structure and time-consuming implementation, which limits their reusability and feasibility especially for large-scale clustering.

Another research direction concentrates on improving the local and/or global search ability of PSO for data clustering. For example, Omran et al. (2005) represented a PSO-based dynamic clustering approach called DCPSO for providing the clustering solution and the optimal clustering number simultaneously. For DCPSO, *k*-means functions as a local refining strategy, which is also adopted by this paper. Abbas et al. (2010) compared three PSO-based clustering algorithms with three traditional techniques. Experiments on nine small-sized datasets showed the superiorities of PSO-based clustering methods against three traditional counterparts. Szabo et al. (2010) proposed a modified particle swarm clustering (mPSC) method which eliminates the inertia weight and makes the velocity memoryless. The implementation of mPSC, however, is still time-consuming (Yuwono et al. 2014). Chuang et al. (2011) represented an accelerated chaotic particle swarm optimization (ACPSO) for data clustering. Empirical studies on six small-sized datasets showed better clustering ability for ACPSO by comparing it with six other clustering methods. The concept of chaos was employed by adding more randomness, in order to increase the exploration ability. Nevertheless, excessive randomness might make it difficult and even impossible to theoretically analyze the search behavior of PSO and the clustering performance of ACPSO.

More recently, a state-of-the-art version of PSO-based clustering algorithms, abbreviated as RCE, was developed by Yuwono et al. (2014). For RCE, each particle only represents a cluster centroid, and the entire population constitutes a candidate partitioning solution for the clustering problem. This representation way is more efficient than another commonly used way that each particle stands for a candidate partitioning solution, in term of both computational time and memory cost. Simulation experiments have also shown that, in most cases, the run time of the former is several orders of magnitude larger than that of the latter (Yuwono et al. 2014). Although RCE has simplified the rule of particle updating, it is still complicated to choose the *pbest* and *gbest* for each particle. It has shown in Trelea (2003) that, given enough number of iterations, all particles will converge to one equilibrium point. In other words, the attractive efforts of both *pbest* and *gbest* may make all cluster centroids (i.e., all the particles) aggregate together, which is against the fact that all cluster centroids should be separated as soon as possible.

Roughly speaking, almost all PSO-based clustering algorithms suffer from three serious issues: (1) how to enhance the computational efficiency via properly choosing the individuals' representation strategy and optimizing the implementation procedure; (2) how to properly set related control parameters for different clustering problems; and (3) how to prove the superiority from the mathematic viewpoint instead of via empirical comparative experiments. This may explain that fact that, although various PSO-based clustering methods have been studied by the academic community over the past decades, they were not widely accepted and used by the industrial and commercial institutions. For more details of the review and discussions on EA-based clustering, please refer to Abbas et al. (2010), Mohamed and Sivakumar (2011), Hruschka et al. (2009), Ahmed et al. (2013). In summary, how to design an effective PSO-based clustering technique is still an interesting yet challenging task, which will be further investigated in the following sections.

3 PSO-based clustering algorithms

In this section, a population-based algorithm framework, which combines six different PSOs with *k*-means, is proposed. These six PSO variants with different properties and performances have been studied and compared extensively in the context of continuous function optimization (e.g., Eberhart and Kennedy 1995; Shi and Eberhart 1998; Eberhart and Shi 2000; Eberhart and Shi 2001; Clerc and Kennedy 2002; Mendes et al. 2004; Ratnaweera et al. 2004; Liang et al. 2006; Kennedy 2010; Chen et al. 2013), presented as following:

1. Particle swarm optimizer with a *global* (namely, "*all*") topology and a *random* inertia weight (GPSO-RW) (Ratnaweera et al. 2004): the inertia weight W is randomly distributed in [0.5, 1], which simplifies its settings for different optimization problems.
2. Particle swarm optimizer with a *global* neighborhood structure and an inertia weight *decreasing linearly* from 0.9 to 0.4 (GPSO-WV) (Shi and Eberhart 1998): This is perhaps the most commonly used version, owing to the good balance between local and global search on many benchmark functions.
3. *Global* particle swarm optimizer with a *constriction factor* (GPSO-CF) (Eberhart and Shi 2000): the differences between it and the above two lie in the introduction of the constriction factor that prevents explosion of the particle system coupled with the mathematic derivation-based parameter configure (Clerc and Kennedy 2002).
4. *Local* particle swarm optimizer which adopts the "*von Neumann*" population topology (LPSO-VN) (Eberhart and Kennedy 1995): it has been widely argued that, in general, LPSO outperforms GPSO on most multimodal functions in term of solution quality, due to its higher probability of avoiding premature convergence resulted in by slower information flow (Eberhart and Shi 2001).
5. *Fully informed* particle swarm optimizer (FIPS) (Mendes et al. 2004): the weight-based *pbest* of all neighbors are used to update each particle' position, which reduces the influence of the best-performing one (i.e., *gbest*). Note that only the *Ring* topology is employed for FIPS in this paper, as suggested in Mendes et al. (2004).
6. *Comprehensive learning* particle swarm optimizer (CLPSO) (Liang et al. 2006): its main contribution may be that for different dimensions, each agent learns towards different neighbors in a dynamical population topology. CLPSO show good exploitation abilities on lots of multimodal functions.

To avoid repetition, it is assumed that the readers are familiar with these six PSO versions. Otherwise, please refer to the corresponding papers for more details of implementation. These PSO variants can tackle various continuous optimization problems but with different performances. Note that it is still unknown whether these PSOs can work well and show significant different performances on diverse clustering problems as on continuous benchmark functions. To further investigate such issue, a simple population-based algorithm framework (***PopbAF***) is designed, where six different PSO versions can be integrated separately into it. The detailed description is represented in the following paragraphs.

3.1 Real-coded representation strategies

For evolutionary algorithms (EAs), the representation ways of a candidate clustering solution can be simply classified into three groups, as illustrated in the following.

1. The first category adopts a direct (*integer/binary*-based) encoding strategy, which is also called the object-cluster association (Das et al. 2008). Specifically, a partitioning solution is shown via a ($N \times K$) matrix where N is the total number of data points in a dataset and K is the predefined number of clusters (see examples: Murthy and Chowdhury 1996; Chioua and Lan 2001; Hamid et al. 2012). This representation way leads easily to two serious drawbacks, i.e., prohibitively expensive storage and computation cost (especially for large–scale clustering) and redundancy (see Das et al. 2008 for further analyses).
2. The second one, which is perhaps the most widely used, only takes into account the cluster centroids as *k*-means. It encodes all cluster centers as an agent, which is stored in a real-valued vector or matrix (e.g., Chen and Fun 2012; Kao et al. 2008; Alam et al. 2008; Niknam and Amiri 2010; Cohen and Castro 2006; Tsai and Kao 2011). In other words, an agent (i.e., a particle for PSO) can represent a partitioning result. Related studies have clearly shown that the second representation strategy is better than the first one with regard to computational complexity, in particular when solving large-sized clustering problems.
3. The third (see Merwe and Engelbrecht 2003; Yuwono et al. 2014; Alam et al. 1995) does also encode only the cluster centers, which is similar to the second. However, a candidate clustering solution consists of all agents (rather than one agent), and therefore the number of agents equals to K. In general, the third way needs smaller storage spaces and less computation time as compared with others. Nevertheless, for the third way, a potential problem lies in that it may not take full use of the advantages provided by the parallel search of multiple agents, which will be in-depth illustrated in

the experiments. For any EA, the parallel search of multiple agents is extensively argued to be a crucial driven force for optimization.

