Expert Systems with Applications 38 (2011) 14555–14563

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Chaotic particle swarm optimization for data clustering

Li-Yeh Chuang [a], Chih-Jen Hsiao [b], Cheng-Hong Yang [b,c,*]

[a] Department of Chemical Engineering, I-Shou University, Kaohsiung 80041, Taiwan
[b] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 80708, Taiwan
[c] Department of Network Systems, Toko University, Chiayi 61363, Taiwan

ARTICLE INFO

Keywords:
Data clustering
Chaotic map
Particle swarm optimization

ABSTRACT

Data clustering is a popular analysis tool for data statistics in several fields, including includes pattern recognition, data mining, machine learning, image analysis and bioinformatics, in which the information to be analyzed can be of any distribution in size and shape. Clustering is effective as a technique for discerning the structure of and unraveling the complex relationship between massive amounts of data. An improved technique which combines chaotic map particle swarm optimization with an acceleration strategy is proposed, since results of one of the most used clustering algorithm, K-means can be jeopardized by improper choices made in the initializing stage. Accelerated chaotic particle swarm optimization (ACPSO) searches through arbitrary data sets for appropriate cluster centers and can effectively and efficiently find better solutions. Comparisons of the clustering performance are obtained from tests conducted on six experimental data sets; the algorithms compared with ACPSO includes PSO, CPSO, K-PSO, NM-PSO, K-NM-PSO and K-means clustering. Results of the robust performance from ACPSO indicate that this method an ideal alternative for solving data clustering problem.

1. Introduction

Clustering analysis is a very popular data mining technique. It is the process of grouping a set of objects into clusters so that objects within the same cluster are similar to each other but are dissimilar to objects in other clusters (Han, Kamber, & Tung, 2001; Jain, Murty, & Flynn, 1999; Maimon & Rokach, 2005). When a set of objects has been applied to by a clustering algorithm, the resulting clusters can be used to reveal inherent structures resident in the data. The purpose of cluster analysis is to classify the clusters into groups holding that have some meaning in the context of a particular problem. More specifically, a set of patterns, usually represented by multi-dimensional vectors in a predefined space, are clumped into clusters based on their similarity. If the number of clusters, K, is known a prior clustering may be formulated as the distribution of n objects in an N-dimensional space among K groups in such a way that the objects in the same group are more similar with regard to certain criteria than those in different groups (Anderberg, 1973). This involves the minimization of some extrinsic optimization criteria.

Many clustering algorithms are based on evolutionary computation techniques, e.g., genetic algorithms (Bandyopadhyay & Maulik, 2002; Murthy & Chowdhury, 1996); however, particle swarm optimization is seldom chosen for solving clustering problem (Paterlini & Krink, 2006). The typical work flow of genetic algorithms starts with an initialization of a set of candidate solutions for the optimization problem. The candidates are subsequently put through genetic operations such as selection, crossover and mutation, and evolve towards a better solution. Particle swarm optimization (PSO) is a population-based algorithm (Kennedy & Eberhart, 1995). It simulates the behavior of birds flocking or fish schooling in order to achieve a self-evolving system. PSO searches automatically for the optimum solution in the search space by using a search process that is not random. Depending on the different nature of problems, a fitness function decides the best way to conduct this search. The PSO algorithm has rapidly become popular and has been applied in electric power systems (AlRashidi & El-Hawary, 2009), data clustering (Omran, Engelbrecht, & Salman, 2005), biclustering of microarray data (Liu, Li, Hu, & Chen, 2009), engineering design (He & Wang, 2007), etc.

