Appl Intell (2013) 38:289–314
DOI 10.1007/s10489-012-0373-9

Dynamic clustering using combinatorial particle swarm optimization

Hamid Masoud · Saeed Jalili ·
Seyed Mohammad Hossein Hasheminejad

Published online: 5 September 2012

Abstract Combinatorial Particle Swarm Optimization (CPSO) is a relatively recent technique for solving combinatorial optimization problems. CPSO has been used in different applications, e.g., partitional clustering and project scheduling problems, and it has shown a very good performance. In partitional clustering problem, CPSO needs to determine the number of clusters in advance. However, in many clustering problems, the correct number of clusters is unknown, and it is usually impossible to estimate. In this paper, an improved version, called CPSOII, is proposed as a dynamic clustering algorithm, which automatically finds the best number of clusters and simultaneously categorizes data objects. CPSOII uses a renumbering procedure as a preprocessing step and several extended PSO operators to increase population diversity and remove redundant particles. Using the renumbering procedure increases the diversity of population, speed of convergence and quality of solutions. For performance evaluation, we have examined CPSOII using both artificial and real data. Experimental results show that CPSOII is very effective, robust and can solve clustering problems successfully with both known and unknown number of clusters. Comparing the obtained results from CPSOII with CPSO and other clustering techniques such as KCPSO, CGA and K-means reveals that CPSOII yields promising results. For example, it improves 9.26 % of the value of DBI criterion for Hepato data set.

Keywords Combinatorial particle swarm optimization · Combinatorial optimization problems · Partitional clustering · Dynamic clustering

H. Masoud · S. Jalili (✉) · S.M.H. Hasheminejad
SCS lab, Computer Engineering Department, Electrical and Computer Engineering Faculty, Tarbiat Modares University, Tehran, Iran
e-mail: sjalili@modares.ac.ir

H. Masoud
e-mail: h.masoud@modares.ac.ir

S.M.H. Hasheminejad
e-mail: smh.hasheminejad@modares.ac.ir

1 Introduction

Clustering is one of the most popular techniques in data mining, where the goal is to partition a set of unlabeled data objects into a number of groups of similar objects. Each group, called a cluster, includes objects that are similar to objects within the cluster and are different from those in other clusters. Clustering techniques have been used in many different applications, such as machine learning, image segmentation, web mining, bioinformatics and economics.

So far, many clustering techniques have been proposed [1]. In general, these techniques can be classified basically into two main categories: hierarchical and non-hierarchical (or partitional) clustering techniques [2].

Hierarchical clustering techniques organize data into a hierarchical structure and do not need to determine the number of clusters in advance. These techniques are mainly classified as agglomerative and divisive techniques [3]. An agglomerative clustering technique is a "bottom up" approach which treats each data object as a singleton cluster in the beginning and then recursively merges two or more of the most appropriate clusters to form a larger cluster until an appropriate clustering result emerges. Divisive clustering is a "top down" approach and proceeds in an opposite way. In the beginning, all data objects belong to a cluster and are successively divided into two smaller clusters until an appropriate clustering result is obtained.

On the other hand, partitional clustering techniques attempt to directly partition data objects into a set of disjoint clusters [4]. Partitional clustering is a combinatorial problem, which is a branch of discrete optimization problems. Also, in partitional clustering, the set of feasible solutions is finite and grows combinatorially with the problem size [5]. Several studies have used Particle Swarm Optimization (PSO) to solve Combinatorial Optimization Problems (COP) [6]. Clerc [6] provides several examples of PSO applied to combinatorial problems such as the knapsack, the traveling salesman, and the quadratic assignment problems. Recently, some researches have used the combinatorial particle swarm optimization to solve the partitional clustering problems. Jarboui et al. [7] proposed a clustering method based on the Combinatorial Particle Swarm Optimization (CPSO) with fixed number of clusters. In another study, Yucheng and Szu-Yuan [8] proposed a clustering method with variable number of clusters, and they used the CPSO and K-means algorithms. A comprehensive review of evolutionary algorithms for the clustering problem can be found in [9].

The partitional clustering aims at optimizing cluster centers and the number of clusters. One of the major drawbacks of the partitional approaches is the difficulty in determining the number of clusters [2]. In many clustering problems, the correct number of clusters is not known, and it is impossible to estimate. Most clustering algorithms need to determine the number of clusters in advance. A solution for this problem is to use dynamic clustering techniques. Dynamic clustering techniques have two general objectives, finding the optimal number of clusters and partitioning the data objects into clusters.

This paper proposes a combinatorial particle swarm optimization for dynamic data clustering. The proposed method improves the ideas presented by Jarboui et al. [7] and is called Combinatorial Particle Swarm Optimization II (CPSOII). It should be noted that CPSOII can be used for other COPs as well, with only some modifications. CPSOII automatically finds the best number of clusters and partitions the data objects into clusters effectively. Most existing methods for dynamic clustering, such as K-means and Combinatorial Particle Swarm Optimization (KCPSO) [8], Dynamic Clustering using Particle Swarm Optimization (DCPSO) [10] and Two-leveled Symbiotic Evolutionary Clustering Algorithm (TSECA) [11], divide the problem into two subproblems: first, finding the number of clusters and second, partitioning the data objects into clusters. They usually use K-means clustering algorithm for the second objective, however, in K-means algorithm, the initialization step may have an effect on the clustering results [3] and decreases the general performance of the clustering method. CPSOII considers the clustering problem as a single problem and simultaneously finds both the number of clusters and the corresponding clustering results.

Like most existing methods [7, 8, 12–16], CPSOII uses a representation called label-based to encode each clustering solution. This representation is naturally redundant and can reduce the diversity of population in swarm intelligence algorithms. Most of the existing methods, such as CPSO [7] and KCPSO [8] have not paid attention to this problem. In contrast, CPSOII proposes a novel renumbering procedure to solve this problem. The proposed renumbering procedure is used as a preprocessing step before applying the extended PSO operators, and has two distinct advantages. First, CPSOII algorithm will be independent of different encodings of a clustering solution. Second, it increases the rate of distinct particles in swarm and hence, the diversity of population, speed of convergence and quality of solution are increased.

The performance of CPSOII has been evaluated by both artificial and real-world data sets with respect to three clustering metrics, Sum of Squared Error (SSE), Variance Ratio Criterion (VRC) and Davies-Bouldin Index (DBI). The experimental results of CPSOII are compared with CPSO [7], Clustering Genetic Algorithm (CGA) [12] and K-means algorithms, when the number of clusters is known and with KCPSO [8] and CGA [12], when the number of clusters is unknown.

The experimental results show that CPSOII is very effective, robust and can solve clustering problems successfully with both known and unknown number of clusters. Comparing the obtained results of CPSOII with CPSO and other clustering techniques, such as KCPSO, CGA and K-means, reveals that CPSOII yields promising results, e.g., it improves 9.26 % of the value of DBI criterion for Hepato [17] data set. Additionally, the number of clusters found by CPSOII algorithm is the closest number to that of classes in different data sets. Furthermore, the rate of distinct particles in swarm and the speed of convergence of CPSOII are higher than the other compared methods.

The rest of this paper is organized as follows. In Sect. 2, related works are briefly reviewed. In Sect. 3, clustering problem, particle swarm optimization and combinatorial PSO algorithm for partitional clustering problem are described as a background. In Sect. 4, CPSOII algorithm is described in detail. Section 5 presents the clustering criteria, implementations, benchmark data sets, experimental setups and the results obtained by CPSOII in comparison to the other clustering methods. Finally, conclusions are presented in Sect. 6.

2 Related works

Clustering techniques based on Evolutionary Computing (EC) and Swarm Intelligence algorithms (SI) have outperformed many classical clustering techniques [18]. There have been a lot of studies in the literature for data clustering with evolutionary algorithms [9]. Although most of them are based on a fixed number of clusters, there are methods with a variable number of clusters, in which the clustering algorithm attempts to find optimal number of clusters. These works will be briefly reviewed in this section.

Compared with other evolutionary algorithms, Genetic Algorithm (GA) has been most frequently used in the clustering problems. Hruschka et al. [12, 13] proposed a dynamic clustering algorithm based on GA. Their algorithm was called CGA, it uses label-based integer encoding, in which a chromosome is an integer vector of N positions. Each element of this vector is associated with a data object and takes a value (cluster label) over the alphabet $\{1, 2, 3, \ldots, K\}$, where K is the number of clusters. In this clustering algorithm, a limited crossover and a proportional selection are used. Bandyopadhyay and Maulik [19] proposed a GA-based algorithm to solve dynamic clustering problems. Their algorithm is called Genetic Clustering for Unknown K (GCUK). Liu et al. [20] developed a GA-based clustering method which is called Automatic Genetic Clustering for Unknown K (AGCUK). In this method, noisy selection and division–absorption mutation were designed to keep a balance selection between pressure and population diversity.

Modified PSO algorithm for solving clustering is the emphasis of PSO-based clustering methods. Jarboui et al. [7], presented a discrete PSO algorithm called CPSO to solve partitional clustering problems. This method uses the label-based representation to encode clustering solutions and the number of clusters is known or set in advance. Karthi et al. [21] proposed a Discrete Particle Swarm Optimization Algorithm (DPSOA) for data clustering. In DPSOA algorithm, particles are represented as groups of data objects called cluster blocks. Also, Omran et al. [10] proposed a dynamic clustering approach, based on PSO. Their algorithm was called DCPSO and applied to an unsupervised image classification. In DCPSO, the binary PSO is used to find the best number of clusters, and the center of the chosen clusters is then refined via K-means clustering algorithm. Yucheng and SzuYuan [8] used CPSO [7] and K-means algorithms for dynamic data clustering. Their algorithm was called KCPSO. In each iteration of KCPSO, the CPSO algorithm is used to optimize the number of clusters, and then K-means is used to find the best clustering result. The authors compared KCPSO with DCPSO and GCUK and the results revealed that in most cases, KCPSO outperforms DCPSO and GCUK in both finding the best number of clusters and clustering results. Latif et al. [22] presented a dynamic clustering method, based on binary multi-objective PSO that is called Dynamic Clustering using Binary Multi-Objective Particle Swarm Optimization (DCBMPSO). Paoli et al. [23] also used multi-objective PSO for dynamic clustering. This method was applied on hyper-spectral images as a case study. In this method, each particle position is represented as a vector of $2 \times K \times D + D$ elements, where K is the number of clusters represented by a particle, and D is the number of dimensions of data objects. First $K \times D$ elements of particle position are the mean of clusters. Second $K \times D$ elements are the variance of clusters and last D elements determine the selected and unselected dimensions of data objects for clustering.

The hybrid evolutionary algorithms have also been used in clustering. Niknam et al. [4] presented a hybrid evolutionary algorithm based on PSO and Simulated Annealing (SA) to find optimal cluster centers. In another study, Niknam and Amiri [24] proposed a cluster analysis optimization algorithm based on a combination of PSO, Ant Colony Optimization (ACO) and K-means. Siriporn and Kim [25] also proposed a combination of GA, ACO and fuzzy c-means for partitional clustering problem.

3 Background

In this section, we describe the clustering problem, particle swarm optimization and combinatorial PSO algorithm for partitional clustering problems.

3.1 Clustering problem

The clustering problem is formally defined as follows:

Consider a set of N d-dimensional data objects $O = \{O_1, O_2, \ldots, O_\mathrm{N}\}$, where $O_i = (o_{i1}, o_{i2}, \ldots, o_{id}) \in \mathscr{R}^d$. Each o_{ij} called a feature (attribute, variable, or dimension) and presents value of data object i at dimension j.

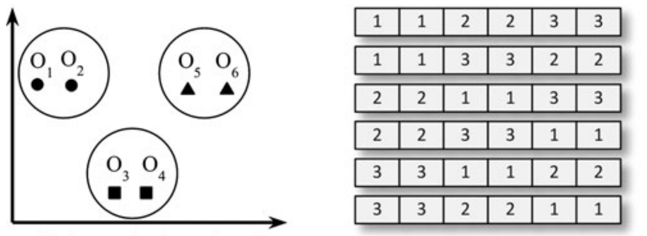
Given O the set of data objects, the goal of partitional clustering is to divide the data objects into K clusters $\{C_1, C_2, \ldots, C_\mathrm{K}\}$, that satisfies the following conditions:

(a) $C_i \neq \varphi, i = 1, \ldots, K$
(b) $\bigcup_{i=1}^{K} C_i = O$
(c) $C_i \cap C_j = \varphi, i, j = 1, \ldots, K$ and $i \neq j$

In general, the data objects are assigned to clusters based on a similarity measure. In this paper, we use Euclidean distance [3] for this purpose.

3.1.1 Encoding schemes

Several encoding schemes of clustering solutions have been proposed in the literature. Hruschka et al. [9] categorized them into three types: binary, integer and real. In this paper, CPSOII uses a label-based representation. This representation is an important integer encoding, in which a solution is an integer vector of N labels, each label corresponds to a particular data object and determines the cluster number of data object in the solution [9]. However, the label-based representation is naturally redundant and one-to-many, i.e.,



1	1	2	2	3	3
1	1	3	3	2	2
2	2	1	1	3	3
2	2	3	3	1	1
3	3	1	1	2	2
3	3	2	2	1	1

(a) A sample clustering of data set

(b) Redundant clustering encoding representing a clustering solution

Fig. 1 An example of the label-based integer encoding and redundant solutions

there are $K!$ different solutions that represent the same solution [9]. Figure 1 shows an example of the label-based representation with redundant solutions. The problem can be solved by applying a renumbering procedure [26].

3.2 Particle swarm optimization (PSO)

PSO is an evolutionary optimization technique introduced by Kennedy and Eberhart [27], which is inspired by the social behavior of bird flocking and fish schooling. PSO is a population-based search method, where the individuals, referred to as particles, are grouped into a swarm. Each particle represents a candidate solution to the optimization problem and is determined by its position and velocity. In addition, each particle has memory and retains its best experience (*Pbest*).

PSO combines self-experiences with social-experience. In this algorithm, a swarm of particles flies through the search space. However, this search process is not carried out entirely randomly and the position of a particle is influenced by the following factors: best position visited by itself (i.e., its own best experience), the position of the best particle in its neighborhood (i.e., the best social experience), and its current velocity. When a particle takes the entire population as its neighbors, the best value is a global best (called *Gbest*) and when it takes the smaller group as its neighbors, the best value is a local best, called *Lbest*. The performance of each particle is measured according to a predefined fitness function.

During the search process, the particles are moved according to the following equations:

$$v_i^{t+1} = wv_i^t + c_1r_1\big(Pbest_i^t - x_i^t\big) + c_2r_2\big(Gbest^t - x_i^t\big) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

Where x_i^t and v_i^t are the current position and the velocity of ith particle at iteration t, respectively. $Gbest^t$ and $Pbest_i^t$ are the global and personal best position of ith particle during iterations 1 to t, respectively. w is the inertia weight that controls the impact of the previous velocities on the current velocity. r_1 and r_2 are uniformly distributed random variables in range [0, 1] to provide stochastic weighting of the different components participating in the particle velocity definition. c_1 and c_2 are the factors to determine the impact of the personal best and the global best, respectively. After updating the velocity and position of a particle, its *Pbest* is updated according to (3).

$$Pbest_i^{t+1} = \begin{cases} x_i^{t+1} & \text{if } f(x_i^{t+1}) < f(Pbest_i^t), \\ Pbest_i^t & \text{otherwise.} \end{cases} \quad (3)$$

Where $f(x_i^{t+1}) < f(Pbest_i^t)$ means that the new position x_i^{t+1} is better than the current *Pbest* of the ith particle. After updating the velocity, position and *Pbest* of all particles, the particle with the best fitness is selected as $Gbest^{t+1}$. These operations are repeated until a termination criterion is met (e.g., the number of iterations is performed, or the adequate fitness is reached).