Based on the following comparisons, the second is adopted in the *PopbAF*. The real-coded representation strategy for particle $\overrightarrow{P_t}$ ($t = 1, \ldots, S$, where S is the swarm size) is presented in the following vector:

$$\overrightarrow{P_t} = \left(\overrightarrow{C_1^t}, \ldots, \overrightarrow{C_K^t}\right), \text{ where } \overrightarrow{C_k^t} = \left(M_{k1}^t, \ldots, M_{kd}^t\right). \quad (1)$$

$\overrightarrow{C_k^t} = \left(M_{k1}^t, \ldots, M_{kd}^t\right)$ is a real-coded vector corresponding to a clustering centroid for particle t, and M_{kd}^t is the d-dimensional position of centroid $\overrightarrow{C_k^t}$.

3.2 Population initializations

For most EA-based clustering techniques, two initialization approaches are commonly studied and used. One is to locate all the individuals at random in the entire search space (note that the search space needs to be normalized/standardized before employing the partitioning clustering algorithms); while the other is to choose K random samples from the entire dataset as the centroids. Other advanced initialization methods can be founded in Peña et al. (1999), Bradley and Fayyad (1998), Celebi et al. (2013) and Zhang et al. (2014). These studies have attempted to investigate the effects of different initialization methods on the clustering performance. However, different conclusions were drawn from different research papers, which may be due to the fact that different datasets with arbitrary shapes and sizes are used by different papers while the performances of partitioning clustering methods depend heavily on the chosen datasets, the settings of parameters, and the choices of initial seeds.

The proposed algorithm chooses the first method to initialize the population. Note that the second initialization method will be compared with the former on some datasets in the following experiments, to show the effects of different initialization approaches on the clustering performance. For refining the quality of clustering and accumulating the convergence of population, the centroids obtained by k-means can be organized as a particle (e.g., acting as *gbest* or *lbest*) before executing the PSO algorithm. However, such trick initial method is *optional*. This refining strategy will be compared with the non-refining strategy in the following experiments to validate its influence on the convergence performance.

3.3 Fitness evaluations

In practice, different fitness functions [e.g., Davies–Bouldin index (1979), Silhouettes index (Rousseeuw 1987), just to name a few] can be used for measuring the quality of a solution (Das et al. 2008). For example, Omran et al. (2002) considered the maximization of inter-cluster distances as well as the minimization of intra-cluster distances. However, the extra introduction of control parameters in function evaluations (FEs) increases the difficulty of parameter settings and computational burden.

For partitioning clustering algorithms, one widely used metric, i.e., mean squared error (*MSE*), is used to evaluate the quality of portioning, according to the suggestion from Tsai et al. (2014). *MSE* (namely, the sum of intra-cluster distances) is calculated by

$$MSE = \sum_{j=1}^{K} \sum_{\overrightarrow{X_i} \in C_j} \left\| \overrightarrow{X_i} - \overrightarrow{C_j} \right\|^2 \quad (2)$$

where $\overrightarrow{X_i} = (X_{i1}, \ldots, X_{id})$ is the data point i ($i = 1, \ldots, N$) in the d-dimensional real space, and $\overrightarrow{C_j}$ is the centroid of cluster j, and $\|\cdot\|^2$ denotes the squared Euclidean distance between two points (note that other distance metrics can be also used here). The centroid of cluster j is updated by

$$\overrightarrow{C_j} = \sum_{i \in C_j} \overrightarrow{X_i} / \left|C_j\right| \quad (3)$$

where $\left|C_j\right|$ is the number of instances in cluster j. Note that *MSE* is only suitable to clustering problems where the number of clusters is predefined (see Chioua and Lan 2001 for detailed explanations). How to determine the optimal number of clusters is ongoing research. Please refer to Das et al. (2008), Pham et al. (2005), Milligan and Cooper (1985)) and so on for related studies. In this paper, the number of clusters, K, is supposed to be known *a prior*. In fact, multiple fitness functions can be taken into account during the optimization process, leading to multiobjective clustering (Mukhopadhyay et al. 2014a, b), which are beyond the scope of this paper.

3.4 Position updating rules of particles

For PSOs, each particle t with the velocity itself moves stochastically toward its personally historically best position ($\overrightarrow{PB_t}$) and its neighbors' best positions ($\overrightarrow{NB_t}$), until the maximum number of iterations (generations) is exceeded. For the *global* PSO version, the neighbors of a particle are defined as the entire population. For the *local* PSO version, the neighbors of a particle consist of l particles satisfying $l \ll S$. Different neighborhood topologies (e.g., *Ring* and *von Neumann*) use different l values and different ways to judge its neighbors. Learning toward *pbest* and *lbest* provides some particles the opportunities of escaping from local optima, to some extent. The velocity $\overrightarrow{V_t}$ and position

$\overrightarrow{P_t}$ adjustment rule for particle t ($t = 1, \ldots, S$) are presented in the following:

$$\overrightarrow{V_t} = k * \left(w * \overrightarrow{P_t} + c_1 * R_{t1} * \left(\overrightarrow{PB_t} - \overrightarrow{X_t} \right) + c_2 * R_{t2} * \left(\overrightarrow{NB_t} - \overrightarrow{X_t} \right) \right) \tag{4}$$

$$\overrightarrow{P_t} = \overrightarrow{P_t} + \overrightarrow{V_t} \tag{5}$$

where k is the constriction factor that prevents explosion of the particle system (Clerc and Kennedy 2002), w is the inertia weight that controls the balance between exploitation and exploration (Shi and Eberhart 1998), c_1 and c_2 are the cognitive and social learning coefficients that have important effects on the convergence rate of algorithm, and R_{t1} and R_{t2} are two separately generated number randomly distributed in the range [0,1). Then, the updating rule of $\overrightarrow{PB_t^{g+1}}$ at generation (g + 1) is illustrated as following:

$$\overrightarrow{PB_t^{g+1}} = \begin{cases} \overrightarrow{P_t^g}, & if\ f\left(\overrightarrow{PB_t^g}\right) \geq f\left(\overrightarrow{P_t^g}\right) \\ \overrightarrow{PB_t^g}, & otherwise \end{cases} \tag{6}$$

Note that six different PSO variants have some differences in terms of the definition of neighbors, detailed implementation procedure, and concrete parameter configure. Their comprehensive descriptions are referred to the original papers, to save space.

3.5 Boundary handling

Recent study (Chu et al. 2011) has shown that the boundary handling strategy has an important impact in the optimization performance of PSO when solving a complex problem, however, which is often ignored on optimizing benchmark functions. A variety of boundary handling schemes have been designed: (1) periodic model (Zhang and Xie 2003; Zhang et al. 2004); (2) absorbing model (Robinson and Samii 2004); (3) invisible model (Robinson and Samii 2004); (4) damping model (Huang and Mohan 2005); (5) reflecting model (Chu et al. 2011); (6) random model (Chu et al. 2011); and (7) zoomed model (Cao et al. 2013). In the *PopbAF*, the most commonly used scheme, viz., the *random* model, is employed to handle the particles flying outside of the search space.

Specifically, once the particle t flies outside the search bound for any dimension d, its position is reinitialized uniformly randomly in the predefined range which is limited by the upper and lower bound. Empirically, the upper and low bound for dimension d are fixed as the maximum and minimum of dataset on dimension d.