Chaos can be described as a bounded nonlinear system with deterministic dynamic behavior that has ergodic and stochastic properties (Schuster & Just, 2005). It is very sensitive to the initial conditions and the parameters used. In other word, cause and effect of chaos are not proportional to small differences in the initial values. In what is called the "butterfly effect", small variations of

* Corresponding author at: Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 80708, Taiwan. Tel.: +886 7 381 4526x5639; fax: +886 7 383 6844.
E-mail addresses: chuang@isu.edu.tw (L.-Y. Chuang), 1097305142@cc.kuas.edu.tw (C.-J. Hsiao), chyang@cc.kuas.edu.tw (C.-H. Yang).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.05.027

an initial variable can result in huge differences in the solutions after some iteration. Mathematically, chaos is random and unpredictable, yet it also possesses an element of regularity (Alatas, Akin, & Ozer, 2009).

Although evolutionary computation techniques do eventually locate the desired solution, severely limited by the high computational cost associated with the slow convergence rate. PSO applied to clusters in multi-dimensional space has shown outstanding performance. However, the rate of convergence when searching for global optima is still not sufficient (Kao, Zahara, & Kao, 2008). For this reason, we combined chaotic map particle swarm optimization (CPSO) with an accelerated convergence rate strategy, and introduce this accelerated chaotic map particle swarm optimization (ACPSO) in this research. The choice of chaotic sequences as a legitimate complement to PSO is justified by their unpredictability, i.e., by the ergodic properties and spread-spectrum characteristics of these sequences (Alatas et al., 2009). We used these characteristics on chaotic maps and adaptive action to avoid entrapment of the PSO in a local optimum (Chuanwen & Bompard, 2005; Xiang, Liao, & Wong, 2007). This technique allows the ACPSO algorithm to cluster arbitrary data better than previous algorithms. Results of the conducted experimental trials on a variety of data sets taken from several real-life situations demonstrate that ACPSO is superior to the K-means, PSO, NM-PSO, CPSO, K-PSO and K-NM-PSO algorithms (Kao et al., 2008).

2. Methods

2.1. Particle swarm optimization

The robust and efficient PSO evolutionary computation learning algorithm was developed by Kennedy and Eberhart (1995). The original PSO is a population-based optimization technique, where a population is called a swarm. A swarm consists of n particles moving around in a D-dimensional search space.

The position of the ith particle can be represented by $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The velocity for the ith particle can be written as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The positions and velocities of the particles are confined within $[X_{\min}, X_{\max}]^D$ and $[V_{\min}, V_{\max}]^D$, respectively. Each particle coexists and evolves simultaneously based on knowledge shared with its neighboring particles. It makes use of its own memory and knowledge gained by the swarm as a whole to find the best solution.

The best previously encountered position of the ith particle is denoted its individual best position $p_i = (p_{i1}, p_{i2}, ..., p_{iD})$, a value that is called $pbest_i$. The best value of all individual $pbest_i$ values is denoted the global best position $g = (g_1, g_2, \ldots, g_D)$ and called $gbest$. The PSO process is initialized with a population of random particles, and the algorithm then executes a search for optimal solutions by continuously updating generations. In each generation, the position and velocity of the ith particle are updated by $pbest_i$ and $gbest$ of the swarm population. The update equations can be formulated as:

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times r_1 \times (pbest_{id} - x_{id}^{old}) + c_2 \times r_2 \times (gbest_d - x_{id}^{old}) \quad (1)$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new} \quad (2)$$

where r_1 and r_2 are random numbers between (0, 1), and c_1 and c_2 are acceleration constants that control how far a particle moves in a single generation. Velocities v_{id}^{new} and v_{id}^{old} denote the velocities of the new and old particle, respectively. x_{id}^{old} is the current particle position, and x_{id}^{new} is the new, updated particle position. The inertia weight w controls the impact of the previous velocity of a particle on its current one (Shi & Eberhart, 1998). It is defined in Eq. (3)

$$w = 0.5 + \frac{rand}{2.0} \quad (3)$$

In Eq. (3), *rand* is a randomly generated number between zero and one. The pseudo-code of the PSO process is shown below.