The main strength of PSO is its fast convergence, in contrast to many global optimization algorithms like GA, SA, Tabu Search (TS) and etc. [28]. But for applying PSO successfully, one of the key issues is to find how to map the problem solution into the PSO particle, which have great impacts on its feasibility and performance [29].

3.3 Combinatorial PSO

Jarboui et al. [7] presented a new discrete PSO algorithm called CPSO to solve partitional clustering problems. In this paper, CPSO algorithm is presented as CPSOI. CPSOI, like many clustering algorithms, needs to determine the number of clusters in advance.

It is worth remembering that the original PSO, as opposed to CPOSI, is basically developed for continuous optimization problems. CPSOI essentially differs from the original (or continuous) PSO in two characteristics: particle and velocity definitions. In the original PSO, a real-based representation is employed to encode each particle. Analogously, CPSOI uses the label-based representation to encode clustering solutions. In contrast to the velocity definition of the original PSO mentioned in (1), CPSOI uses (4) to update the velocity of each particle.

$$v_{ij}^{t+1} = wv_{ij}^t + r_1c_1\big(-1 - y_{ij}^t\big) + r_2c_2\big(1 - y_{ij}^t\big) \quad (4)$$

where

$$y_{ij}^t = \begin{cases} 1 & \text{if } x_{ij}^t = Gbest_j^t, \\ -1 & \text{if } x_{ij}^t = Pbest_{ij}^t, \\ -1 \text{ or } 1 \text{ randomly} & \text{if } (x_{ij}^t = Gbest_j^t = Pbest_{ij}^t), \\ 0 & \text{otherwise.} \end{cases}$$

Where Y_i^t is a dummy variable used to permit the transition from the combinatorial state to the continuous state and vice versa. The change of the velocity v_{ij}^t depends on the value of Y_i^{t-1}. It should be noted that if the velocity of a particle is more than the maximum user-defined speed limit (V_{max})

or less than the minimum user-defined speed limit (V_{min}), it will be set V_{max} or V_{min}, respectively.

After updating the velocity, the position of each particle is updated according to (5).

$$x_{ij}^{t+1} = \begin{cases} Gbest_j^t & \text{if } y_{ij}^{t+1} = 1, \\ Pbest_{ij}^t & \text{if } y_{ij}^{t+1} = -1, \\ \text{a random number} & \text{otherwise,} \end{cases} \tag{5}$$

where

$$y_{ij}^{t+1} = \begin{cases} 1 & \text{if } \lambda_{ij}^{t+1} > \alpha, \\ -1 & \text{if } \lambda_{ij}^{t+1} < -\alpha, \\ 0 & \text{otherwise,} \end{cases}$$

$$\lambda_{ij}^{t+1} = y_{ij}^t + v_{ij}^{t+1}.$$

Where α is manually determined by experts as a parameter for fitting intensification and diversification.

4 Proposed CPSOII

CPSOI has several shortcomings. First, it needs to determine the number of clusters in advance. Second, it uses the label-based representation to encode clustering solutions, therefore, the particles of swarm are prone to the redundancy. Third, during the evolution process, it may generate invalid particles, because the number of clusters is fixed and it is possible to generate an empty cluster. CPSOII is an extension of the original PSO algorithm for solving partitional clustering problems, it automatically finds the best number of clusters and simultaneously categorizes data objects. The general process of CPSOII algorithm is given in Fig. 2.

CPSOII algorithm consists of seven steps. In the first step, algorithm parameters, such as swarm size, maximum number of iterations, and parameters used in the velocity equation are initialized. In the second step, the initial particles of swarm are generated and the *Pbest* of each particle is initialized with current position of the corresponding particle. This step includes two activities: generating particles by using K-means algorithm and generating particles randomly. After generating initial particles, the fitness of each particle is calculated in the third step. Thereafter, in the fourth step, the particle with the best fitness is selected as the *Gbest* particle. In the fifth step, the position and velocity of each particle X_i^t is updated. This step has four sub-steps. First, the renumbering procedure is used to renumber *Pbest* and *Gbest* clusters according to their similarity with X_i^t clusters. Then, the velocity equation of CPSOII (see (6)) is used to compute the velocity of the particle and then the particle is moved to a new position. Finally, the validity of the particle is evaluated and if it is invalid, the correction procedure is used to make it valid. After updating the velocity and position of each particle according to the fifth step, the fitness

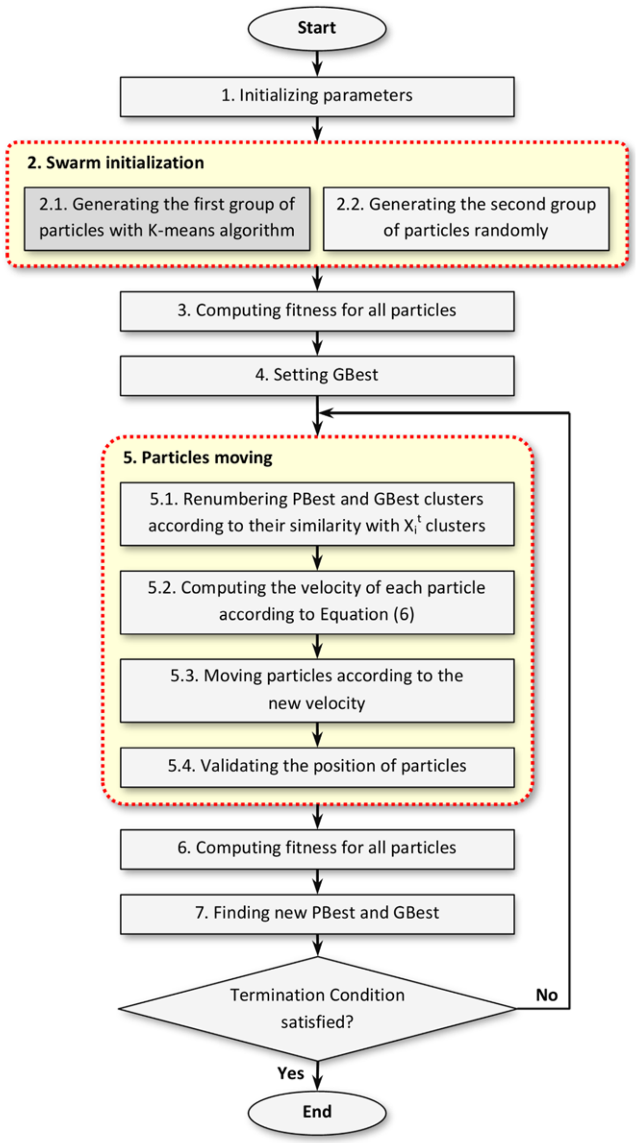

Start
1. Initializing parameters
2. Swarm initialization
2.1. Generating the first group of particles with K-means algorithm
2.2. Generating the second group of particles randomly
3. Computing fitness for all particles
4. Setting GBest
5. Particles moving
5.1. Renumbering PBest and GBest clusters according to their similarity with X_i^t clusters
5.2. Computing the velocity of each particle according to Equation (6)
5.3. Moving particles according to the new velocity
5.4. Validating the position of particles
6. Computing fitness for all particles
7. Finding new PBest and GBest
Termination Condition satisfied?
No
Yes
End

Fig. 2 General process of CPSOII algorithm

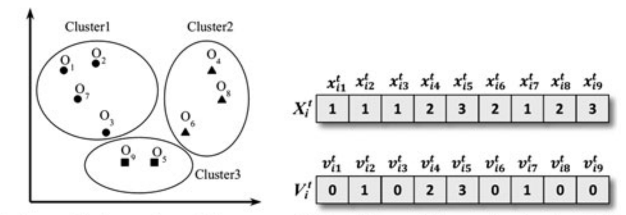
of all particles are computed again in the sixth step. Finally, in the seventh step, *Gbest* and *Pbest* of each particle are determined. In the mentioned steps, the steps 1–4 are executed only once, but steps 5–7 are repeated until the termination condition is satisfied.

The main concepts of CPSOII and its steps are described in the following sections.

4.1 Cluster encoding in particles

CPSOII uses the label-based integer encoding to represent the clustering solutions. With this schema, the position X_i of each particle will simply be a vector that provides integer numbers representing the cluster number of data objects. The velocity V_i of each particle will be a vector of integer numbers representing the recent move.

In this paper, we represent the swarm as a set of M particles. Each particle is determined with its position, velocity and Pbest. Each particle position X_i and particle velocity V_i, $i = 1, \dots, M$, characterized by N elements, $x_{i1}, \dots, x_{ij}, \dots, x_{iN}$ and $v_{i1}, \dots, v_{ij}, \dots, v_{iN}$, where N is the number of data objects, $x_{ij} \in \{1, \dots, K_i\}$, represents the cluster number of jth data object in ith particle and $v_{ij} \in \{0, \dots, K_i\}$, represents the recent move of jth data object in ith particle. The velocity value of 0 in each element means that cluster number of corresponding data objects in the previous move was not changed. A representation of the particle position and velocity are shown in Fig. 3. It should be noted that K_i is the number of clusters related to particle i and is assumed to lie in the range $[K_{min}, K_{max}]$, where the value of K_{min} by default is 2, unless it is manually specified and the value of K_{max} by default is chosen $\sqrt{N}+1$ [30], unless it is manually specified.


Cluster1
Cluster2
Cluster3
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9
X_i^t
x_{i1}^t x_{i2}^t x_{i3}^t x_{i4}^t x_{i5}^t x_{i6}^t x_{i7}^t x_{i8}^t x_{i9}^t
1 1 1 2 3 2 1 2 3
V_i^t
v_{i1}^t v_{i2}^t v_{i3}^t v_{i4}^t v_{i5}^t v_{i6}^t v_{i7}^t v_{i8}^t v_{i9}^t
0 1 0 2 3 0 1 0 0

(a) A sample clustering of data set (b) Particle position and velocity of (a)

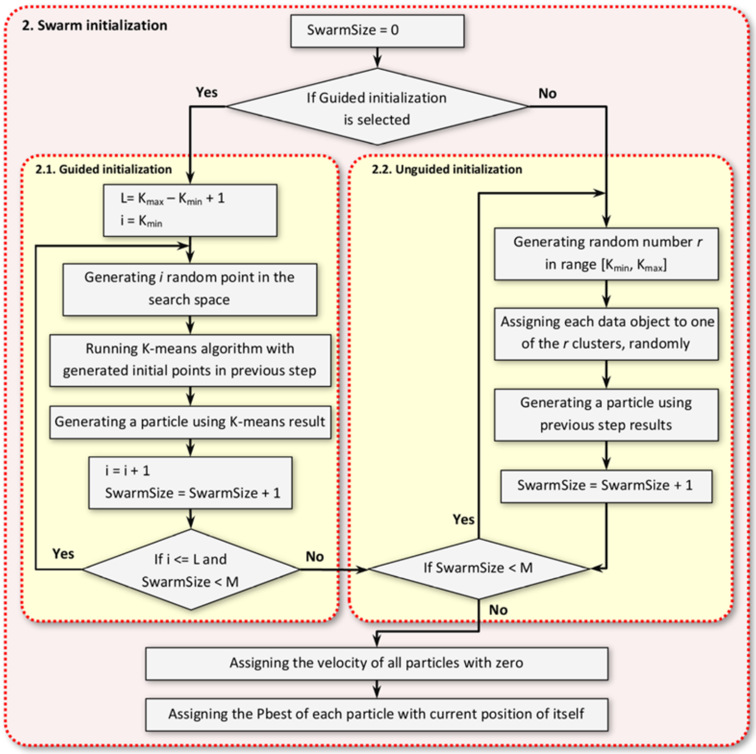
Fig. 3 CPSOII particle position and velocity structure

4.2 Swarm initialization

The most common technique in EC for initializing population is *random uniform initialization*, in which each particle of the initial swarm and, consequently, the initial best positions are drawn by sampling a uniform distribution over the search space [31]. Random uniform initialization is an *unguided initialization* process to generate the initial swarm and if there are items of information available regarding the location of the global minimizer in the search space, then it is capable to initialize the swarm around global minimizer and generate better particles. This is called *guided initialization*.

CPSOII algorithm uses both guided and unguided initialization processes to generate the initial swarm of particles. The flowchart of the swarm initialization is shown in Fig. 4. The guided initialization in CPSOII algorithm is optional and if it is not used, only the unguided initialization is used to generate all particles of swarm. When the guided initialization is used, the initial swarm of particles is partitioned into two groups, the first group is generated by K-means algorithm (guided initialization) and the second one is randomly generated (unguided initialization). The maximum number of particles in the first group is $L = K_{max} - K_{min} + 1$ (note that if the maximum swarm size (M) is smaller than L, then $L = M$). K-means algorithm is performed L times and each time with 5 iterations. Each run of K-means algorithm generates one particle with K clusters, where the value of K for run i is $K_{min} + i - 1$. Initial points of K-means algorithm

Fig. 4 Flowchart of swarm initialization


2. Swarm initialization
SwarmSize = 0
If Guided initialization is selected
Yes
No
2.1. Guided initialization
$L = K_{max} - K_{min} + 1$
$i = K_{min}$
Generating i random point in the search space
Running K-means algorithm with generated initial points in previous step
Generating a particle using K-means result
i = i + 1
SwarmSize = SwarmSize + 1
If i <= L and SwarmSize < M
Yes
No
2.2. Unguided initialization
Generating random number r in range $[K_{min}, K_{max}]$
Assigning each data object to one of the r clusters, randomly
Generating a particle using previous step results
SwarmSize = SwarmSize + 1
If SwarmSize < M
Yes
No
Assigning the velocity of all particles with zero
Assigning the Pbest of each particle with current position of itself

are randomly generated, and thereafter, each data object is assigned to the closest cluster center, based on the Euclidean distance.

The second group of particles is randomly generated. The number of particles in the second group is $M - L$. For particle i from the second group, the number of clusters K_i is randomly generated in the range $[K_{min}, K_{max}]$, then, each data object of data set is randomly assigned to one cluster.

After generating the initial particles of swarm, the initial velocity of all particles are assigned to zero and the initial *Pbest* of each particle is assigned to its current position.

4.3 Fitness evaluation

In this paper, three different criteria, SSE, VRC, and DBI, are separately used to compute the fitness of a particle. SSE is a simple criterion and considers only within-cluster distance. This criterion is appropriate for algorithms with fixed number of clusters. VRC and DBI consider both within-cluster and between-cluster distances and are suitable for the dynamic clustering algorithms. However, the DBI criterion has been most frequently used in dynamic clustering. Hence, only DBI criterion will be used to compare CPSOII with dynamic clustering algorithms, and the three aforementioned criteria will be used to compare algorithms with the fixed number of clusters.

4.4 Particles moving

In CPSOII algorithm, the movement of particles to a new position is performed in four steps as shown in Fig. 2. These steps are described with more details in below.