3.6 Pseudocode of the PSO-based clustering algorithm framework

In this section, six kinds of PSO-based clustering methods are unified into a simple population-based algorithmic framework, as presented in the follows:

```
Step 1 Initialize $\overrightarrow{P_t}$ (see Sects. 3.1--3.2) for each particle $t = 1, \ldots, S$, and distribute $\overrightarrow{V_t}$ uniformly randomly in the standardized search space;
Step 2 Assign each data point $\overrightarrow{X_i}$ ($i = 1, \ldots, N$) into its nearest clusters $\overrightarrow{C_j}$ ($j = 1, \ldots, K$) for $t = 1, \ldots, S$;
Step 3 Update centroids $\overrightarrow{C_j}$ of all clusters for $t = 1, \ldots, S$ by Eq. (3) and Lloyd's $k$-means (optional);
Step 4 Calculate all particles' fitness by Eq. (2), update $\overrightarrow{PB_t}$ by Eq. (6), and update $\overrightarrow{LB_t}$ according to the corresponding topology structure;
Step 5 Repeat step 6 to step 8:
    Step 6 Update $\overrightarrow{P_t}$ and $\overrightarrow{V_t}$ for $t = 1, \ldots, S$ by Eq. (4) and (5), handle particles flying outside the search space using random model (see Sect. 3.5);
    Step 7 Refine all the centers provided by all the particles using Lloyd's $k$-means algorithm;
    Steo.8 Update all particles' fitness by Eq. (2), update $\overrightarrow{PB_t}$ by Eq. (6), and update $\overrightarrow{LB_t}$ according to the corresponding topology structure,
Step 9 Until some stopping conditions (e.g., convergence of $MSE$, no changes of clustering indexes during two successive iterations, or maximum number of iterations) is met.
Step 10 Output the cluster centroids, clustering indexes, and $MSE$.
```

Note that, in this paper, only the maximum allowable number of iterations (*max*Iter) is set as the stopping condition for fair comparisons among different PSO versions.

4 Experimental studies

To evaluate the effectiveness and efficiency of the proposed algorithm, 12 datasets and 6 benchmark algorithms are chosen for the comparative experiments. These 12 datasets consist of 3 synthetic datasets and 9 real-life datasets. Three synthetic datasets, which are abbreviated as SD1, SD2, and SD3, are generated following the suggestions from (Niu et al. 2013; Laszlo and Mukherjee 2007). All they have distinct data structures and regular distribution rules (see Niu et al. 2013 for graphical descriptions). It appears to be hard to claim that a clustering technique that performs poor on artificial well-behaved datasets can be able to success on real-life datasets with less well-defined distribution rules (Milligan and Cooper 1985). Therefore, the efficiency of the clustering algorithm is assessed firstly on these artificial datasets. Nine real-world datasets have been widely used in the data mining community and reported comprehensively in the famous UCI machine learning repository.[1] Short descriptions of these datasets are summarized in Table 1, where N and D represent the number of instances and attributes, respectively. For further information of these datasets, please refer to the corresponding papers (Tzortzis and Likas 2014; Niu et al. 2013; Laszlo and Mukherjee 2007) or the openly available website (see footnote 1). Note that almost all datasets (except the Coil2 dataset) have been standardized for eliminating the discrepancy among different scales of different attributes.

The experiments are conducted using the Matlab computation platform on a single personal computer with the Win8 operating system (2.5 GHz Intel Core i5-3210 M CPU and 2 GB RAM). For comprehensive comparisons, the experiments choose six clustering algorithms as the baselines, as presented in the following.

1. Agglomerative clustering algorithm with single link[2]: it is abbreviated as ACA-SL in this paper. The details of its Matlab implementation are presented in the Statistics Toolbox (see footnote 3).[3]
2. Agglomerative clustering algorithm with complete link[4]: it is named as ACA-CL in this paper. Its Matlab implementation can be also found in footnote 3.
3. Agglomerative clustering algorithm with average link[5]: it is called as ACA-AL in this paper. For the Matlab implementation, please refer to footnote 3.
4. Lloyd's *k*-means (Lloyd 1982): It adopts the *batch* scheme to iteratively and separately update the centroids of clusters and clustering memberships of data points. The Matlab code can be seen in Lloyd[6].
5. MacQueen's *k*-means (MacQueen 1967): the clustering memberships and centroids are iteratively updated in the *online* scheme. For the Matlab implementation, please refer to MacQueen[7].
6. Particle swarm clustering method using variants of rapid centroid estimation (Yuwono et al. 2014): it is abbreviated as PSC-RCE in this paper. The Matlab code is openly available at http://www.mathworks.com/matlabcentral/fileexchange/38107-rapid-centroid-estimation via the Internet, thanks to the authors' courtesy.

Note that the parameter settings of all involved algorithms follow the suggestions of the original papers and/or the most commonly used configurations. Each algorithm (except three ACAs) performs on each dataset for 30 independent runs. Note that the mean and worst values are recorded on each dataset for all algorithms. The worst case can provide an upper bound on the performance of a clustering technique, which is always expected to be minimized. To visualize the worst-case performance of different algorithms, the final clustering solutions obtained on two synthetic datasets (SD1 and SD2) are depicted in Figs. 1 and 2. In each figure, the sub-figure in the top left corner represents the actual partitioning, while the remaining sub-figures are the partitioning solutions obtained by two *k*-means and one PSO-based clustering algorithm, respectively. Note that the final partitional results achieved by all PSO-based clustering methods are similar, and therefore only one is drawn for saving space.

For the dataset SD1, there exists a distinct clustering pattern, where 1000 data points are distributed regularly and can be clearly divided into five groups, as seen from Fig. 1. However, for Lloyd's *k*-means algorithm, not all runs (only 16 out of 30 runs) can find the ideal partitioning. It can be seen in the top right corner of Fig. 1 that the cluster 3 and cluster 5 should be merged into one cluster while the cluster 1 coupled with cluster 2 compresses excessive data points. The similar case is also found for MacQueen's *k*-means algorithm. On contrary, the optimal partitioning results are obtained by the proposed algorithm on all runs.

As we can see from Fig. 2, both Lloyd's *k*-means and MacQueen's *k*-means obtain the wrong partitioning solutions on the dataset SD2 at the worst case. This is due to the fact that the performance of *k*-means heavily depends on the initial seed coupled with the order of data points (Ahmed et al. 2013). For all proposed PSO-based algorithms, however, the optimal position for the centroids can be properly detected for all runs. The reason that the population-based methods outperform individual-based counterparts may lie in that for the former, each individual

[1] http://archive.ics.uci.edu/ml/.

[2] http://en.wikipedia.org/wiki/Single-linkage_clustering.

[3] http://www.mathworks.cn/cn/help/stats/hierarchical-clustering.html.

[4] http://en.wikipedia.org/wiki/Single-linkage_clustering.

[5] http://nlp.stanford.edu/IR-book/completelink.html.

[6] http://lear.inrialpes.fr/~verbeek/software.

[7] http://www.mathworks.cn/cn/help/stats/kmeans.html.

Table 1 Brief descriptions of 12 datasets chosen in the experiements

	SD1	SD2	SD3	Iris	Wine	Coil2
N	1000	800	10,000	150	178	216
D	3	2	2	4	13	1000
Type	Synthetic	Synthetic	Synthetic	Real-life	Real-life	Real-life
	Breast cancer	**German credit**	**Optdigits**	**Musk**	**Magic04**	**Road network**
N	683	1000	5620	7074	19,020	434,874
D	9	20	64	166	10	4
Type	Real-life	Real-life	Real-life	Real-life	Real-life	Real-life

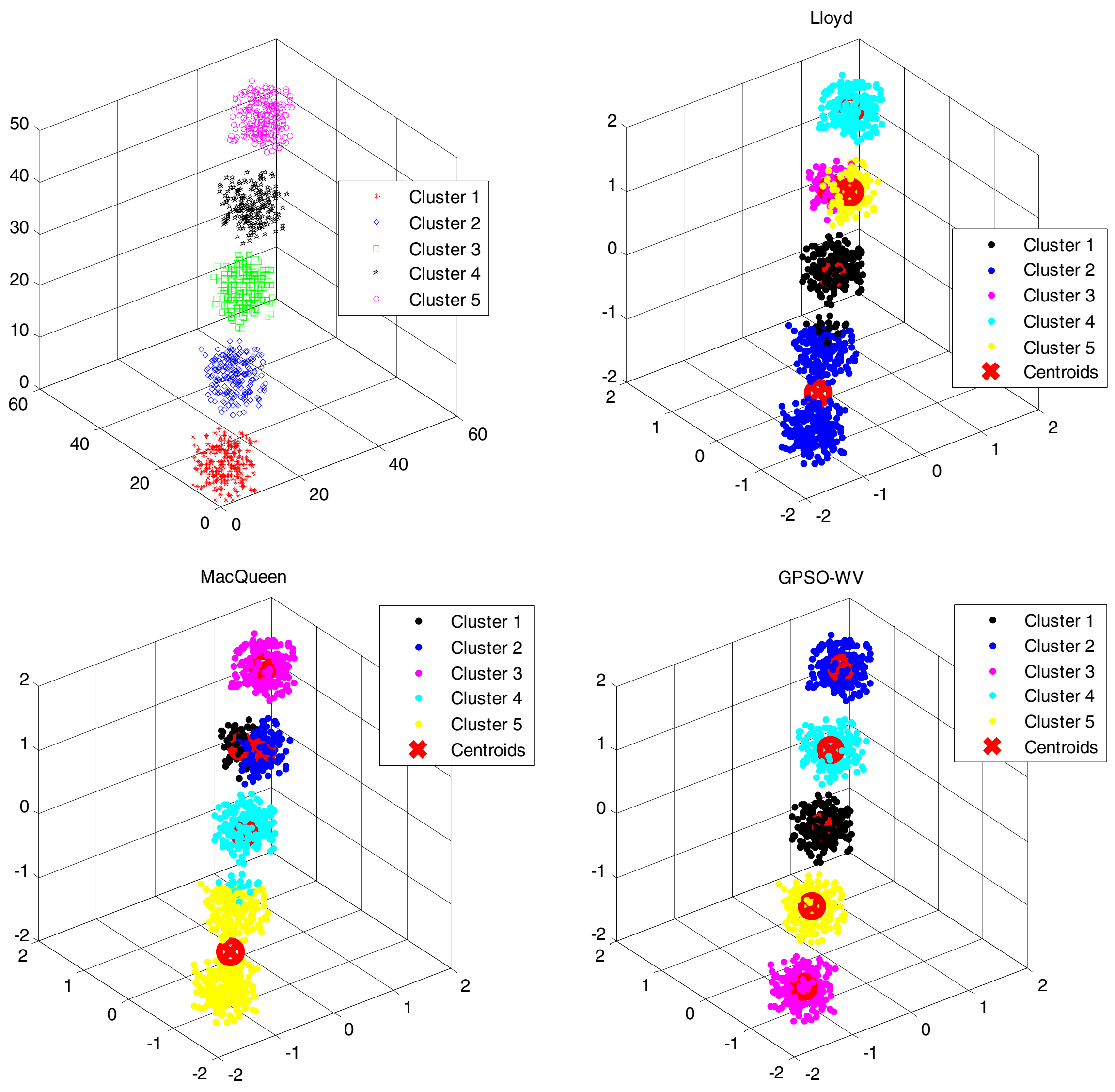

Fig. 1 Comparisons of the worst-case final clustering solutions on the SD1 dataset

can not only make full use of the inherent information obtained by it, but also the external information collected by other individuals. On the dataset SD3, MacQueen's *k*-means obtains the worst result in terms of *MSE*, following by the Lloyd's *k*-means, as compared with all the proposed algorithms.

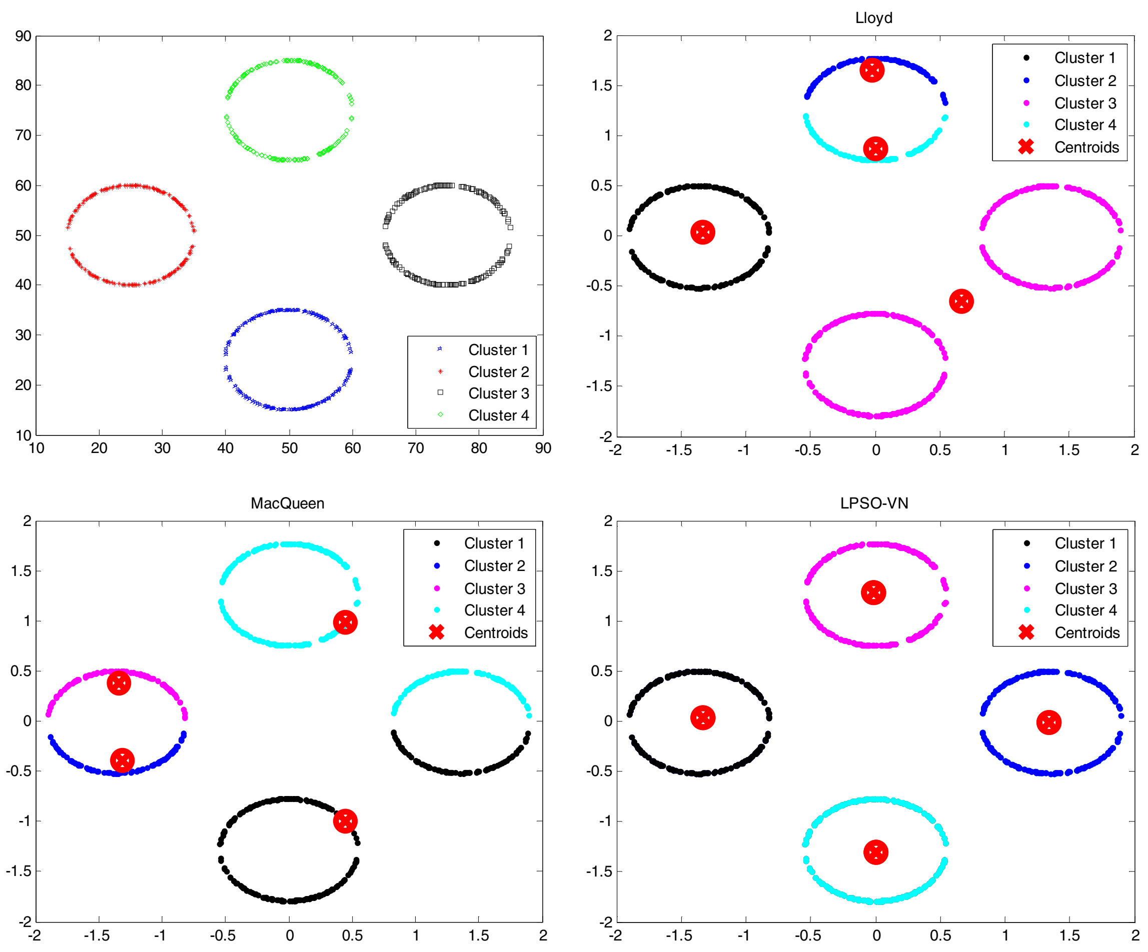

Fig. 2 Comparisons of the worst-case final partitioning solutions on the SD2 dataset

In the following subsections, the comparative experiments are divided into two parts. The experiments I focus mainly on the comparisons between the hierarchical and partitioning clustering methods, while in the experiments II the comparisons between different kinds of PSO-based clustering are concentrated on.