4.4.1 Cluster renumbering

The goal of the cluster renumbering step is to remove the redundant particles described in Sect. 3.1.1, and to prepare $Pbest_i$ and *Gbest* particles to calculate their difference with particle X_i in the step 5.2. Figure 5 presents a pseudocode of the proposed renumbering procedure. In part 5 of this procedure, the similarity degree between the clusters of two input particles is calculated by using a similarity function, then, in part 6, two clusters with the highest similarity are matched to each other. Part 6 of the procedure is repeated R times, where $R = min\{K_i, K_{Gbest}\}$ (or $R = min\{K_i, K_{Pbest_i}\}$) and in each iteration, two clusters of two input particles are matched to each other and not be considered in the next iterations. Thereafter, in part 9, the clusters of *Gbest* (or $Pbest_i$) are renumbered according to their matches with clusters of X_i. Due to different variances of the number of clusters in different particles, it is possible that some clusters of *Gbest* or $Pbest_i$ will not be matched to any cluster of X_i or vice versa. In this case, after renumbering matched clusters, unmatched clusters of particles use the unused numbers in the

```
5.1. Renumbering procedure
Input: Particle $X_i$ with clusters $\{C_1, ..., C_{K_i}\}$ and Gbest (or Pbest$_i$) with
       clusters $\{\hat{C}_1, ..., \hat{C}_{K_{Gbest}}\}$
Output: Gbest (or Pbest$_i$) with renumbered clusters
Begin
  1.  A = $\{C_1, ..., C_{K_i}\}$
  2.  B = $\{\hat{C}_1, ..., \hat{C}_{K_{Gbest}}\}$
  3.  R = min $\{K_i, K_{Gbest}\}$
  4.  MatchSet = $\emptyset$
  5.  Compute the similarity degree of A members with B members by
      using a similarity function
  6.  For i=1 to R
        6.1.  Select cluster $C_i$ from A, and cluster $\hat{C}_j$ from B with the
              highest similarity degree
        6.2.  MatchSet = MatchSet U {(i, j)}
        6.3.  A = A – $\{C_i\}$
        6.4.  B = B – $\{\hat{C}_j\}$
      End for
  7.  D = $\{\hat{C}_1, ..., \hat{C}_{K_{Gbest}}\}$
  8.  NewGbest = $\emptyset$
  9.  For each (i, j) ϵ MatchSet
        9.1.  Change the cluster number $\hat{C}_j$ from j to i ($\hat{C}_i$)
        9.2.  NewGbest = NewGbest U $\{\hat{C}_i\}$
        9.3.  D = D – $\{\hat{C}_j\}$
        9.4.  MatchSet = MatchSet – {(i, j)}
      End for
  10. For each $\hat{C}_j$ from D
        10.1. $\hat{C}_j$ is renumbered to the one of unused numbers in cluster
              numbering of NewGbest, with start from small numbers
        10.2. NewGbest = NewGbest U $\{\hat{C}_j\}$
        10.3. D = D – $\{\hat{C}_j\}$
      End for
  11. Gbest = NewGbest
End
```

Fig. 5 Pseudocode of the renumbering procedure

cluster numbering (part 10). Table 1 shows the list of binary similarity functions used in CPSOII. Choi et al. [32] collected and analyzed 76 binary similarity functions and distance measures used over the last century.

Example 1 Figure 6 illustrates an example of the cluster renumbering process in CPSOII algorithm. The similarity degree of Cluster1 of *Gbest* with Cluster1 of X_i, using Jaccard similarity function (see Table 1), is $\frac{3}{4}$ and it is the highest similarity degree in this example. Therefore, these clusters are matched to each other. After that, Cluster4 of *Gbest* and Cluster2 of X_i, with similarity $\frac{2}{3}$ is the highest similarity degree and are matched to each other. Finally, Cluster2 of *Gbest* is matched to Cluster3 of X_i. Figure 6(c) illustrates the result of the matching process. After matching, the clusters of *Gbest* are renumbered to the corresponding matched cluster numbers. Cluster3 from *Gbest* is an unmatched cluster, so it gets an unused cluster number that is 4 in this example.

The renumbering procedure makes CPSOII insensitive to different encodings of clustering solutions.

Example 2 Consider two particles with the following encodings: [2222333111], [1112222333]—each one represents three clusters. Assuming that the first particle is a better solution. In the movement of the second particle toward

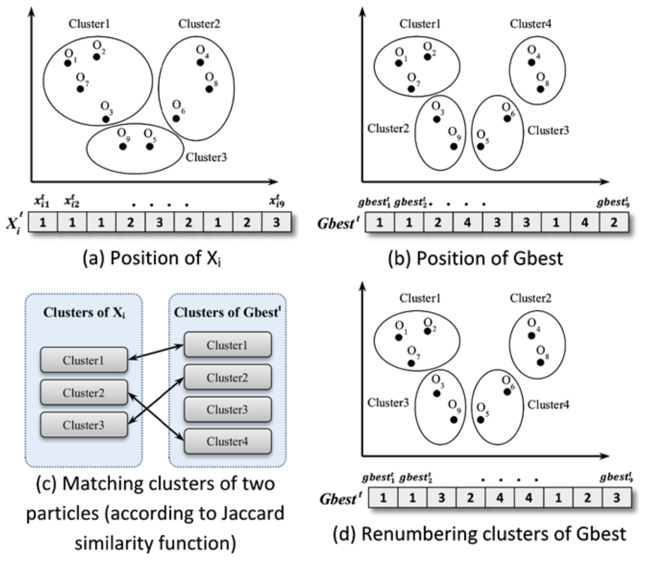

Cluster1
Cluster2
Cluster3
Cluster4
(a) Position of Xi
(b) Position of Gbest
Clusters of Xi
Clusters of Gbest
(c) Matching clusters of two particles (according to Jaccard similarity function)
(d) Renumbering clusters of Gbest

Fig. 6 Example of the cluster renumbering process in CPSOII

Table 1 Used similarity measures for binary data

Measure	Forms
Simple Matching coefficient	$S(c_i, c_j) = \frac{n_{11}+n_{00}}{n_{11}+n_{00}+n_{10}+n_{01}}$
Rogers and Tanimoto measure	$S(c_i, c_j) = \frac{n_{11}+n_{00}}{n_{11}+n_{00}+2(n_{10}+n_{01})}$
Jaccard coefficient	$S(c_i, c_j) = \frac{n_{11}}{n_{11}+n_{10}+n_{01}}$
Sokal and Sneath measure1	$S(c_i, c_j) = \frac{n_{11}}{n_{11}+2(n_{10}+n_{01})}$
Sokal and Sneath measure2	$S(c_i, c_j) = \frac{n_{11}+n_{00}}{n_{11}+n_{00}+\frac{1}{2}(n_{10}+n_{01})}$
Dice coefficient	$S(c_i, c_j) = \frac{n_{11}}{n_{11}+\frac{1}{2}(n_{10}+n_{01})}$
NEI & LI measure	$S(c_i, c_j) = \frac{2n_{11}}{(n_{11}+n_{10})+(n_{11}+n_{01})}$

n_{00} denotes the number of data objects absent in both c_i and c_j (negative matches)

n_{11} denotes the number of data objects present in both c_i and c_j (positive matches)

n_{01} denotes the number of data objects absent in c_i and present in c_j.

n_{10} denotes the number of data objects present in c_i and absent in c_j

the first particle with current information of encoding, it is diverted from better solution because the encodings of two particles are very different and this may divert particles from better solution. Using the renumbering procedure, the first particle encoding is changed to [1111222333] and with this encoding, the movement of the second particle toward the first one, is more deliberate and rational than the previous case. It is an important benefit and it guides the algorithm toward better solutions.

4.4.2 Velocity computation

In PSO algorithm and its extensions, it is the velocity vector that drives the optimization process. In CPSOII algorithm, the velocity of each particle is modified by (6).

$$V_i^{t+1} = W \otimes V_i^t \oplus \left(\left(R_1 \otimes (Pbest_i^t \ominus X_i^t)\right) \oplus \left(R_2 \otimes (Gbest^t \ominus X_i^t)\right)\right) \tag{6}$$

Where X_i^t and V_i^t are the position and velocity of particle i at iteration t, respectively. $Pbest_i^t$ and $Gbest^t$ are the best positions obtained by the particle i and the swarm of particles during time t, respectively. W, R_1 and R_2 are vectors with size N comprising 0 or 1 elements, such that the values of these vectors are randomly generated with probability w, r_1 and r_2, respectively. The values of w, r_1 and r_2 have major effect on the performance of CPSOII.

In (6), *Difference* ($\ominus$), *Multiply* ($\otimes$) and *Merge* ($\oplus$) operators are introduced. Although, *Difference* operator is approximately similar to the calculation of y in the CPSOI algorithm mentioned in (4), *Multiply* and *Merge* operators are quite different from the operators used by CPSOI. In addition, the type of each velocity vector element of CPSOI and CPSOII are different. In CPSOI, the velocity vector is a vector with N real elements, on the contrary, CPSOII uses the velocity vector with N integer elements. The definitions of the operators used in the body of (6) are defined as follows:

Difference operator ($\ominus$) This operator computes the difference between the current position of ith particle, X_i^t, and personal-best position (or global-best position). The $\ominus$ operator is defined in (7) and (8). We use λP_i^t to show the difference of particle X_i^t and its $Pbest_i^t$ position, and λG_i^t for X_i^t and $Gbest^t$ position.

$$\lambda P_i^t = Pbest_i^t \ominus X_i^t \tag{7}$$

where

$$\lambda p_{ij}^t = \begin{cases} pbest_{ij}^t & \text{if } x_{ij}^t \neq pbest_{ij}^t \\ 0 & \text{otherwise} \end{cases},$$
$$pbest_{ij}^t \in \{1, \ldots, \mathrm{K}_{Pbest_i^t}\},$$

$$\lambda G_i^t = Gbest^t \ominus \mathrm{X}_i^t, \tag{8}$$

where

$$\lambda g_{ij}^t = \begin{cases} gbest_j^t & \text{if } x_{ij}^t \neq gbest_j^t \\ 0 & \text{otherwise} \end{cases},$$
$$gbest_j^t \in \{1, \ldots, \mathrm{K}_{Gbest^t}\}.$$

The difference between particle position X_i^t and $Pbest_i^t$ (or $Gbest^t$) is a vector of N elements. If each nonzero element of the difference is replaced with the corresponding element in X_i^t, then, the position of X_i^t will be modified to the position of $Pbest_i^t$ (or $Gbest^t$). CPSOII uses this property to move toward a desired position $Pbest_i^t$ (or $Gbest^t$). According to this property, the difference between two particles with same position is a vector with N zero elements.

Multiply operator ($\otimes$) This operator manages the exploration and exploitation abilities of the swarm, using W, R_1 and R_2 vectors. The values of these vectors are controlled by w, r_1 and r_2 variables. Generally, the value of w controls the movement of particles in the previous direction and the values of r_1 and r_2 control the movement of particles in the direction of *Pbest* and *Gbest*, respectively. The $\otimes$ operator is equivalent to Hadamard product which is defined in (9).

$$C = A \otimes B \quad \text{where } c_i = a_i b_i. \tag{9}$$

Where A and B are two vectors with the same size; a_i and b_i are the ith element of vector A and B, respectively.

Merge operator ($\oplus$) This operator merges two vectors of N elements and defined by (10).

$$Y_i^t = \lambda P_i^t \oplus \lambda G_i^t \tag{10}$$

where

$$y_{ij}^t = \begin{cases} \lambda g_{ij}^t & \text{if } \lambda g_{ij}^t \neq 0 \text{ and } \lambda p_{ij}^t = 0, \\ \lambda p_{ij}^t & \text{if } \lambda g_{ij}^t = 0 \text{ and } \lambda p_{ij}^t \neq 0, \\ \lambda g_{ij}^t \text{ or } \lambda p_{ij}^t \text{ randomly} & \text{if } \lambda g_{ij}^t \neq 0 \text{ and } \lambda p_{ij}^t \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

In CPSOII algorithm, the merge operator is used to merge the three components of the velocity equation (see (6)).

4.4.3 Particle moving

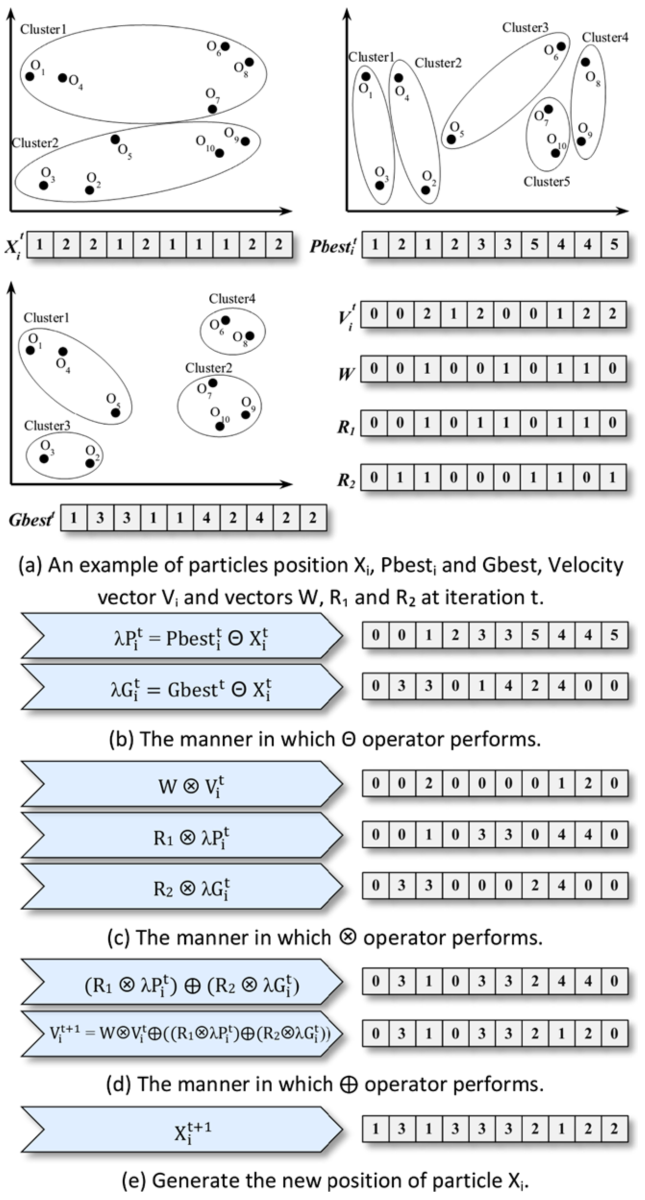
After computing the velocity of a selected particle, the new position of the particle is generated based on (11).

$$x_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1} & \text{if } v_{ij}^{t+1} \neq 0, \\ \text{a random number } r & \text{otherwise.} \end{cases} \tag{11}$$

Where r is an integer random value uniformly distributed in the range $[1, K_i + 1]$ and $K_i + 1 < K_{max}$. With this range, CPSOII algorithm acquires the ability to add a new cluster. However, the difference in the number of clusters of X_i with $Pbest_i$ and $Gbest$ can also cause some clusters in X_i to be added or removed.

After computing the velocity value of each particle in the original PSO, it is added to the current position of the particle and the new position of the particle is generated. In the CPSOII, the velocity vector contains the recent movements of each particle. This means that the non-zero elements of velocity vector are the same as the corresponding elements of the position vector. With this property of velocity vector, we actually map (2) of the original PSO to (6) of CPSOII. Therefore, in (11), there is no *Merge* operator.

Example 3 Figure 7 illustrates an example of creating a new position and velocity of particle X_i^t. Figure 7(b, c, d) illustrates how the velocity operators ($\ominus$, $\otimes$ and $\oplus$) are performed. Figure 7(e) shows the particle movement and the generation of its new position.