4.1 Comparative experiments-I

For the comparative experiment I, all results obtained by all algorithms on each dataset are listed in Table 2, where Lloyd and MQ represent Lloyd's ad MacQueen's *k*-means. On the Iris and Wine dataset which are probably the most cited datasets in the data mining field, six PSO-based clustering techniques are the best performers with the same performance, followed by MacQueen' *k*-means. For the Coil2 dataset, different PSO-based methods show slightly (but *not significantly*) performance differences, where GPSO-CF achieves the best solution and GPSO-RW ranks the second. However, the worst cases are encountered by all them on some runs. Note that, for the Coil2 dataset, only ACA-CL obtains the worst performance as compared with all its competitors. On the Breast Cancer dataset, all partitioning-based clustering techniques significantly outperform three agglomerative methods. Both the Lloyd's and MacQueen's *k*-means lack robustness (i.e., a small variance) for the German Credit dataset while the proposed algorithms are not sensitive to the initial seeds.

Some abnormalities (where the proposed algorithms achieve the worst clustering performance) can be observed on the Optdigits dataset. By carefully investigations, we found such exception put down to the difference between population initialization strategies (see Sect. 3.2). In effect, if the proposed algorithm adopts the second initialization method, the best clustering performance (i.e., 224,860.82827) can be attained with 18 % improvement. For the Musk dataset, the proposed algorithms are significantly better than all rivals in terms of *MSE*. Note that since three ACAs are computationally prohibitively unfeasible for large-scale datasets (here, Magic04 and Road Network can be regarded as large-sized

Table 2 Comparative results of 11 algorithms on 9 real-life datasets

Algorithm	Dataset								
	Iris			Wine			Coil2		
	Mean	Worst	Std	Mean	Worst	Std	Mean	Worst	Std
ACA-SL	213.65	213.65	0.00	2198.93	2198.93	0.00	154.03	154.03	0.00
ACA-CL	152.86	152.86	0.00	1475.11	1475.11	0.00	174.91	174.91	0.00
ACA-AL	196.86	196.86	0.00	2199.65	2199.65	0.00	154.03	154.03	0.00
Lloyd	154.02	220.87	27.71	1305.54	1650.63	100	155.00	158.00	1.47
MQ	147.57	195.96	19.78	1281.00	1578.26	56.1	154.53	160.05	1.74
GPSO-RW	138.89	138.89	0.00	1270.75	1270.75	0.00	154.02	155.24	0.41
GPSO-WV	138.89	138.89	0.00	1270.75	1270.75	0.00	154.05	155.24	0.42
GPSO-CF	138.89	138.89	0.00	1270.75	1270.75	0.00	153.99	155.24	0.42
LPSO-VN	138.89	138.89	0.00	1270.75	1270.75	0.00	154.08	155.24	0.42
FIPS	138.89	138.89	0.00	1270.75	1270.75	0.00	154.08	155.24	0.42
CLPSO	138.89	138.89	0.00	1270.75	1270.75	0.00	154.08	155.24	0.42
Algorithm	Dataset								
	Breast cancer			German credit			Optdigits		
	Mean	Worst	Std	Mean	Worst	Std	Mean	Worst	Std
ACA-SL	6094.20	6094.20	0.00	22,836.1	22,836.1	0.00	317,533	317,533	0.00
ACA-CL	3378.81	3378.81	0.00	22,836.1	22,836.1	0.00	305,714	305,714	0.00
ACA-AL	4987.70	4987.70	0.00	22,836.1	22,836.1	0.00	309,332	309,332	0.00
Lloyd	2724.42	2724.47	0.09	22,400.2	24,000.0	335	230,000	236,000	2400
MQ	2724.16	2724.16	0.00	22,298.8	22,532.1	105	228,754	235,327	2519
GPSO-RW	2724.16	2724.16	0.00	22,171.1	22,171.1	0.00	288,987	304,216	6989
GPSO-WV	2724.16	2724.16	0.00	22,171.1	22,171.1	0.00	289,006	303,669	7107
GPSO-CF	2724.16	2724.16	0.00	22,171.1	22,171.1	0.00	289,439	302,969	7107
LPSO-VN	2724.16	2724.16	0.00	22,171.1	22,171.1	0.00	289,877	304,216	7111
FIPS	2724.16	2724.16	0.00	22,171.1	22,171.1	0.00	289,851	304,216	7415
CLPSO	2724.16	2724.16	0.00	22,171.1	22,171.1	0.00	289,851	304,216	7415
Algorithm	Dataset								
	Musk			Magic04			Road network		
	Mean	Worst	Std	Mean	Worst	Std	Mean	Worst	Std
ACA-SL	1,171,570	1,171,570	0.00	–	–	–	–	–	–
ACA-CL	1,152,430	1,152,430	0.00	–	–	–	–	–	–
ACA-AL	1,167,550	1,167,550	0.00	–	–	–	–	–	–
Lloyd	901,000	982,000	41,900	137,000	137,000	0.00	1.003e6	1.012e6	1e4
MQ	899,887	981,513	42,632	136,919	136,919	0.00	1.003e6	1.012e6	1e4
GPSO-RW	869,655	869,655	0.00	136,919	136,919	0.00	9.939e5	9.940e5	1e3
GPSO-WV	869,655	869,655	0.00	136,919	136,919	0.00	9.939e5	9.940e5	1e3
GPSO-CF	869,655	869,655	0.00	136,919	136,919	0.00	9.939e5	9.940e5	1e3
LPSO-VN	869,655	869,655	0.00	136,919	136,919	0.00	9.938e5	9.940e5	1e3
FIPS	869,655	869,655	0.00	136,919	136,919	0.00	9.939e5	9.940e5	1e3
CLPSO	869,655	869,655	0.00	136,919	136,919	0.00	9.939e5	9.940e5	1e3

Mean, worst and Std denote the mean, worst and standard deviations of MSE obtained in 30 runs