Cluster1
Cluster2
X_i^t 1 2 2 1 2 1 1 1 2 2
Cluster1
Cluster2
Cluster3
Cluster4
Cluster5
$Pbest_i^t$ 1 2 1 2 3 3 5 4 4 5
Cluster1
Cluster2
Cluster3
Cluster4
$Gbest^t$ 1 3 3 1 1 4 2 4 2 2
V_i^t 0 0 2 1 2 0 0 1 2 2
W 0 0 1 0 0 1 0 1 1 0
R_1 0 0 1 0 1 1 0 1 1 0
R_2 0 1 1 0 0 0 1 1 0 1
(a) An example of particles position X_i, $Pbest_i$ and Gbest, Velocity vector V_i and vectors W, R_1 and R_2 at iteration t.
$\lambda P_i^t = Pbest_i^t \ominus X_i^t$ 0 0 1 2 3 3 5 4 4 5
$\lambda G_i^t = Gbest^t \ominus X_i^t$ 0 3 3 0 1 4 2 4 0 0
(b) The manner in which $\ominus$ operator performs.
$W \otimes V_i^t$ 0 0 2 0 0 0 0 1 2 0
$R_1 \otimes \lambda P_i^t$ 0 0 1 0 3 3 0 4 4 0
$R_2 \otimes \lambda G_i^t$ 0 3 3 0 0 0 2 4 0 0
(c) The manner in which $\otimes$ operator performs.
$(R_1 \otimes \lambda P_i^t) \oplus (R_2 \otimes \lambda G_i^t)$ 0 3 1 0 3 3 2 4 4 0
$V_i^{t+1} = W \otimes V_i^t \oplus ((R_1 \otimes \lambda P_i^t) \oplus (R_2 \otimes \lambda G_i^t))$ 0 3 1 0 3 3 2 1 2 0
(d) The manner in which $\oplus$ operator performs.
X_i^{t+1} 1 3 1 3 3 3 2 1 2 2
(e) Generate the new position of particle X_i.

Fig. 7 An example of creating a new position and velocity of particle X_i

4.4.4 Position validation

To avoid generating solutions with empty clusters, the new position of each particle is checked. Then, if there are empty clusters, they are removed by the renumbering process. In the label-based representation, a particle is invalid if the number of its clusters is smaller than the largest cluster number used in the cluster encoding. In each iteration of the correction procedure, the cluster with the largest cluster number is renumbered to the smallest unused cluster number.

Example 4 Figure 8 illustrates an invalid particle (in which clusters 1 and 3 are empty) as an example for performing validation via the renumbering process. In this figure, par-

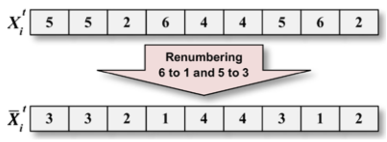


Fig. 8 A sample of invalid solution and applying the correction process

ticle X_i^t is an invalid particle, because it has just 4 clusters, but the largest cluster number is 6.

5 Experimental results

In this section, the performance of five clustering algorithms: CPSOII, CPSOI [7], CGA [12], KCPSO [8] and K-means, are compared. The used criteria for comparing the results are SSE, VRC and DBI.

5.1 Clustering criteria

SSE [3] is one of the most common partitional clustering criteria and its general objective is to obtain a partition which minimizes the squared error. This criterion is defined in (12).

$$SSE = \sum_{i=1}^{K} \sum_{O \in C_i} (c_i - O)(c_i - O)^T \tag{12}$$

where

$$c_i = \frac{1}{N_i} \sum_{O \in C_i} O.$$

Where c_i is the mean of cluster C_i (cluster centroid) and N_i denotes the number of data objects belonging to cluster C_i.

SSE is an appropriate measurement when the number of clusters is known. But for dynamic clustering or when the number of clusters is unknown, SSE cannot be used because its value is reduced by increasing the number of clusters and if the number of clusters is equal to the number of objects, then the value of SSE will be zero.

VRC [33] is another criterion which can be used for cluster validation. This criterion considers both the within-cluster and between-cluster distances. The VRC is defined in (13).

$$VRC = \frac{Inter}{SSE} \times \frac{N-K}{K-1} \tag{13}$$

where

$$Inter = \sum_{i=1}^{K} N_i (c_i - c)(c_i - c)^T$$

$$c = \frac{1}{N} \sum_{i=1}^{N} O.$$

The ratio $(N-K)/(K-1)$ is the normalization term and prevents VRC to increase monotonically with the number of clusters. *Inter* is the between-cluster distance and c is the mean of all data objects (i.e., data centroid). A large value of VRC shows better clustering results based on (13).

One of the popular criteria for the evaluation of clusters is DBI [34]. It combines inter (between-cluster separation) and intra (within-cluster scatter) cluster distances. The within-cluster scatter for cluster C_i is defined in (14) and the between-cluster separation for two clusters C_i and C_j is defined in (15).

$$S_{i,q} = \left(\frac{1}{N_i} \sum_{O \in C_i} \| O - c_i \|_2^q \right)^{\frac{1}{q}} \tag{14}$$

$$d_{ij,t} = \| c_i - c_j \|_t. \tag{15}$$

$S_{i,q}$ denotes the qth root of the qth moment of the objects belonging to cluster C_i with respect to their mean. In (14) and (15), q and t are integer numbers, where (q and $t \geq 1$) can be selected independently. Finally, DBI is defined as:

$$DB = \frac{1}{K} \sum_{i=1}^{K} R_{i,qt} \tag{16}$$

where

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

A small value of DBI shows better clustering results. The computational complexity of DBI is higher than SSE and VRC. However, it is an appropriate criterion for dynamic clustering and is used recently in [8, 11, 14, 20, 35] for this purpose.

5.2 Implementation

CPSOII is implemented in Microsoft Visual C#.Net 2010. All experiments conducted in this paper were run in windows 7 on Desktop pc with Intel Core i5, 2.4 GHz processor and 4 GB real memory. We also implemented CPSOI described in Sect. 3.3, KCPSO and CGA algorithm.

5.3 Experimental data

We evaluate CPSOII using artificial data sets (Table 2) and real-world data sets (Table 3). Note that: (i) in Table 2, just Dataset 1 has non overlapping clusters, (ii) the data sets shown in Table 3 are used as a benchmark for the evaluation of clustering methods in recent years.

5.4 Experiments setup

CPSOII has multiple parameters that must be set by user, including *Swarm Size*, *Max Iterations*, w, r_1, r_2, K_{min}, and

K_{max}. The values of w, r_1 and r_2 parameters determine the exploration and exploitation abilities of the swarm. In CPSOII algorithm, with high values of r_1 and r_2, particles tend to move toward *Pbest* and *Gbest*, respectively. With low values, particle motions are dependent on the value of w, in which if the value of w is small, the particle motions will be more random and if the value of w is large, the particle motions are dependent on the recent value of their velocities. To improve the efficiency and accelerate the search process, it is vital to determine the best value for each of these parameters. For this purpose, we ran CPSOII algorithm with different values of parameters (for all cases of these parameters in the range [0, 1] with step 0.01) using all introduced data sets. Then, we selected the values with the best results. In this experiment, other parameters were set as follows: The swarm is comprised of 150 particles and 2500 iterations. Both K_{min} and K_{max} take the number of classes in the corresponding data sets. Figure 9 illustrates the impacts of w, r_1 and r_2 parameters on the VRC criterion using Hepato data set. The re-

Table 2 Description of the artificial data sets [36] used in [8, 19, 35]

Dataset	# Data objects	# Features	# Classes	Size of Classes
Dataset 1	400	3	4	Size of each class is 100
Dataset 2	250	2	5	Size of each class is 50
Dataset 3	300	2	6	Size of each class is 50
Dataset 4	500	2	10	Size of each class is 50

Table 3 Description of the real-world data sets

Dataset	# Data objects	# Features	# Classes	Size of Classes
Iris [37]	150	4	3	Size of each class is 50
Glass [37]	214	9	7	70, 76, 17, 29, 13, 9, 0
Breast Cancer [37]	683	9	2	239, 444
Libras Movement [37]	360	90	15	Size of each class is 24
Hepato [17]	536	9	4	116, 178, 124, 118
Vowel [17]	871	3	6	72, 89, 172, 151, 207, 180

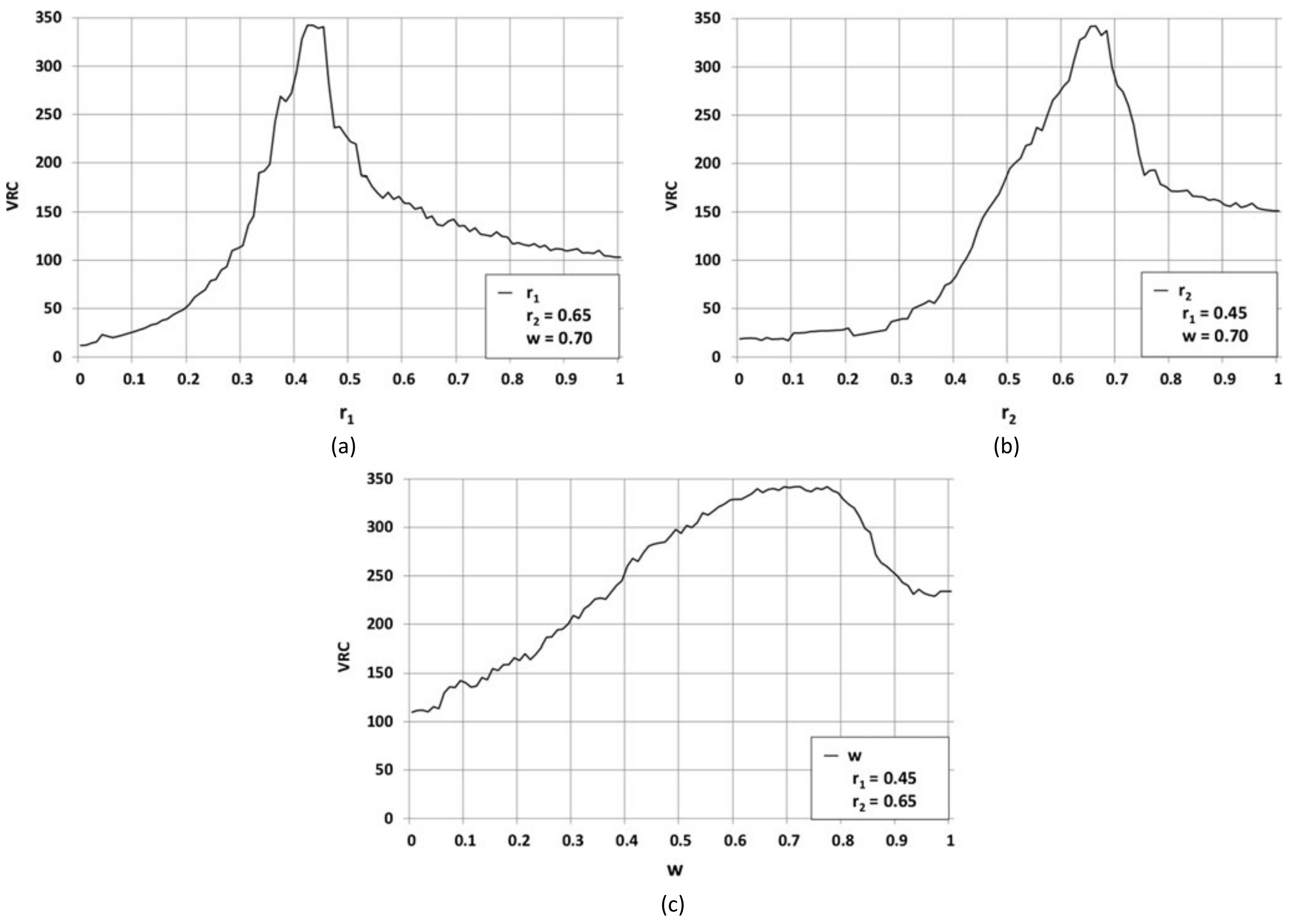


Fig. 9 Effects of w, r_1 and r_2 parameters on the VRC criterion using Hepato data set (Average of 10 runs)

sults show that the suitable value for r_1 is in range [0.4, 0.5] (Fig. 9(a)), for r_2 is in range [0.6, 0.7] (Fig. 9(b)) and for w is in range [0.6, 0.8] (Fig. 9(c)). We tested other introduced data sets and got almost the same results. However, to obtain better results on other data sets, the values of w, r_1 and r_2 may be modified.

The parameter settings of CPSOI, KCPSO and CGA algorithms were determined by both referring to their original papers and performing empirical studies. Table 4 illustrates the parameter settings of CPSOI, CPSOII, KCPSO, CGA and K-means algorithms.

5.5 Experimentations and results

In this section, we describe experiments carried out to test the performance of CPSOII algorithm. For this purpose, we investigate the effectiveness of CPSOII in two cases, with the unknown and known number of clusters. In the first case, the performance of CPSOII is compared with KCPSO [8] and CGA [12]. In the second case, the performance of CPSOII is compared with CPSOI [7], CGA [12] and K-means clustering algorithms.

Thereafter, we compare the speed of convergence of CPSOII, CPSOI, CGA and KCPSO algorithms toward qualified solutions, in both with and without using K-means algorithm for the swarm initialization (Guided and Unguided initialization). Finally, we evaluate the effect of the renumbering procedure and various similarity functions on the quality of solutions and the rate of distinct particles.

In all cases, algorithms run 10 times and the obtained results for each algorithm are reported. Note that the runtime of each algorithm is reported when the fitness value of the evolutionary algorithms (CPSOII, CPSOI, KCPSO and CGA) does not change during 50 iterations.

5.5.1 Experimental results with unknown number of clusters

CPSOII is a dynamic clustering method and automatically finds the best number of clusters during the clustering process. In this section, we compare CPSOII with KCPSO [8] and CGA [12], when the number of clusters is unknown. It should be mentioned that KCPSO uses K-means in its initialization phase. Therefore, Guided & Unguided initialization is not applicable on KCPSO. For this reason, we have executed KCPSO without any modifications. The results provided by CPSOII and their comparison with KCPSO and CGA algorithm, are shown in Tables 5 and 6. For each data set, the best results obtained by algorithms are highlighted in bold face. The differences between number of classes and Avg K of CPSOII, KCPSO and CGA for artificial and real-world data sets are shown in Figs. 10 and 11, respectively.