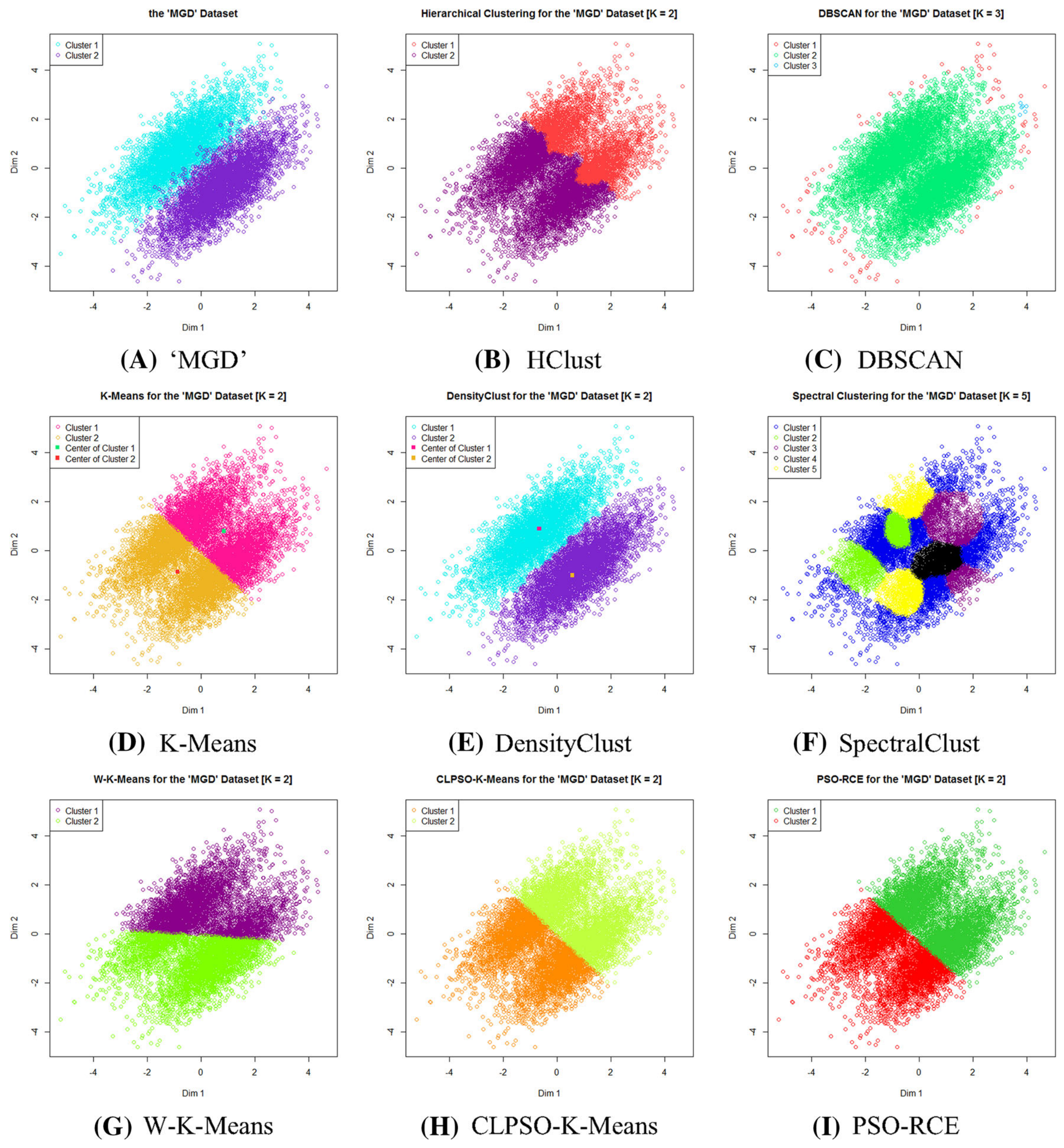

Fig. 3 Final clustering results on 'MGD' for different clustering algorithms (*different colors mark* different cluster assignments for each sub-figure)

datasets), the results obtained on them by ACAs are not provided in Table 2. For large-scale datasets, MacQueen's *k*-means shows better refining ability than Lloyd's *k*-means.

In summary, all the proposed algorithms show ***better*** **(at least as good as)** ***and more robust*** clustering performance than other five competitors on most cases. However, PSO-based clustering algorithms require more computation recourses. But we believe that such price (i.e., extra computational burden) is worthy of being paid in many real-life cases (e.g., clustering-based customer segmentation). Furthermore, six different PSO-based clustering techniques ***do not show significant differences*** in the clustering performance. This means that, in the context of PSO, these optimization strategies (e.g., different population topologies and parameter settings) which can perform well on continuous benchmark functions ***do not necessarily*** work well on the clustering problems. The reason behind it may be that the clustering optimization problems have ***unique fitness landscape structures***, which need to be investigated further, while such unique fitness landscape structures may not be fully represented by most benchmark functions (Fig. 3).

4.2 Comparative experiments-II

The well-designed 2-dimensional ‘MGD’ dataset, as presented in Fig. 1, consists of two separated clusters with mixed *Gaussian* distributions. In effect, mixed normal distributions have been extensively accepted as the base for generating artificial data sets in the clustering literature (e.g., Huang et al. 2005), perhaps as *Gaussian* distribution is the most commonly used probability distribution in both academic research and real-life datasets. On contrary, uniform distribution is chosen as the base for creating noise variables.

As shown in Fig. 1, only the *DensityClust* method (sub-fig. E) finds the almost same clustering assignments as the

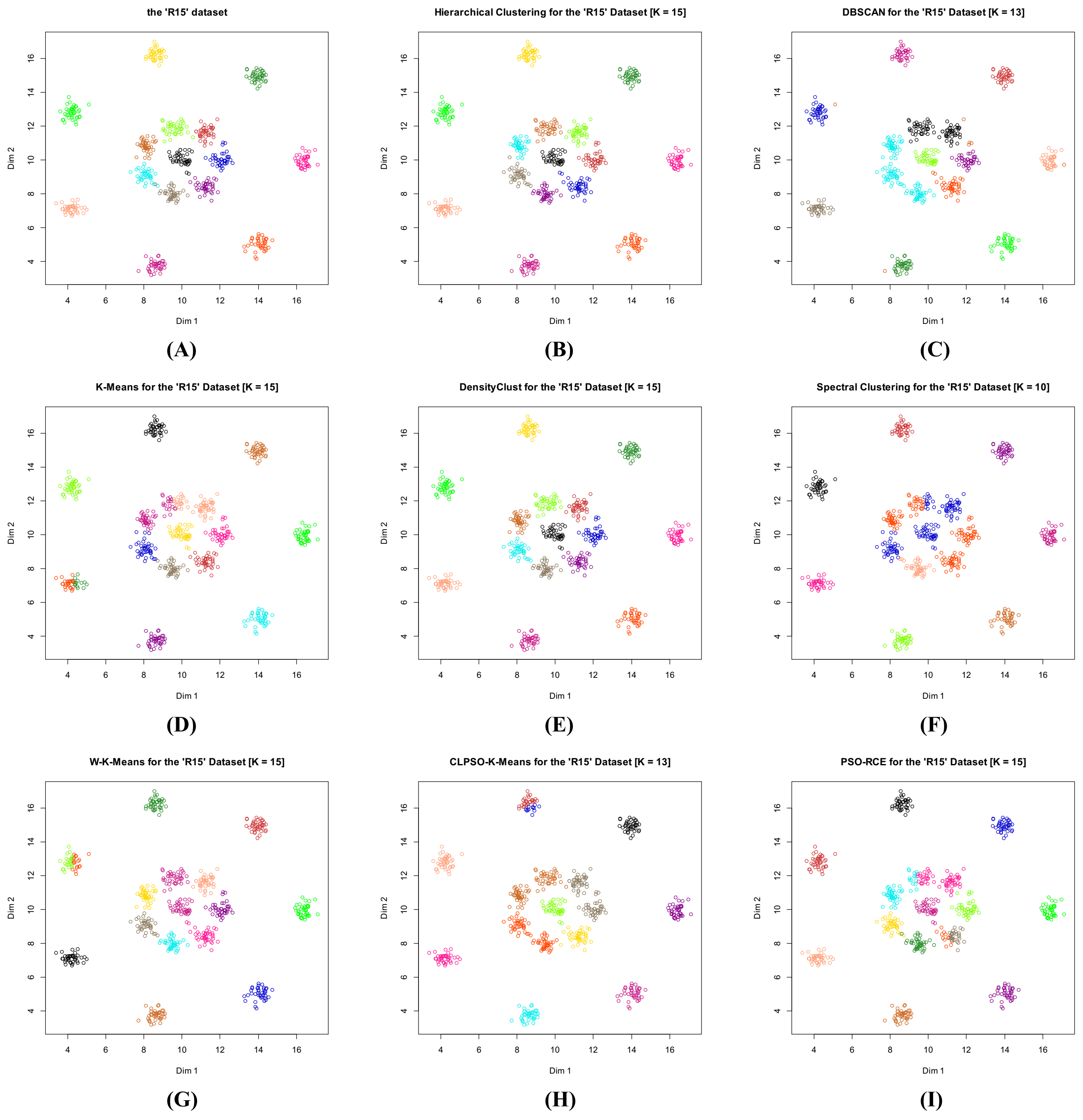

Fig. 4 Final clustering results on the ‘R15’ dataset of different algorithms (*different colors mark* different cluster assignments in each sub-figure). **a** ‘R15’ Dataset (2-D), **b** hierarchical, **c** DBSCAN, **d** K-means, **e** DensityClust, **f** spectral clustering, **g** W–K-means, **h** CLPSO-K-means, **i** PSO-RCE

original dataset (sub-fig. A), with two centroids located in two highest-density areas. For *K-Means*, *W-K-Means* and *CLPSO-K-Means*, three similar linearly-separated decision boundaries are obtained, which are presented in sub-fig. D, G and H respectively. By visual inspection of sub-fig. D, we can see that two predicted centroids are sited in regions with low density, which seems to violate the intuitive definition of centroids. Owing to more complex optimization space, *W-K-Means* (sub-fig. G) cannot correctly update the weights of two variables (consequently, 0.4603 vs. 0.5364), resulting in unbalanced clustering assignments. In fact, these two dimensionalities have the exactly same weight, perhaps which is the most common clustering scenario in the literature. In general, *K-Means-type* clustering algorithms, coupled with *PSO-RCE* (sub-fig. I), excel in the detection of sphere-shaped clustering structures, rather than eclipse-shaped ones. For the 'MGD' dataset, the *HClust* method (sub-fig. B) gives the similar clustering result as *K-Means*, but with slightly different non-linear segmentation boundary. Note that, for all the above clustering techniques, it is assumed that the number of clusters (i.e., *k*) can be specified in advance.

For both the *DBSCAN* (sub-fig. C) and *SpectralClust* (sub-fig. F) method, however, it is difficult to find the right number of clusters and good clustering solutions, as they cannot directly put *k* as the system parameter. Although the above fact appears to give unfair advantages to *K-Means-type* clustering algorithms, absolutely fair comparisons for different types of clustering methods with different optimization mechanisms seems to be unrealistic and even impossible in many situations, since they may have totally different understandings for the concept of clustering and optimization procedure for different objective functions. Fortunately, at least for the 'MGD' dataset, the *DensityClust* methods can find both the correct number of clusters and clustering solution simultaneously, though at the cost of computational complexity (i.e., $\theta(n^2)$).

The above conclusion can be also found in Fig. 4.

4.3 Comparative experiments-III

In the experiment II, one state-of-the-art version of PSO-based clustering, viz., PSC-RCE, is chosen as the benchmark algorithm. The comparative results are summarized in Table 3. For clarity of comparisons, in Table 3, the best *means* and *worst* for each dataset are marked in boldface. Note that since six PSO-based clustering algorithms achieve the same or very similar clustering performance on these nine chosen dataset, only one version is selected to save space.

As we can see from Table 3, the proposed algorithm outperforms PSC-RCE in most cases except on the Optdigits and Magic04 dataset. However, the proposed algorithm shows more robust and the same performance on the Optdigits and Magic04 dataset as compared with the competitor, respectively.

Table 3 Comparative results of PSO-based algorithms on 9 real-life datasets

Algorithm	Dataset								
	Iris			Wine			Coil2		
	Mean	Worst	Std	Mean	Worst	Std	Mean	Worst	Std
PSC-RCE	142.90	148.61	2.33	1321.46	1358.12	12.7	156.48	157.21	0.45
CLPSO	**138.89**	**138.89**	**0.00**	**1270.75**	**1270.75**	**0.00**	**154.08**	**155.24**	**0.42**
Algorithm	Dataset								
	Breast cancer			German credit			Optdigits		
	Mean	Worst	Std	Mean	Worst	Std	Mean	Worst	Std
PSC-RCE	2798.83	2841.83	19.02	23,012.1	23,485.3	194	**278,013**	319,329	13,392
CLPSO	**2724.16**	**2724.16**	**0.00**	**22,171.1**	**22,171.1**	**0.00**	289,851	**304,216**	**7415**
Algorithm	Dataset								
	Musk			Magic04			Road network		
	Mean	Worst	Std	Mean	Worst	Std	Mean	Worst	Std
PSC-RCE	890,520	909,881	11,481	**136,919**	**136,919**	**0.00**	1.024e6	1.026e6	2e3
CLPSO	**869,655**	**869,655**	**0.00**	**136,919**	**136,919**	**0.00**	9.939e5	9.940e5	1e3

Where mean, worst and Std denote the mean, worst and standard deviations of MSE obtained in 30 runs

5 Conclusion

In this paper, six different PSO versions have been hybridized with *k*-means to solve the data clustering problems. In terms of *MSE*, the proposed algorithms show better performances than six competitors on 12 datasets in most case. This *should be put down to* their parallel search abilities obtained via multiple agents. Interesting, no significant differences on the clustering performances among these PSOs have been observed. Note that the *k-means-type* optimization problems have their own unique fitness landscapes, which may be not fully represented by continuous benchmark functions. Hence, some commonly used optimization strategies (e.g., four types of population topologies and three kinds of parameter setting methods in this paper) can work well on many benchmark functions while fail on the *k-means-type* optimization problems. Further, the population initialization approaches may have a critical impact on the clustering performance.

In this paper, we only focus on the study of hybridizing the PSO algorithms with K-means algorithm. However, other evolutionary computation techniques, such as artificial bee colony optimization, bacterial foraging optimization, and differential evolution, can also be used to hybridize with K-means algorithm. In the future work, more other recent evolutionary computation techniques or some other variants of PSO will be combined with K-means algorithm to solve the data clustering problems

Acknowledgments This work is partially supported by The National Natural Science Foundation of China (Grants Nos. 71571120, 71001072, 71271140, 71471158, 71461027, 61472257).

References

Abbas A, Fakhri K, Mohamed SK (2010) Flocking based approach for data clustering. Nat Comput 9(3):767–794

Ahmed AAE, Rodrigo AC, Stan M (2013) A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. Artif Intell Rev 44(1):23–45

Alam S, Dobbie G, Riddle P, Naeem MA (1995) Particle swarm optimization based hierarchical agglomerative clustering. In: 2010 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology (WI-IAT), Toronto, ON, pp 64–68

Alam S, Dobbie G, Riddle P (2008) An evolutionary particle swarm optimization algorithm for data clustering. In: Proceedings of IEEE swarm intelligence symposium, pp 1–6

Average Link. http://nlp.stanford.edu/IR-book/completelink.html. Visited: 2014-09-16

Bradley PS, Fayyad UM (1998) Refining initial points for k-means clustering. Microsoft Res http://research.microsoft.com/apps/pubs/default.aspx?id=68490, MSR-TR-98-36

Cao H et al (2013) Cluster analysis based on attractor particle swarm optimization with boundary zoomed for working conditions classification of power plant pulverizing system. Neurocomputing 117:54–63

Celebi ME, Kingravi HA, Vela PA (2013) A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst Appl 40(1):200–210

Chen CY, Fun Y (2012) Particle swarm optimization algorithm and its application to clustering analysis. In: Proceedings of 17th conference on electrical power distribution networks (EPDC), pp 789–794

Chen WN, Zhang J et al (2013) Particle swarm optimization with an aging leader and challengers. IEEE Trans Evol Comput 17(2):241–258

Chioua YC, Lan LW (2001) Genetic clustering algorithms. Eur J Oper Res 135(2):413–427

Chu W, Gao XG, Sorooshian S (2011) Handling boundary constraints for particle swarm optimization in high-dimensional search space. Inf Sci 128(20):4569–4581

Chuang LY, Hsiao CJ, Yang CH (2011) Chaotic particle swarm optimization for data clustering. Expert Syst Appl 38(12):14555–14563