Table 4 Parameter settings of CPSOI, CPSOII, KCPSO, CGA and K-means algorithms

CPSOI		CPSOII		KCPSO		CGA		K-means	
Swarm size	150	Swarm size	150	Swarm size	150	Population size	150	Max Iterations	1500
Max Iterations	4000	Max Iterations	2500	Max Iterations	2500	Max Iterations	4000	K	Fixed
w, α	0.85, 0.6	w	0.70	w, α	0.72, 0.6	Mutation probability	0.05	Distance Measure	Euclidean
c_1, c_2	0.3, 1.2	r_1, r_2	0.45, 0.65	c_1, c_2	0.5, 0.5	K_{min}	2		
K	Fixed	K_{min}	2	K_{min}	2	K_{max}	By user		
Distance Measure	Euclidean	K_{max}	$\sqrt{N}+1$	K_{max}	By user	Distance Measure	Euclidean		
		Distance Measure	Euclidean	Distance Measure	Euclidean				
				K-means Iteration	500				

Table 5 Experimental results for artificial data sets with unknown number of clusters (Avg: Average, SD: Standard Deviation)

Dataset	# Classes	CPSOII with the Unguided initialization			CPSOII with the Guided & Unguided initialization			KCPSO			CGA with the Unguided initialization			CGA with the Guided & Unguided initialization		
		Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)
Dataset 1	4	4 ±0.00	0.4412 ±0.000	32.20	4 ±0.00	0.4412 ±0.000	6.63	4 ±0.00	0.4412 ±0.000	42.20	4 ±0.00	0.4412 ±0.000	37.17	4 ±0.00	0.4412 ±0.000	6.92
Dataset 2	5	5 ±0.00	0.6218 ±0.010	24.70	5 ±0.00	**0.6180** **±0.005**	6.24	5.2 ±0.60	0.6537 ±0.020	49.33	4.9 ±0.30	0.6314 ±0.014	31.27	4.9 ±0.30	0.6225 ±0.008	6.80
Dataset 3	6	6 ±0.00	0.3555 ±0.000	18.04	6 ±0.00	0.3555 ±0.000	6.88	5.5 ±0.67	0.3986 ±0.078	34.51	5.9 ±0.30	0.3571 ±0.005	27.87	6 ±0.00	0.3555 ±0.000	8.66
Dataset 4	10	**9.6** **±0.66**	0.5889 **±0.010**	25.33	9.5 ±0.67	**0.5780** ±0.016	8.84	8.8 ±0.75	0.6360 ±0.038	62.36	9.1 ±0.94	0.5961 ±0.035	28.06	9.2 ±0.87	0.5908 ±0.015	12.79

Table 6 Experimental results for real-world data sets with unknown number of clusters

Dataset	# Classes	CPSOII with the Unguided initialization			CPSOII with the Guided & Unguided initialization			KCPSO			CGA with the Unguided initialization			CGA with the Guided & Unguided initialization		
		Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)	Avg *K* ± SD	Avg DBI ± SD	Avg Time (s)
Iris	3	**2.8** **±0.40**	0.4228 ±0.024	17.88	2.7 ±0.46	**0.4148** ±0.021	9.38	2.3 ±0.46	0.4356 ±0.043	47.30	2.2 ±0.40	0.4341 ±0.045	21.49	2 ±0.00	0.4231 **±0.017**	11.42
Glass	6	5.9 ±0.54	0.6816 ±0.065	41.42	**6.1** **±0.54**	**0.6556** ±0.038	32.37	5.4 ±1.02	0.7449 ±0.061	89.37	5.6 ±0.80	0.7231 ±0.060	74.16	5.5 ±1.28	0.6994 **±0.023**	50.55
Breast Cancer	2	2 ±0.00	0.7570 ±0.000	117.76	2 ±0.00	0.7570 ±0.000	41.08	2 ±0.00	0.7572 ±0.000	211.81	2 ±0.00	0.7570 ±0.000	157.12	2 ±0.00	0.7570 ±0.000	68.13
Libras Movement	15	**15.3** **±0.64**	1.0600 ±0.045	753.12	15.5 ±0.67	**1.0323** **±0.031**	524.36	16.1 ±1.14	1.2973 ±0.052	1862.00	16.1 ±1.04	1.1923 ±0.052	958.06	15.6 ±1.02	1.1675 ±0.059	922.22
Hepato	4	4.6 **±0.92**	0.4593 ±0.097	110.07	**4.3** ±1.00	**0.3973** **±0.072**	38.13	5.6 ±1.20	0.5879 ±0.131	347.62	5.3 ±1.19	0.4981 ±0.106	201.33	4.8 ±0.98	0.4273 ±0.121	69.05
Vowel	6	5.5 ±0.81	0.7038 ±0.048	131.59	**5.7** **±0.78**	**0.6840** **±0.018**	92.69	4.8 ±1.33	0.7709 ±0.046	534.64	5.2 ±0.87	0.7599 ±0.059	194.89	5.3 ±0.81	0.7301 ±0.045	161.03

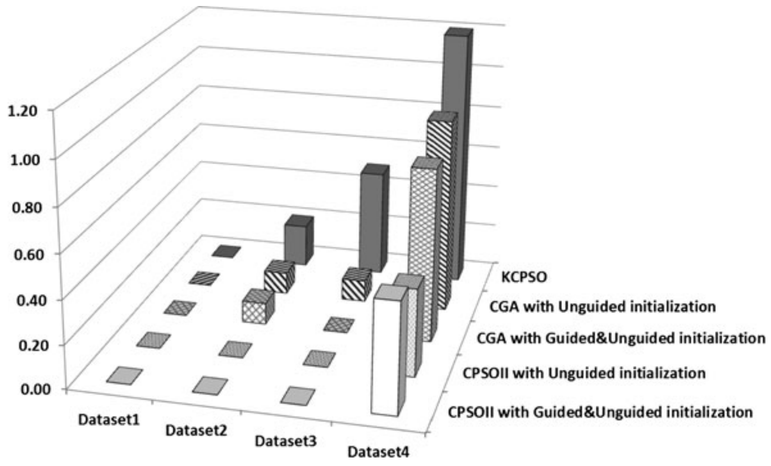


Fig. 10 Differences between number of classes and Avg K of CPSOII, CGA and KCPSO for artificial data sets

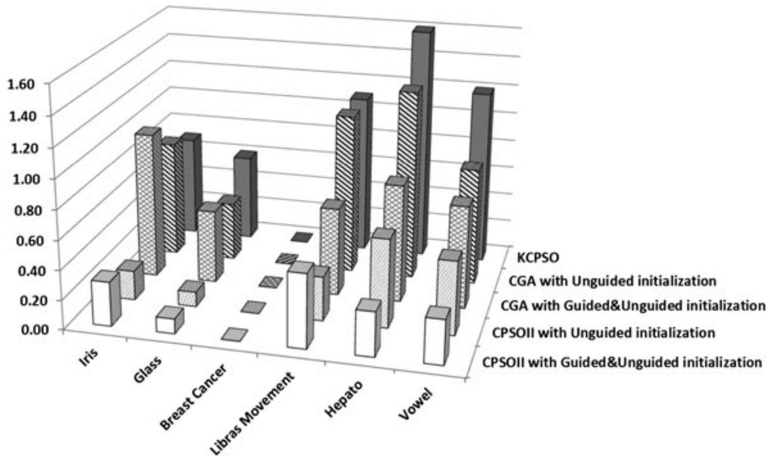


Fig. 11 Differences between number of classes and Avg K of CPSOII, CGA and KCPSO for real-world data sets

The results indicate that CPSOII with the Guided & Unguided initialization performs better than the average DBI for artificial data sets (Table 5). But the value of average K, obtained by CPSOII with only the Unguided initialization, is the closest number to the number of classes of data sets. For real-world data sets (Table 6), CPSOII with the Guided & Unguided initialization performs better than the average DBI and K for most data sets. However, in some cases, CPSOII with the Unguided initialization shows better results. Generally, the average number of clusters found by CPSOII algorithm is the closest number to the number of classes in different data sets.

5.5.2 Experimental results with known number of clusters

For the fixed value of K and parameter settings according to Table 4, the provided results by CPSOII algorithm are compared to CPSOI, CGA and K-means algorithms in Tables 7, 8 and 9. In these tables, Improvement Ratio (IR) is a simple metric to compute the improvement extent of CPSOII in comparison to other methods. Equations (17), (18) and (19) show IR metric according to the DBI, VRC and SSE criteria, respectively.

$$IR_{DBI} = \frac{y - x}{y} \times 100, \tag{17}$$

$$IR_{VRC} = \frac{x - y}{x} \times 100, \tag{18}$$

$$IR_{SSE} = \frac{y - x}{y} \times 100. \tag{19}$$

Where x is the result obtained by CPSOII and y is the best result obtained by other methods.

Based on the results shown in Table 7, for the artificial data sets, CPSOII, CPSOI and CGA algorithms obtain the same best value in the SSE and VRC criteria, but for the DBI criterion, CPSOII algorithm obtains better than the best value for Dataset 2 and Dataset 4. Also, the average value of DBI criterion, obtained by CPSOI and CGA for Dataset 2, Dataset 3 and Dataset 4, are worse than CPSOII. The results of K-means algorithm for all data sets, except Dataset 1, are worse than other algorithms.

Based on the results shown in Tables 8 and 9, for real-world data sets, with both Guided and Unguided initializa-

Table 7 Experimental results for artificial data sets using the Unguided initialization and known number of clusters

Dataset	K	Crite-rion	CPSOII			CPSOI			CGA			K-Means			IR of CPSOII in Best	IR of CPSOII in Avg
			Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)		
Dataset 1	4	DBI	0.4412 ± 0.000	0.4412	20.73	0.4412 ± 0.000	0.4412	24.60	0.4412 ± 0.000	0.4412	22.68	0.5010 ± 0.180	0.4412	2.94	0.00 %	0.00 %
		VRC	3206.68 ± 0.00	3206.68	13.32	3206.68 ± 0.00	3206.68	16.00	3206.68 ± 0.00	3206.68	13.46	3206.68 ± 0.00	3206.68		0.00 %	0.00 %
		SSE	1544.52 ± 0.00	1544.52	10.15	1544.52 ± 0.00	1544.52	12.18	1544.52 ± 0.00	1544.52	12.85	1544.52 ± 0.00	1544.52		0.00 %	0.00 %
Dataset 2	5	DBI	$\mathbf{0.6228} \pm 0.010$	**0.6135**	15.01	0.6277 ± 0.012	0.6175	16.27	0.6278 ± 0.015	0.6175	16.68	0.6717 ± 0.002	0.6692	3.13	**0.65 %**	**0.78 %**
		VRC	389.31 ± 0.00	389.31	9.84	389.31 ± 0.00	389.31	10.19	389.31 ± 0.00	389.31	10.66	386.44 ± 3.18	389.31		0.00 %	0.00 %
		SSE	488.02 ± 0.00	488.02	7.14	488.02 ± 0.00	488.02	9.13	488.02 ± 0.00	488.02	8.16	489.10 ± 1.79	488.02		0.00 %	0.00 %
Dataset 3	6	DBI	$\mathbf{0.3555 \pm 0.000}$	0.3555	13.58	0.3817 ± 0.040	0.3555	16.38	0.3729 ± 0.035	0.3555	15.74	0.5555 ± 0.104	0.3555	2.90	0.00 %	**4.68 %**
		VRC	2713.65 ± 0.00	2713.65	9.01	2713.65 ± 0.00	2713.65	10.42	2691.31 ± 44.68	2713.65	9.42	1562.57 ± 657.82	2713.65		0.00 %	0.00 %
		SSE	543.17 ± 0.00	543.17	7.75	543.17 ± 0.00	543.17	7.79	543.17 ± 0.00	543.17	8.74	1057.01 ± 341.81	543.17		0.00 %	0.00 %
Dataset 4	10	DBI	$\mathbf{0.6082 \pm 0.034}$	**0.5499**	21.45	0.6598 ± 0.053	0.5873	28.34	0.6638 ± 0.064	0.5873	22.96	0.7309 ± 0.055	0.6692	6.97	**6.37 %**	**7.83 %**
		VRC	1742.76 ± 0.00	1742.76	15.38	1742.76 ± 0.00	1742.76	21.33	1742.76 ± 0.00	1742.76	16.84	1059.88 ± 188.35	1351.53		0.00 %	0.00 %
		SSE	1623.84 ± 0.00	1623.84	12.28	1623.84 ± 0.00	1623.84	14.94	1623.84 ± 0.00	1623.84	14.86	2690.93 ± 427.69	2075.70		0.00 %	0.00 %

tion, CPSOII performed better than CPSOI, CGA and K-means in all of the three used criteria significantly. For example, as shown in Table 8, the values of IR_{DBI} for Libras Movement and Hepato data sets are 8.44 % and 7.19 %, respectively. Also, this value for Hepato using Guided & Unguided initialization (Table 9) is 9.26 %.

Figure 12 shows the IR values (IR_{DBI}, IR_{VRC} and $\mathrm{IR}_{\mathrm{SSE}}$) of CPSOII in the best and average values of three used criteria with both the Guided and Unguided initialization.

5.5.3 Convergence test

The speed of convergence and the quality of solution are two important problems in the optimization algorithms. These problems are dependent on several factors, such as algorithm type and heuristics used in various steps of the algorithm. CPSOII as an extension of PSO algorithm inherits the fast convergence property of PSO. Additionally, using the renumbering procedure and the Guided initialization increase the speed of convergence.

Experimental results show that the convergence of CPSOII toward the qualified solutions, in both cases of known and unknown number of clusters, is faster than the other compared algorithms. Figure 13 shows the convergence toward the qualified solutions for CPSOII, CPSOI and CGA algorithms using the Unguided initialization and based on the DBI criterion. Based on the curves shown in Fig. 13, the convergence of CPSOII toward the qualified solutions in all of six real-world data sets is faster than CPSOI and CGA. The difference of the convergence in Libras Movement data set is more than others. The Libras Movement data set is a high dimensional one (with 90 features) that makes partitioning of its data objects more complex. The results shown in Fig. 13(d) indicate that CPSOII for complex data sets achieves far better performance and convergence than CGA and CPSOI.

The Guided initialization as a heuristic improves the speed of convergence and the quality of solutions. Figure 14 shows the effect of the Guided initialization in the convergence of CPSOII toward qualified solutions. The results show that by using the Guided initialization, the number of iterations for finding better solutions is reduced and therefore, the speed of algorithms is increased. As mentioned above, KCPSO uses K-means in the swarm initialization phase, so the convergence of it is compared with CPSOII only using Guided & Unguided initialization (Fig. 15).

5.5.4 Effects of renumbering procedure and various similarity functions

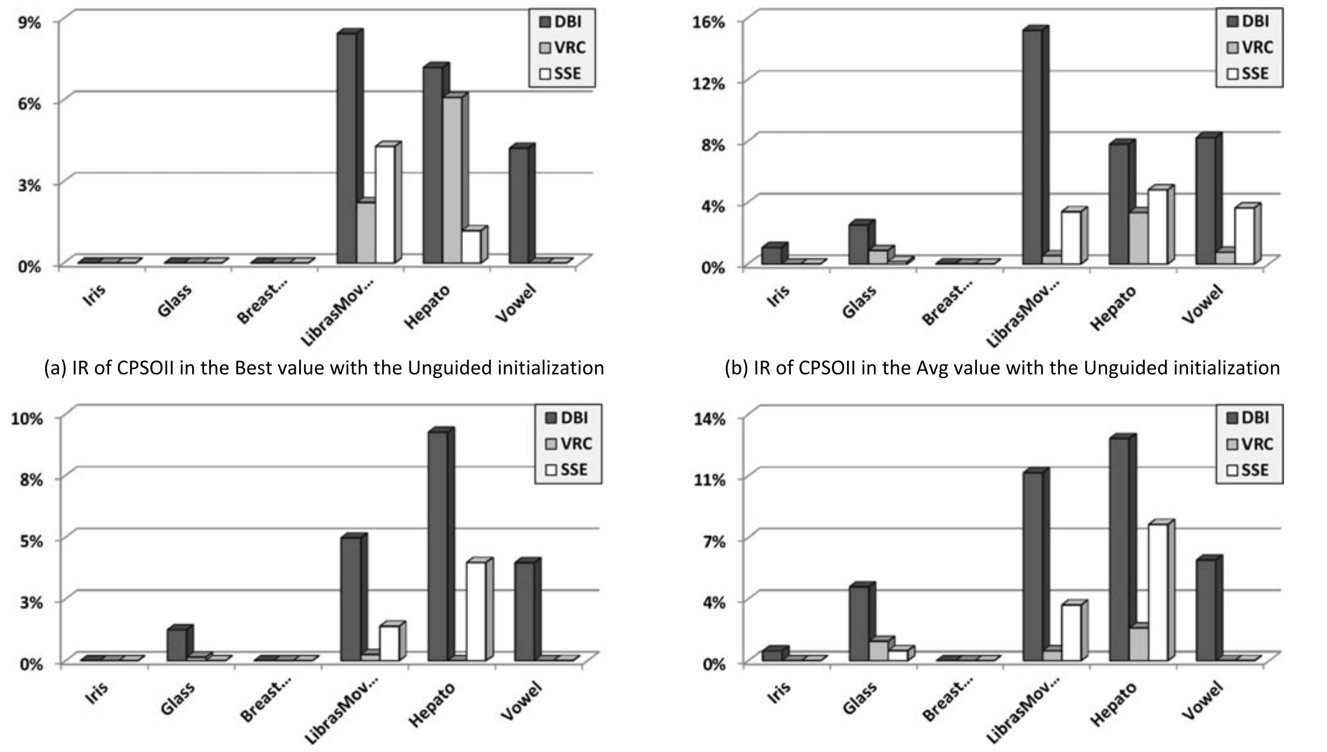
Population diversity is a way to monitor the degree of the convergence or divergence in PSO search process [38]. Sev-

Table 8 Experimental results for real-world data sets using the Unguided initialization and known number of clusters

Dataset	K	Crite-rion	CPSOII			CPSOI			CGA			K-Means			IR of CPSOII in Best	IR of CPSOII in Avg
			Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)		
Iris	3	DBI	**0.4495 ±0.0183**	0.4320	12.31	0.4544 ± 0.0312	0.4320	18.82	0.4994 ± 0.0812	0.4320	14.30	0.7386 ± 0.1261	0.6623	2.39	0.00 %	**1.08 %**
		VRC	561.63 ± 0.00	561.63	8.72	561.63 ± 0.00	561.63	11.93	561.63 ± 0.00	561.63	9.08	514.90 ± 83.90	561.63		0.00 %	0.00 %
		SSE	78.94 ± 0.00	78.94	6.58	78.94 ± 0.00	78.94	8.75	78.94 ± 0.00	78.94	8.23	87.79 ± 19.40	78.94		0.00 %	0.00 %
Glass	6	DBI	**0.8015 ±0.0983**	0.6142	32.76	0.8224 ± 0.1226	0.6142	45.54	0.8541 ± 0.1003	0.6524	59.95	1.1423 ± 0.1237	0.9351	12.02	0.00 %	**2.54 %**
		VRC	**121.67 ±5.13**	124.62	27.30	120.18 ± 6.37	124.62	33.83	120.60 ± 5.37	124.62	38.37	104.33 ± 18.88	122.62		0.00 %	**0.88 %**
		SSE	**343.94 ±12.80**	336.06	23.68	344.47 ± 14.00	336.06	27.26	352.65 ± 26.48	336.06	30.72	399.66 ± 71.39	338.27		0.00 %	**0.15 %**
Breast Cancer	2	DBI	**0.7570 ±0.0000**	0.7570	104.98	0.7571 ± 0.0002	0.7570	138.65	0.7571 ± 0.0002	0.7570	146.50	0.7573 ± 0.0000	0.7573	2.45	0.00 %	**0.02 %**
		VRC	1026.26 ± 0.00	1026.26	69.52	1026.26 ± 0.00	1026.26	86.90	1026.26 ± 0.00	1026.26	89.16	998.03 ± 32.27	1026.26		0.00 %	0.00 %
		SSE	19323 ± 0.00	19323	53.60	19323 ± 0.00	19323	65.32	19323 ± 0.00	19323	67.30	19363.27 ± 33.64	19323.17		0.00 %	0.00 %
Libras Movement	15	DBI	**1.1012 ±0.0545**	**1.0383**	604.04	1.2983 ± 0.0613	1.1340	761.52	1.3011 ± 0.0646	1.1609	822.61	1.4852 ± 0.0896	1.3829	60.14	**8.44 %**	**15.18 %**
		VRC	**53.26** ± 2.37	**56.26**	414.05	52.97 ± **1.91**	55.01	513.95	52.36 ± 2.46	54.93	568.91	47.87 ± 3.04	50.95		**2.22 %**	**0.54 %**
		SSE	**320.69** ± 10.13	**307.41**	333.06	331.94 ± 5.27	321.18	406.09	341.59 ± 6.68	331.14	397.23	352.18 ± 5.16	348.00		**4.29 %**	**3.39 %**
Hepato	4	DBI	**0.8656** ± 0.0747	**0.7965**	91.63	0.9389 ± 0.0675	0.8582	152.28	0.9548 ± **0.0526**	0.8816	145.72	1.1969 ± 0.0610	1.1043	6.61	**7.19 %**	**7.80 %**
		VRC	**331.57** ± 10.54	**342.21**	55.82	320.54 ± **1.29**	321.38	72.79	319.62 ± 4.25	321.38	85.58	312.56 ± 10.05	318.03		**6.09 %**	**3.33 %**
		SSE	**54309455 ±2167725**	**52700335**	53.31	57058951 ± 4922656	53340995	60.14	57388462 ± 4125856	53334853	62.17	63825608 ± 4777769	55705292		**1.19 %**	**4.82 %**
Vowel	6	DBI	**0.7690** ± 0.0898	**0.6467**	112.11	0.8378 ± 0.0763	0.6752	156.61	0.8437 ± **0.0663**	0.7054	173.14	1.0917 ± 0.1169	0.9447	19.31	**4.23 %**	**8.22 %**
		VRC	**1456.34 ±13.03**	1465.82	68.02	1445.12 ± 26.06	1465.64	81.76	1423.98 ± 62.51	1465.82	80.70	1396.40 ± 39.85	1428.41		0.00 %	**0.77 %**
		SSE	**31029689 ±603829**	30689190	56.46	32199653 ± 1359496	30689190	69.53	32769548 ± 1881098	30747988	67.12	34160690 ± 3216732	31404253		0.00 %	**3.63 %**

Table 9 Experimental results for real-world data sets using the Guided & Unguided initialization and known number of clusters

Dataset	K	Crite-rion	CPSOII			CPSOI			CGA			IR of CPSOII in Best	IR of CPSOII in Avg
			Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)		
Iris	3	DBI	**0.4387 ± 0.0116**	0.4320	5.56	0.4412 ± 0.0167	0.4320	7.36	0.4410 ± 0.0240	0.4320	7.17	0.00 %	**0.52 %**
		VRC	561.63 ± 0.00	561.63	4.42	561.63 ± 0.00	561.63	6.22	561.63 ± 0.00	561.63	5.38	0.00 %	0.00 %
		SSE	78.94 ± 0.00	78.94	4.26	78.94 ± 0.00	78.94	5.11	78.94 ± 0.00	78.94	6.09	0.00 %	0.00 %
Glass	6	DBI	**0.7564 ± 0.0724**	**0.6211**	23.84	0.7957 ± 0.0955	0.6301	29.98	0.7895 ± 0.0913	0.6290	33.56	**1.26 %**	**4.19 %**
		VRC	**124.21 ± 0.79**	**124.62**	17.33	122.87 ± 1.42	124.44	19.54	122.17 ± 2.55	124.26	22.25	**0.14 %**	**1.08 %**
		SSE	**337.53 ± 2.69**	336.06	15.63	340.79 ± 7.42	336.06	17.97	339.37 ± 6.13	336.06	21.05	0.00 %	**0.54 %**
Breast Cancer	2	DBI	0.7570 ± 0.0000	0.7570	36.26	0.7570 ± 0.0000	0.7570	48.07	0.7570 ± 0.0000	0.7570	51.56	0.00 %	0.00 %
		VRC	1026.26 ± 0.00	1026.26	30.69	1026.26 ± 0.00	1026.26	39.05	1026.26 ± 0.00	1026.26	42.63	0.00 %	0.00 %
		SSE	19323 ± 0.00	19323	31.75	19323 ± 0.00	19323	37.39	19323 ± 0.00	19323	42.02	0.00 %	0.00 %
Libras Movement	15	DBI	**1.0886 ± 0.0626**	**1.0216**	488.25	1.2187 ± 0.0642	1.0750	567.41	1.2397 ± 0.0741	1.1196	758.05	**4.97 %**	**10.68 %**
		VRC	**56.10 ± 0.09**	**56.17**	232.92	55.80 ± 0.32	56.03	286.32	54.70 ± 0.67	55.43	272.62	**0.25 %**	**0.53 %**
		SSE	**309.64 ± 0.41**	**309.05**	174.41	319.72 ± 1.82	317.37	225.98	323.00 ± 6.74	313.45	257.18	**1.40 %**	**3.15 %**
Hepato	4	DBI	**0.5296 ± 0.1404**	**0.3295**	32.15	0.6132 ± 0.1409	0.3843	44.31	0.6061 ± 0.1198	0.3631	47.39	**9.26 %**	**12.63 %**
		VRC	**337.94 ± 8.28**	342.07	23.63	331.72 ± 10.36	342.07	39.05	328.91 ± 10.83	342.07	39.33	0.00 %	**1.84 %**
		SSE	**50600720 ± 0.00**	**50600720**	29.70	55336805 ± 4175056	52700335	34.41	54836231 ± 2469538	52700335	36.07	**3.98 %**	**7.72 %**
Vowel	6	DBI	**0.7218** ± 0.0545	**0.6497**	70.97	0.7656**±0.0449**	0.6921	86.64	0.7765 ± 0.0562	0.6766	87.62	**3.98 %**	**5.72 %**
		VRC	**1465.84 ± 0.03**	1465.88	39.72	1465.70 ± 0.19	1465.88	48.57	1465.53 ± 0.11	1465.74	45.49	0.00 %	**0.01 %**
		SSE	**30688629 ± 1076.96**	**30687310**	34.02	30691787 ± 6838.2	30689508	41.72	30694426 ± 14755.2	30689508	42.74	**0.01 %**	**0.01 %**



(a) IR of CPSOII in the Best value with the Unguided initialization

(b) IR of CPSOII in the Avg value with the Unguided initialization

(c) IR of CPSOII in the Best value with the Guided&Unguided initialization

(d) IR of CPSOII in the Avg value with the Guided&Unguided initialization

Fig. 12 The IR values of CPSOII in the Best and Avg values of DBI, VRC and SSE criteria with both the Guided and Unguided initialization

eral definitions of PSO population diversity measurements have been proposed in the literature [38]. There are different ways of introducing diversity and controlling its degree. Random restart and controlling swarm size are simple techniques for this purpose. Norouzzadeh et al. [39] used a random restart technique for injecting diversity to swarm. Clerc [6] and Zhang et al. [40] changed the size of swarm according to the performance of the algorithm dynamically. The size of swarm is important, because too few particles will cause the algorithm to converge prematurely to a local optima, while too many particles will slow down the algorithm [41]. As described in Sect. 3.1.1, the integer encoding scheme is naturally redundant. One of the disadvantages of redundant particles is that it reduces the size of swarm because there are more than one particle that represent the same solution. Thus, this problem can reduce the diversity of PSO algorithm. In the CPSOII algorithm, using the renumbering procedure is helpful for solving this problem. Hence, removing redundant particles in CPSOII increases the diversity of population and the speed of convergence appropriately. Table 10 shows the effect of similarity functions on the Rate of Distinct Particles (RDP) and clustering results. In this table, RDP shows the average rate of distinct particles during the evolution process and computed by (20) at the end of each iteration.

$$RDP = \sum_{t=1}^{MaxIteration} \left(1 - \frac{(NOIdentical_t + NORedundant_t)}{SwarmSize}\right) \times 100 \tag{20}$$

Where $NOIdentical_t$ and $NORedundant_t$ are the number of identical and redundant particles at iteration t, respectively. As shown in Table 10 and as expected, when the renumbering procedure is used, it appears that the rate of distinct particles is increased, and the average DBI is improved. Among the various similarity functions, Sokal & Sneath2 similarity function obtains better the average DBI and RDP in most cases. In addition, for some data sets, Sokal & Sneath1 and Rogers & Tanimoto similarity functions achieve suitable results. Based on these results, it is concluded that similarity functions and renumbering procedure play key roles in the evolutionary process and when they are used, the movement toward the correct direction is facilitated. Figure 16 illustrates RDP in CPSOII, CPSOI and CGA during the evolution process. KCPSO uses real encoding, so it has not been considered in this experiment.

Based on the results shown in Table 10 and Fig. 16, the number of distinct particles in CPSOII with the renumbering procedure is higher than CPSOI and CGA, thus, CPSOII is able to explore more regions. Moreover, it is concluded that the rate of distinct particles is dependent upon the number of objects in data sets, because for data sets with more objects, the number of different permutations of

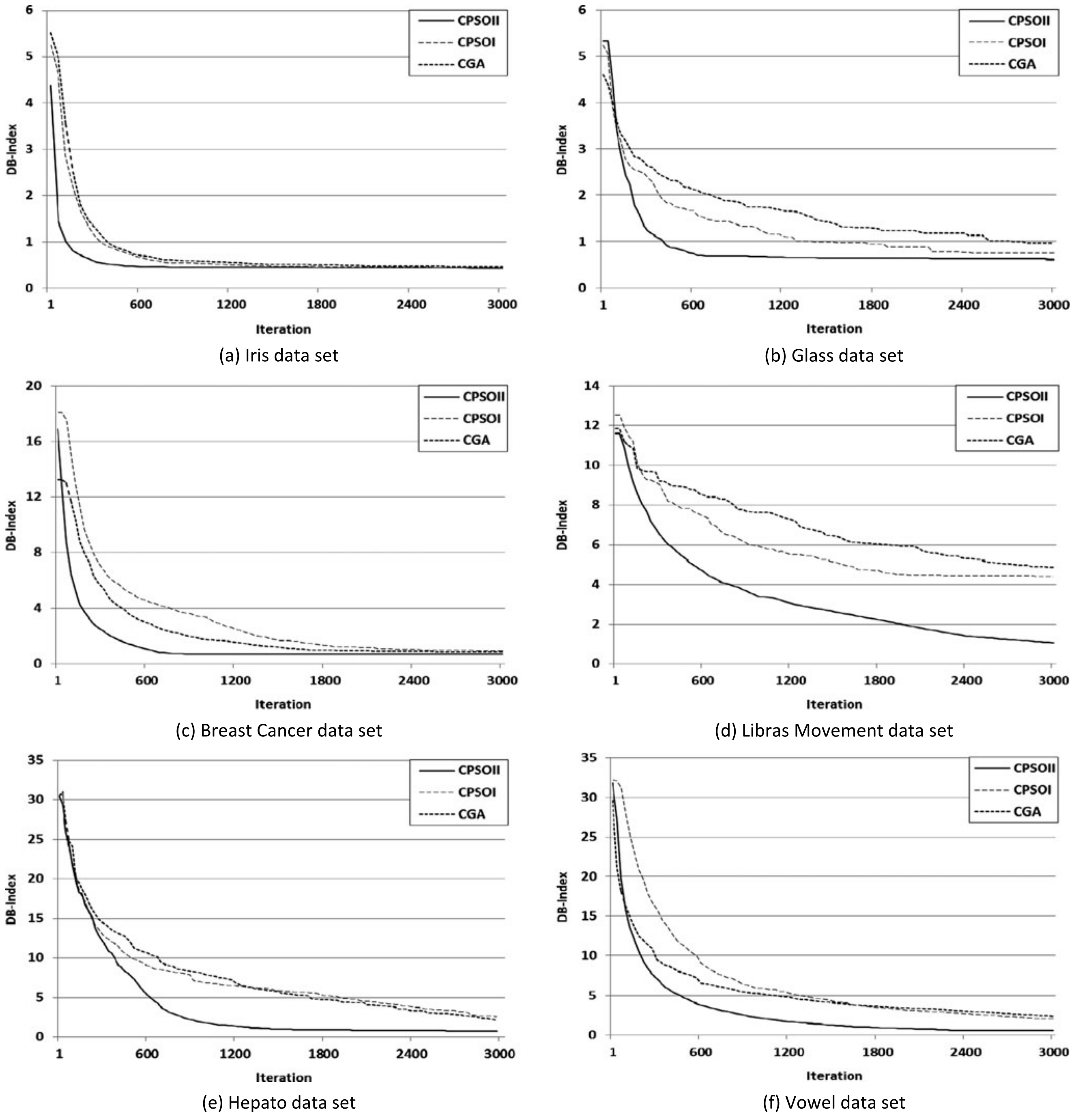
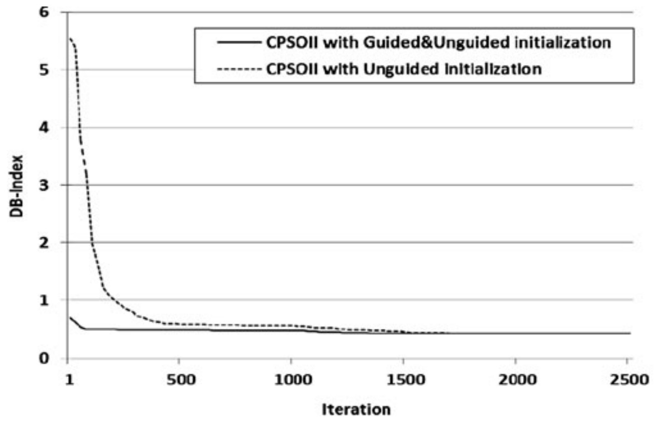
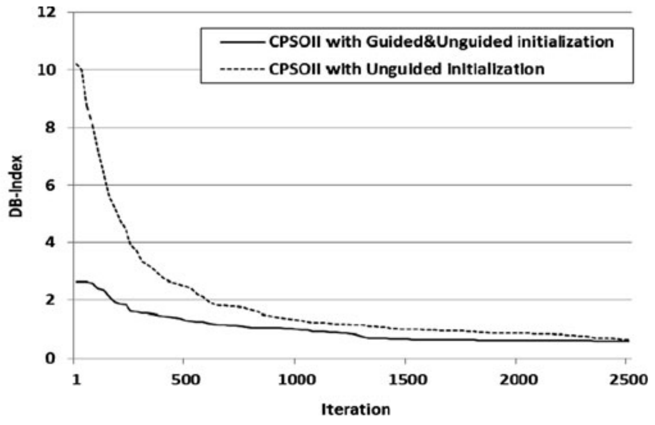