Clerc M, Kennedy J (2002) The particle swarm–explosion, stability, and convergence in a multi-dimensional complex space. IEEE Trans Evol Comput 6(1):58–73

Cohen SCM, Castro LN (2006) Data clustering with particle swarms. In: Proceedings of the IEEE congress on evolutionary computation, Vancouver, BC, pp 1792–1798

Complete Link. http://en.wikipedia.org/wiki/Single-linkage_clustering. Visited: 2014-09-16

Das S, Abraham A, Konar A (2008) Automatic clustering using an improved differential evolution algorithm. IEEE Trans Syst Man Cybern Part A Syst Hum 38(1):218–237

Davies D, Bouldin D (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1(2):224–227

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth symposium on micro machine and human science, Piscataway, NJ, pp 39–43

Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the congress on evolutionary computation, San Diego, CA, pp 84–88

Eberhart RC, Shi Y (2001) Particle swarm optimization: developments, applications and resources. In: Proceedings of the congress on evolutionary computation, Seoul, pp 81–86

Flynn PJ, Murty MN, Jain AK (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323

Hamid M, Saeed J, Seyed MHH (2012) Dynamic clustering using combinatorial particle swarm. Appl Intell 38(3):289–314

Handl J, Knowles J, Dorigo M (2006) Ant-based clustering and topographic mapping. Artif Life 12(1):35–62

Hruschka ER, Campello RJGB, Freitas AA, Carvalho ACPLF (2009) A survey of evolutionary algorithms for clustering. IEEE Trans Syst Man Cybern Part C Appl Rev 39(2):133–155

Huang T, Mohan AS (2005) A hybrid boundary condition for robust particle swarm optimization. IEEE Trans Antennas Wirel Propag Lett. http://epress.lib.uts.edu.au/research/bitstream/handle/10453/5871/2005003730.pdf?sequence=3

Huang JZ, Ng MK, Rong H, Li Z (2005) Automated variable weighting in k-means type clustering. IEEE Trans Pattern Anal Mach Intell 27:657–668

Hierarchical Clustering. http://www.mathworks.cn/cn/help/stats/hierarchical-clustering.html. Visited: 2014-09-16

Jain AK (2010) Data clustering: 50 years beyond K-means. Pattern Recogn Lett 31(8):651–666

Kao YT, Zahara E, Kao IW (2008) A hybridized approach to data clustering. Expert Syst Appl 34(3):1754–1762

Karaboga D, Ozturk C (2011) A novel clustering approach-artificial bee colony (ABC) algorithm. Appl Soft Comput 11(1):652–657

Kennedy J (2010) Particle swarm optimization. In: Encyclopedia of machine learning, Springer, pp 760–766

Laszlo M, Mukherjee S (2007) A genetic algorithm that exchanges neighboring centers for k-means clustering. Pattern Recogn Lett 28(16):2359–2366

Lee CY, Antonsson EK (2000) Dynamic partitional clustering using evolution strategies. In: 26th Annual conference of the IEEE industrial electronics society, Nagoya, pp 2716–2721

Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295

Lloyd SP (1982) Least squares quantization in PCM. IEEE Trans Inf Theory 28(2):129–137

Lloyd' *k*-means Matlab Code. http://lear.inrialpes.fr/~verbeek/software. Visited: 2014-09-16

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematics statistics and probability, vol 1, pp 281–296

MacQueen's *k*-means. http://www.mathworks.cn/cn/help/stats/kmeans.html. Visited: 2014-09-16

Mendes B, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evol Comput 8(3):204–210

Merwe DW, Engelbrecht AP (2003) Data clustering using particle swarm optimization. In: 2003 Congress on evolutionary computation (CEC 2003), vol 1, pp 215–220

Milligan GW, Cooper MC (1985) An examination of procedures for determining the number of clusters in a data set. Psychometrika 50(2):159–179

Mohamed JAH, Sivakumar R (2011) A survey: hybrid evolutionary algorithms for clustering analysis. Artif Intell Rev 36(3):179–204

Mukhopadhyay A, Maulik U, Bandyopadhyay S, Coello CAA (2014a) Survey of multiobjective evolutionary algorithms for data mining: part I. IEEE Trans Evol Comput 18(1):4–19

Mukhopadhyay A, Maulik U, Bandyopadhyay S, Coello CAA (2014b) Survey of multiobjective evolutionary algorithms for data mining: part II. IEEE Trans Evol Comput 18(1):20–35

Murthy CA, Chowdhury N (1996) In search of optimal clusters using genetic algorithms. Pattern Recogn Lett 17(8):825–832

Niknam T, Amiri B (2010) An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis. Appl Soft Comput 10(1):183–197

Niu B, Duan QQ, Liang J (2013) Hybrid bacterial foraging algorithm for data clustering. Lect Notes Comput Sci (IDEAL) 8206:577–584

Omran MGH, Salman AA, Engelbrecht AP (2002) Image classification using particle swarm optimization. In: Proceedings of the Asia-Pacific conference on simulated evolution and learning, pp 370–374

Omran MGH, Salman A, Engelbrecht AP (2005) Dynamic clustering using particle swarm optimization with application in image segmentation. Pattern Anal Appl 8(4):332–344

Peña JM, Lozano JA, Larrañaga P (1999) An empirical comparison of four initialization methods for the k-means algorithm. Pattern Recogn Lett 20(10):1027–1040

Pham DT, Dimov SS, Nguyen CD (2005) Selection of K in K-means clustering. http://www.ee.columbia.edu/~dpwe/papers/PhamDN05-kmeans.pdf

PSC-RCE. http://www.mathworks.com/matlabcentral/fileexchange/38107-rapid-centroid-estimation] Matlab Code. Visited: 2014-09-16

Radha T, Millie P, Ajith A, Pascal B (2011) Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl Math Comput 217(12):5208–5226

Ratnaweera A, Halgamuge S, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255

Robinson J, Samii YR (2004) Particle swarm optimization in electromagnetics. IEEE Trans Antennas Propag 52(2):397–407

Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65

Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of IEEE congress on evolutionary computation, Anchorage, AK, pp 69–73

Szabo A, Prior AKF, Castro LN (2010) The proposal of a velocity memoryless clustering swarm. In: Proceedings of IEEE congress on evolutionary computation, pp 1–5

Single Link. http://en.wikipedia.org/wiki/Single-linkage_clustering. Visited: 2014-09-16

Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85(3):317–325

Tsai CY, Kao IW (2011) Particle swarm optimization with selective particle regeneration for data clustering. Expert Syst Appl 38(6):6565–6576

Tsai CW, Huang WK, Yang CS, Chiang MC (2014) A fast particle swarm optimization for clustering. Soft Comput 19(2):321–338

Tzortzis G, Likas A (2014) The minmax *k*-means clustering algorithm. Pattern Recogn 47(7):2505–2516

UCI Repository. http://archive.ics.uci.edu/ml/. Visited: 2014-09-16

Yuwono M, Su SW, Moulton BD, Nguyen HT (2014) Data clustering using variants of rapid centroid estimation. IEEE Trans Evol Comput 18(3):366–377

Zhang WJ, Xie XF (2003) DEPSO: hybrid particle swarm with differential evolution operator. In: IEEE International conference on systems, man and cybernetics, pp 3816–3821

Zhang WJ, Xie XF, Bi DC (2004) Handling boundary constraints by PSO in periodic search space. In: Proceedings of the congress on evolutionary computation, pp 2307–2311

Zhang H, Yang ZR, Oja E (2014) Improving cluster analysis by co-initializations. Pattern Recogn Lett 45(1):71–77