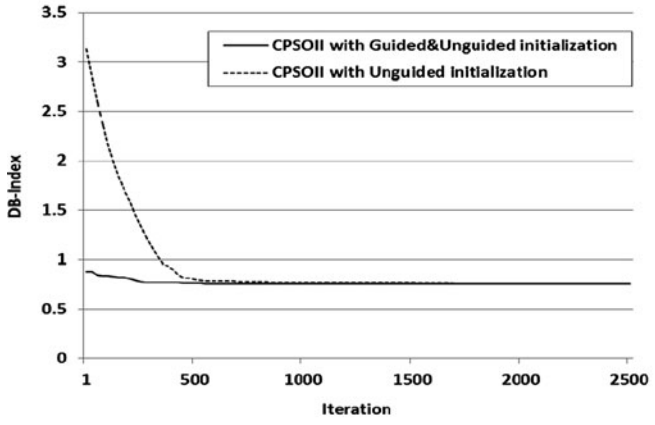
Fig. 13 Comparing convergence of CPSOII, CPSOI and CGA algorithms toward qualified solutions using the Unguided initialization (using the DBI criterion)



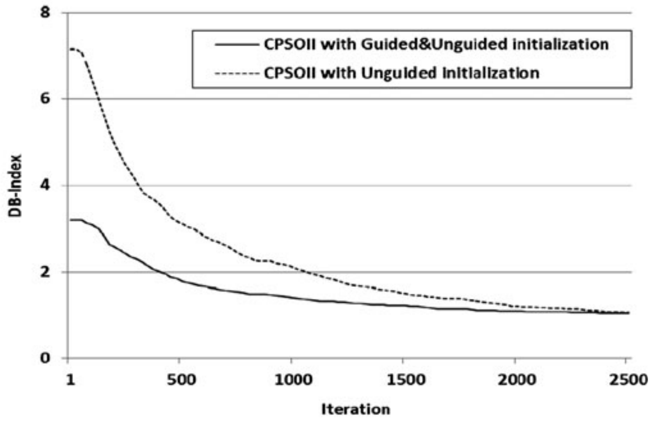
(a) Iris data set



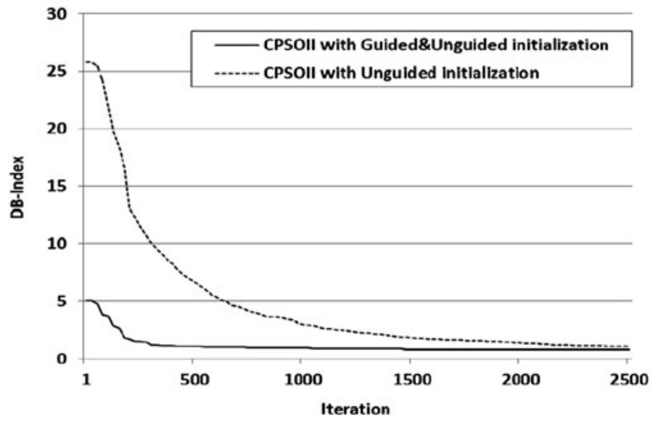
(b) Glass data set



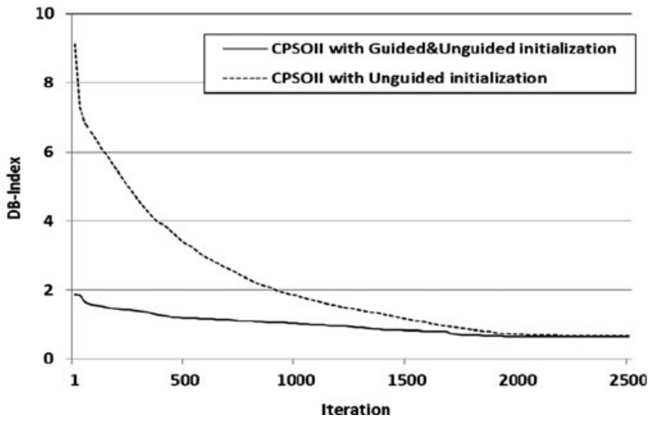
(c) Breast Cancer data set



(d) Libras Movement data set



(e) Hepato data set



(f) Vowel data set

Fig. 14 The effect of the Guided initialization in the convergence of CPSOII toward qualified solutions (using the DBI criterion)

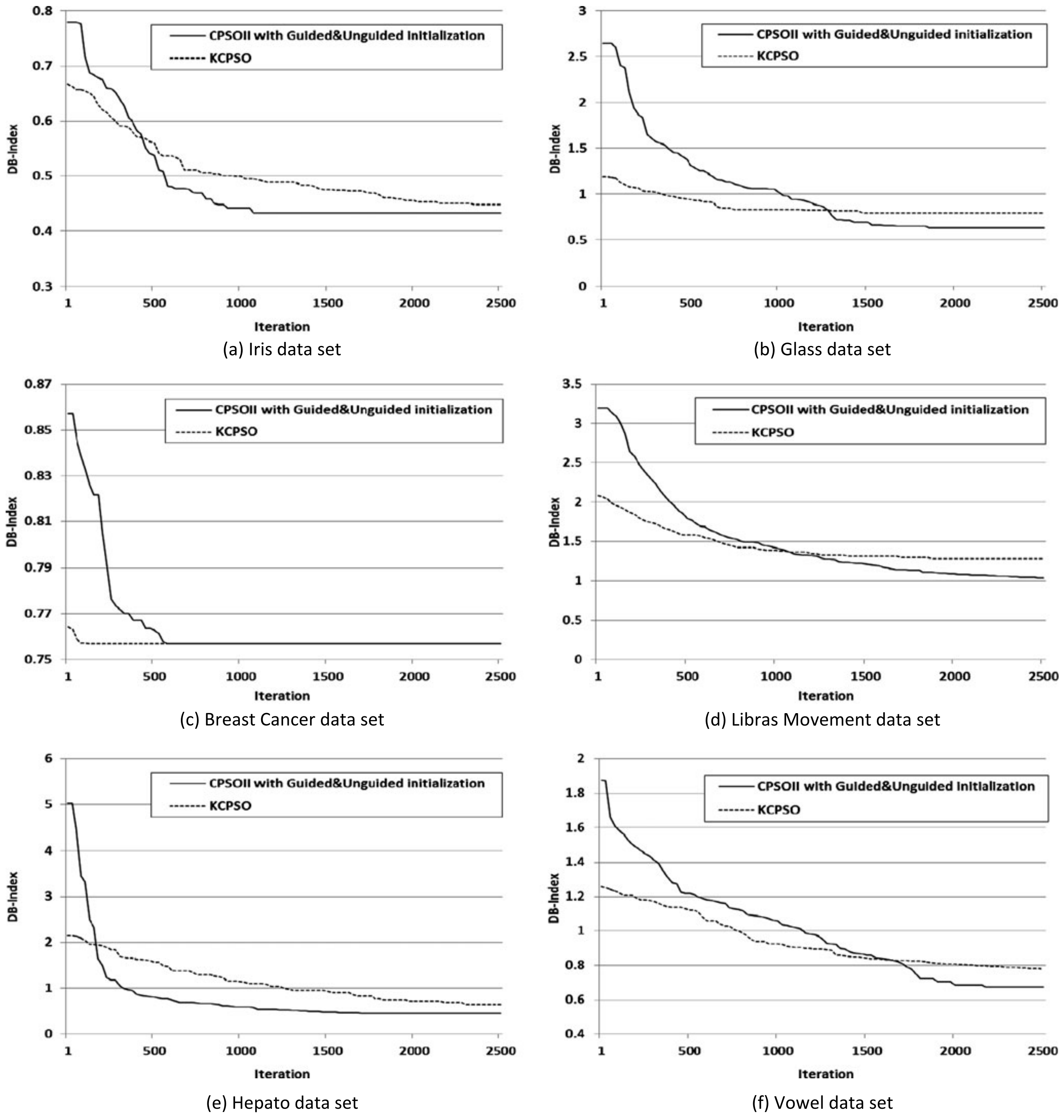


Fig. 15 Comparing convergence of KCPSO and CPSOII with the Guided & Unguided initialization toward qualified solutions (using the DBI criterion)

Table 10 The effect of similarity functions on the RDP value and the clustering results of CPSOII

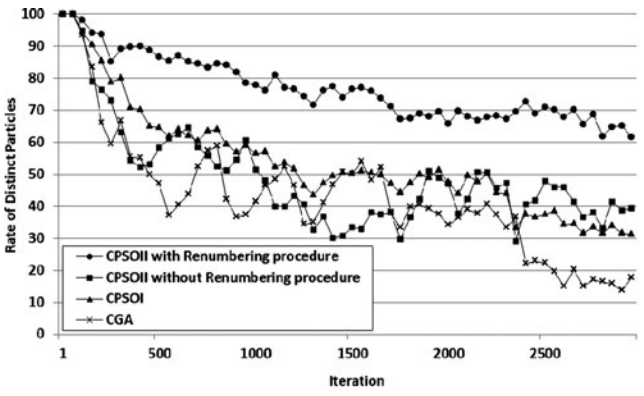
Similarity Function	Dataset											
	Iris		Glass		Breast Cancer		Libras Movement		Hepato		Vowel	
	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP
Simple Matching	0.4393	69.24 %	0.7682	90.39 %	0.7570	99.56 %	1.1425	96.28 %	0.5475	100 %	0.7299	100 %
Jaccard	0.4395	76.15 %	0.7667	92.26 %	0.7570	100 %	1.1337	96.06 %	0.5327	98.86 %	0.7303	100 %
Sokal & Sneath1	0.439	75.29 %	0.7617	**93.37 %**	0.7570	99.99 %	**1.0886**	97.50 %	0.5304	100 %	**0.7218**	100 %
Sokal & Sneath2	0.4391	**76.87 %**	**0.7564**	93.26 %	0.7570	100 %	1.0924	**98.73 %**	**0.5296**	99.99 %	0.7237	100 %
Dice	0.4391	69.38 %	0.7674	87.94 %	0.7570	100 %	1.1274	93.19 %	0.5465	99.27 %	0.7312	99.99 %
Rogers & Tanimoto	**0.4387**	72.09 %	0.7572	92.75 %	0.7570	99.99 %	1.1187	97.64 %	0.5301	100 %	0.7265	100 %
NEI & LI	0.4402	67.98 %	0.7591	92.03 %	0.7570	100 %	1.1262	96.33 %	0.5472	99.98 %	0.7276	100 %
Without Renumbering procedure	0.4406	49.32 %	0.7837	75.28 %	0.7570	95.01 %	1.2089	86.47 %	0.5913	93.19 %	0.7610	96.59 %

Table 11 Comparing the rate of distinct particles and the average DBI in CPSOII, CPSOI and CGA algorithms

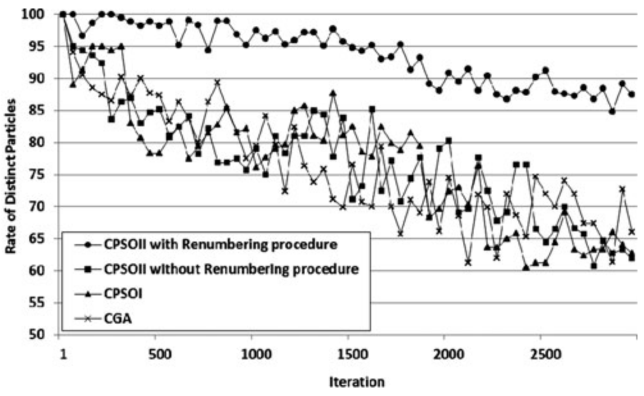
Algorithm	Dataset											
	Iris		Glass		Breast Cancer		Libras Movement		Hepato		Vowel	
	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP	Avg DBI	RDP
CPSOII	**0.4387**	**76.87 %**	**0.7564**	**93.37 %**	0.7570	**100 %**	**1.0886**	**98.73 %**	**0.5296**	**100 %**	**0.7218**	**100 %**
CPSOI	0.4412	53.69 %	0.7957	74.60 %	0.7570	94.29 %	1.2187	87.51 %	0.6132	89.25 %	0.7656	95.80 %
CGA	0.4410	45.09 %	0.7895	74.17 %	0.7570	92.26 %	1.2397	82.38 %	0.6061	87.98 %	0.7765	94.61 %

Table 12 Description of the used data sets for scalability test

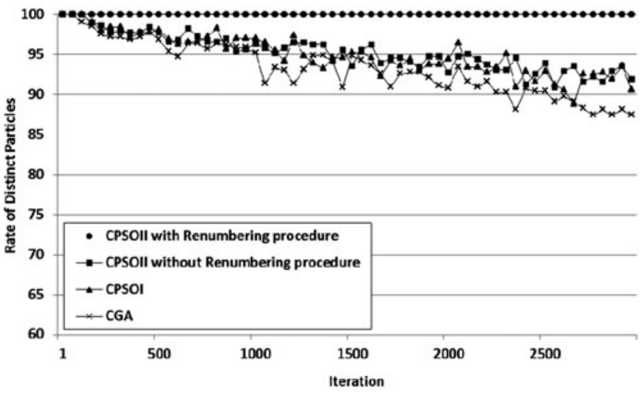
Data Set	# Data objects	# Features	# Classes	Size of Classes
Image Segmentation [37]	2310	19	7	Size of each class is 330
Page Blocks Classification [37]	5473	10	5	4913, 329, 28, 88, 115
Statlog (Landsat Satellite) [37]	6435	36	7	1533, 703, 1358, 626, 707, 0, 1508



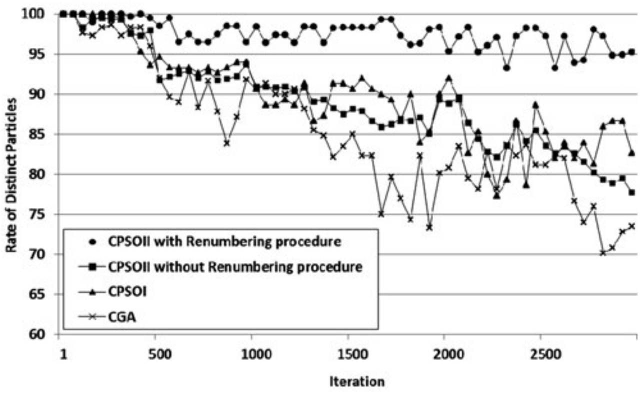
(a) Iris data set



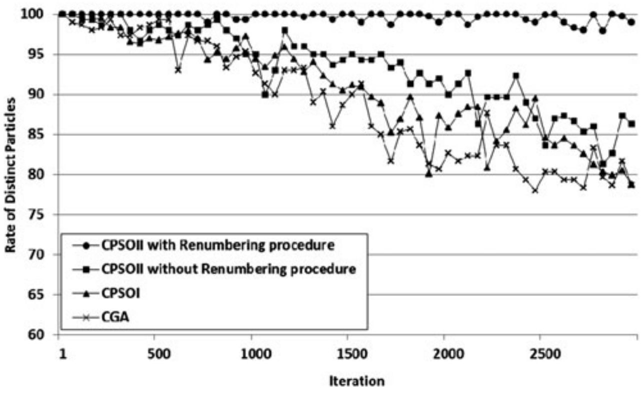
(b) Glass data set



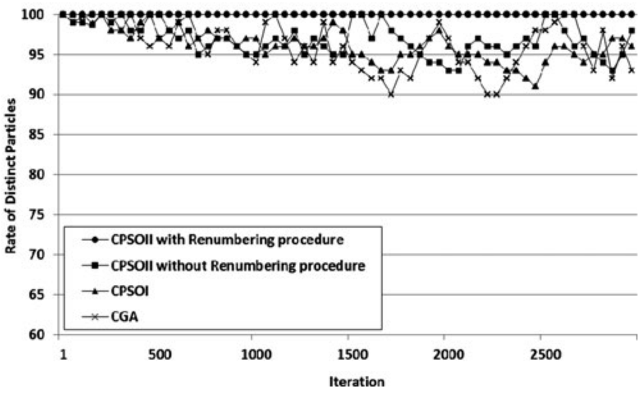
(c) Breast Cancer data set



(d) Libras Movement data set



(e) Hepato data set



(f) Vowel data set

Fig. 16 Comparing the rate of distinct particles in CPSOII, CPSOI and CGA algorithms during the evolution process

objects is high and thus, the probability of generating two particles with the same or redundant clustering solution is low. On the other hand, converging to the global best solution leads to a decrease in the value of RDP, because most of the particles move to the global best region and the probability of generating two particles with the same or redundant clustering solution in a small region is higher than a large region. The RDP of CPSOII has been compared with CPSOI and CGA algorithms and the results are illustrated in Table 11. Based on these results, the CPSOII algorithm has a better average DBI, and also the average RDP during the evaluation process is higher than CPSOI and CGA.

5.5.5 Scalability discussion

It should be noted that a major limitation of evolutionary clustering techniques in comparison to traditional clustering techniques, such as K-Means, is that they are too time consuming. There is a trade-off between time and quality in clustering techniques. Traditional clustering techniques sacrifice the quality of clustering in order to decrease the process time, while evolutionary clustering techniques prefer to achieve better clustering results by consuming more time. To better illustrate this point, we used three large data sets (introduced in Table 12) and compared CPSOII, CPSOI, CGA and K-means algorithms. The results of these experiments are shown in Table 13.

The results indicate that although CPSOII, CPSOI and CGA consume more time, their results are considerably better than K-means.

Combining evolutionary clustering techniques with traditional clustering techniques is a solution to reduce the runtime of evolutionary clustering techniques. Guided & Unguided initialization is a simple example of this combination. Comparing the results of Unguided initialization with Guided & Unguided initialization in Tables 5, 6, 8 and 9 shows that the average time of algorithms with Guided & Unguided initialization is better than Unguided initialization.

6 Conclusions

Partitional clustering methods attempt to partition the data objects into a set of disjoint clusters directly. In this paper, we proposed a combinatorial particle swarm optimization for dynamic data clustering (CPSOII). CPSOII finds the best number of clusters automatically and partitions the data objects into clusters effectively. Compared with other PSO based clustering algorithms, such as CPSOI, KCPSO and DCPSO, CPSOII operators are simple and effective. CPSOII uses both the guided and unguided initialization to

Table 13 Experimental results for scalability test using the Unguided initialization and known number of clusters

Dataset	K	Crite-rion	CPSOII			CPSOI			CGA			K-Means			IR of CPSOII in Best	IR of CPSOII in Avg
			Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)	Avg ± SD	Best	Avg Time (s)		
Image Segmentation	7	DBI	**0.9795 ± 0.1126**	**0.8110**	768.90	1.1219 ± 0.0867	0.9977	932.80	1.1876 ± 0.0824	1.0137	1288.07	1.4217 **±0.0579**	1.2930	79.70	**18.72 %**	**12.69 %**
		VRC	**930.71** ± 202.83	**1104.83**	550.57	882.80 ± 239.69	1100.83	659.43	840.62 ± 225.33	1098.01	671.58	633.60 **±135.22**	970.33		**0.36 %**	**5.15 %**
		SSE	**19598166.43** ± 4036150.71	**13404115.28**	472.55	20544036.28 ± 4555904.30	13920368.28	510.30	20794920.93 ± 4844073.83	13740602.28	554.26	26477323.46 **±1246377.02**	23006387.52		**2.45 %**	**4.60 %**
Page Blocks	5	DBI	**0.3145** ± 0.1108	**0.2350**	938.88	0.4126 ± 0.1046	0.3174	1410.90	0.5500 ± 0.2235	0.2863	1861.11	0.9547 ± 0.1026	0.7971	77.41	**17.93 %**	**23.76 %**
		VRC	**5149.74** ± 6372.76	**14884.11**	735.19	4526.1033 ± 4142.00	14325.45	907.28	4762.52 ± 4757.05	13029.41	961.39	2703.16 **±2320.26**	6078.65		**3.75 %**	**7.52 %**
		SSE	**68117559364** ± 35971966734	**13218377613**	544.19	83772242093 **±32159155893**	18371967630	683.41	85844663719 ± 51793671581	15725891172	692.36	136520379831 ± 86500749027	69210854201		**15.95 %**	**18.69 %**
Statlog	6	DBI	**0.8957 ±0.1768**	**0.7150**	4664.57	1.2493 ± 0.1804	0.9801	5563.25	1.3891 ± 0.2352	1.0492	6612.30	2.2068 ± 0.3542	1.8527	531.73	**27.05 %**	**28.30 %**
		VRC	**3206.61** ± 1269.58	**4869.39**	3109.46	2415.1938 ± 1110.92	3927.09	3979.97	2315.87 ± 1320.38	3762.59	4026.19	1261.74 **±598.88**	2065.85		**19.35 %**	**24.68 %**
		SSE	**20641048.33 ±2629835.93**	**16261071.58**	2842.68	25351547.30 ± 2666957.35	21083910.78	3376.83	26389767.82 ± 4332975.66	22793416.19	3889.76	60107507.78 ± 22714800.10	31951849.07		**22.87 %**	**18.58 %**

generate an initial swarm of particles. Experimental results (presented in Sect. 5.5.3) revealed that the guided initialization improved the speed of convergence and the quality of solution in most cases. However, in some cases, the best value obtained by algorithm with the guided initialization was worse than algorithm with the unguided initialization.

CPSOII uses the renumbering procedure as a preprocessing, before computing the velocity of each particle. This procedure removes redundant particles and consequently increases the rate of distinct particles and the diversity of population. In addition, it makes the CPSOII algorithm insensitive to different encodings of clustering solutions. This is an important advantage and increases the speed of convergence and the quality of solutions, because particles are consciously moved in this case. In this paper, we used seven different binary similarity functions for the renumbering procedure. The effects of these functions were tested and results revealed that the Sokal & Sneath2 similarity function obtained better average DBI and the rate of distinct particles in most cases.

The performance of CPSOII is evaluated with both artificial and real-world data sets and also with the consideration of three clustering metrics, SSE, VRC and DBI. Comparing the obtained results of CPSOII with CPSO [7], KCPSO [8], CGA [12] and K-means algorithms revealed that CPSOII yielded promising results, for example, it improved 9.26 % of the value of DBI criterion for Hepato data set. The obtained results for unknown number of clusters showed that CPSOII with the guided initialization achieved better average DBI for most data sets. Also, the obtained average K (number of clusters) with CPSOII is the closest number to the number of classes of data sets in comparison to KCPSO and CGA.

In future work, we intend to use Multi-Objective Particle Swarm Optimization (MOPSO) to dynamic data clustering to improve the performance of the proposed method. Moreover, we are going to use guided and unguided mutation and population topology to improve the results.

References

1. Pedrycz W (2005) Knowledge-based clustering. Wiley, New York. doi:10.1002/0471708607.fmatter
2. Frigui H, Krishnapuram R (1999) A robust competitive clustering algorithm with applications in computer vision. IEEE Trans Pattern Anal Mach Intell 21(5):450–465
3. Xu R, Wunsch DC (2010) Clustering algorithms in biomedical research: a review. IEEE Rev Biomed Eng 3(1):120–154
4. Niknam T, Amiri B, Olamaei J, Arefi A (2009) An efficient hybrid evolutionary optimization algorithm based on PSO and SA for clustering. J Zhejiang Univ Sci 10(4):512–519. doi:10.1631/jzus.A0820196
5. Papadimitriou CH, Steiglitz K (1998) Combinatorial optimization: algorithms and complexity. Dover, New York
6. Clerc M (2006) Particle swarm optimization. Wiley-ISTE, New York
7. Jarboui B, Cheikh M, Siarry P, Rebai A (2007) Combinatorial particle swarm optimization (CPSO) for partitional clustering problem. Appl Math Comput 192(2):337–345. doi:10.1016/j.amc.2007.03.010
8. Yucheng K, Szu-Yuan, L (2009) Combining K-means and particle swarm optimization for dynamic data clustering problems. In: IEEE international conference on intelligent computing and intelligent systems (ICIS), 20–22 Nov. 2009, pp 757–761
9. 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
10. Omran M, Salman A, Engelbrecht (2005) A dynamic clustering using particle swarm optimization with application in unsupervised image classification. In: 5th world enformatika conference (ICCI 2005), Prague, Czech Republic, Citeseer, pp 199–204
11. Shin K, Jeong Y-S, Jeong M (2012) A two-leveled symbiotic evolutionary algorithm for clustering problems. Appl Intell 36(4):788–799. doi:10.1007/s10489-011-0295-y
12. Hruschka ER, Campello RJGB, de Castro LN (2006) Evolving clusters in gene-expression data. Inf Sci 176(13):1898–1927. doi:10.1016/j.ins.2005.07.015
13. Hruschka ER, Ebecken NF (2003) A genetic algorithm for cluster analysis. Intell Data Anal 7(1):15–25
14. Ma PCH, Chan KCC, Yao X, Chiu DKY (2006) An evolutionary clustering algorithm for gene expression microarray data analysis. IEEE Trans Evol Comput 10(3):296–314
15. Özyer T, Zhang M, Alhajj R (2011) Integrating multi-objective genetic algorithm based clustering and data partitioning for skyline computation. Appl Intell 35(1):110–122. doi:10.1007/s10489-009-0206-7
16. Özyer T, Alhajj R (2009) Parallel clustering of high dimensional data by integrating multi-objective genetic algorithm with divide and conquer. Appl Intell 31(3):318–331. doi:10.1007/s10489-008-0129-8
17. http://www.isical.ac.in/~sushmita
18. Karthi R, Arumugam S, Rameshkumar K (2008) Comparative evaluation of particle swarm optimization algorithms for data clustering using real world data sets. Int J Comput Sci Netw Secur 8(1):203–212
19. Bandyopadhyay S, Maulik U (2002) Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognit 35(6):1197–1208. doi:10.1016/s0031-3203(01)00108-x
20. Liu Y, Wu X, Shen Y (2011) Automatic clustering using genetic algorithms. Appl Math Comput 218(4):1267–1279. doi:10.1016/j.amc.2011.06.007
21. Karthi R, Arumugam S, Kumar K (2009) Discrete particle swarm optimization algorithm for data clustering. In: Nature inspired cooperative strategies for optimization (NICSO), pp 75–88
22. Latiff NM A, Tsimenidis CC, Sharif BS, Ladha C (2008) Dynamic clustering using binary multi-objective particle swarm optimization for wireless sensor networks. In: IEEE 19th international symposium on personal, indoor and mobile radio communications (PIMRC), 15–18 Sept. 2008, pp 1–5
23. Paoli A, Melgani F, Pasolli E (2009) Clustering of hyperspectral images based on multiobjective particle swarm optimization. IEEE Trans Geosci Remote Sens 47(12):4175–4188
24. 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. doi:10.1016/j.asoc.2009.07.001
25. Supratid S, Kim H (2009) Modified fuzzy ants clustering approach. Appl Intell 31(2):122–134. doi:10.1007/s10489-008-0117-z

26. Falkenauer E (1998) Genetic algorithms and grouping problems. Wiley, New York
27. Kennedy J, Eberhart, R (1995) Particle swarm optimization. In: IEEE international conference on neural networks. Nov/Dec, 1995, pp 1942–1948
28. Kao YT, Zahara E, Kao IW (2008) A hybridized approach to data clustering. Expert Syst Appl 34(3):1754–1762. doi:10.1016/j.eswa.2007.01.028
29. Premalatha K, Natarajan A (2009) A new approach for data clustering based on PSO with local search. Comput Inf Sci 1(4):139–145
30. Yang S, Li Y, Hu X, Pan R (2006) Optimization study on k-value of K-means algorithm. Syst Eng-Theory Pract, Inst China Syst Eng, Beijing 26(2):97–101
31. Parsopoulos KE, Vrahatis MN (2010) Particle swarm optimization and intelligence: advances and applications. Information Science Reference-Imprint of IGI Publishing
32. Choi S, Cha S, Tappert CC (2010) A survey of binary similarity and distance measures. Int J Syst Cybern Inform 8(1):43–48
33. Calinski T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat 3(1):1–27
34. Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1(2):224–227
35. Bandyopadhyay S, Maulik U (2001) Nonparametric genetic clustering: comparison of validity indices. IEEE Trans Syst Man Cybern, Part C, Appl Rev 31(1):120–125
36. Bandyopadhyay S Artificial data sets for data mining, available in http://www.isical.ac.in/~sanghami/data.html
37. UCI Repository of Machine Learning Databases retrieved from the World Wide Web: http://www.ics.uci.edu/~mlearn/MLRepository.html
38. Shi C, Yuhui S (2011) Diversity control in particle swarm optimization, Paper presented at the IEEE Symposium on Swarm Intelligence (SIS), 11–15 April 2011
39. Norouzzadeh M, Ahmadzadeh M, Palhang M (2011) LADPSO: using fuzzy logic to conduct PSO algorithm. Appl Intell 37(2):290–304
40. Zhang W, Liu Y, Clerc M (2003) An adaptive PSO algorithm for reactive power optimization. In: 6th international conference on advances in power control, operation and management, Hong Kong, pp 302–307
41. García-Villoria A, Pastor R (2009) Introducing dynamic diversity into a discrete particle swarm optimization. Comput Oper Res 36(3):951–966. doi:10.1016/j.cor.2007.12.001

Hamid Masoud received his M.Sc. degree in Software Engineering from Tarbiat Modares University, Tehran, Iran, in 2012, and the B.Sc. degree in Software Engineering from Tabriz University in 2008. His main research interests are computational intelligence, machine learning, object-oriented analysis and design, and Search Based Software Engineering (SBSE).

Saeed Jalili received Ph.D. in Computer Science from Bradford University, Bradford, in 1991. In 1992 he joined the School of Electrical and Computer Engineering at Tarbiat Modares University, Tehran, Iran. He is currently an associate professor at Tarbiat Modares University and his main research interests are unconventional computing, security protocol verification, network security, machine learning and software runtime verification.

Seyed Mohammad Hossein Hasheminejad is a Ph.D. candidate of computer engineering at Tarbiat Modares University (TMU). He received the M.Sc. degree in Software Engineering from TMU in 2009, and the B.Sc. degree in Software Engineering from Tarbiat Moalem University in 2007. His main research interests are formal methods for software engineering, object-oriented analysis and design, Search Based Software Engineering (SBSE), and self-adaptive systems.