PSO pseudo-code

```
begin
  Randomly initialize particle swarm
  while (number of iterations, or the stopping criterion is not met)
    Evaluate fitness of particle swarm
    for n = 1 to number of particles
      Find pbest
      Find gbest
      for d = 1 to number of dimensions of particle
        update the position of particles by Eqs. (1) and (2)
      next d
    next n
    update the inertia weight value with Eq. (3)
  next generation until stopping criterion
end
```

2.1.1. PSO clustering algorithm

Over the past several years, PSO has been proven to be both effective and fast for solving optimization problems, PSO shows a promising performance on nonlinear function optimization and has thus received much attention (Liu, Qin, Shi, & Lu, 2007). It has been successfully applied in many research and application areas (AlRashidi & El-Hawary, 2009; He & Wang, 2007; Liu et al., 2009; Omran et al., 2005). The clustering problem can be viewed as an optimization problem in data clustering research area of locating the optimal centroids for each cluster instead of other non-optimal partitioning.

The PSO clustering algorithm, analogous to many clustering algorithms and partitioning methods, is employed to minimize intra-cluster distances as well as maximize distances between clusters by uncovering a proper set of cluster centroids fulfilling the given objectives. What separates the PSO algorithm from most other cluster partitioning methods is its capability to perform a global search. Most other partitioning methods perform only a local search, a situation under which the obtained solution is usually close to the solution obtained in the previous step. Take the K-means clustering algorithm for example. This method initializes a search with a set of preliminary cluster centroids from randomly generated seeds, and then iteratively updates the positions of the cluster centroids at each step. This cluster-refining procedure suggests that the K-means algorithm only probes proximal regions around of the randomly generated initial solution.

The partitioning performed by the PSO clustering algorithm is achieved by the integration of two procedures: a global search and a localized search, in which the refining process takes place. Based on the particle velocity updating Eq. (1) from the PSO algorithm, v_i represents the initial velocity for particle i, (r_1, r_2) are two random numbers generated at each iteration from a uniform distribution in the range of (0, 1) and w is the inertia weight factor necessary for diversifying the searching behavior of the particle swarm by altering the momentum. Particles can thus avoid entrapment in a local optimum. Since searches are performed concurrently by the swarm, the algorithm evaluates a wide variety of solutions by particles exploring the problem space. The global search step is performed in the initial iterations. A particle's velocity is gradually

reduced after several iterations and the particle's area of exploration shrinks when the particle approaches the optimal solution. Thus, the searching procedure gradually moves from the global search stage to the local refining stage. With different selections of parameters applied to the PSO algorithm, the transition time from the global search stage to the local refining stage can be controlled. By delaying the transition from the global search stage to the local refining stage, the possibility of finding global optimal solution is increased.

In the PSO clustering algorithm, the problem space is modeled as data vectors in a multi-dimensional space. A single particle in the swarm represents one possible solution for clustering the data collection. Therefore, a swarm contains a group of candidate solutions for clustering the data collection. Each particle is represented by a matrix $x_i = (C_1, C_2, \ldots, C_j, \ldots, C_k)$, where C_j specifies the vector of the jth cluster centroid and k is the number of clusters. The particle will then update the positions of clusters' centroids in the each iteration according to knowledge acquired from its own experience and that of particles in its neighborhood. To evaluate the performance of each solution, the fitness value is defined as the average distance of data points to the cluster centroid.

2.2. Chaos theory

In the field of engineering, it is well recognized that chaos theory can be applied as a very useful technique in practical application. The chaotic system can be described by a phenomenon, in which a small change in the initial condition will lead to nonlinear change in future behavior, besides that the system exhibits distinct behaviors under different phases, i.e. stable fixed points, periodic oscillations, bifurcations, and ergodicity (dos Santos Coelho & Herrera, 2007). Chaos (Lu, Zhang, & Ma, 2006) is also a common nonlinear phenomenon with much complexity and is similar to randomness. Chaos is typically highly sensitive to the initial values and thus provides great diversity based on the ergodic property of the chaos phase, which transits every state without repetition in certain ranges. It is generated through a deterministic iteration formula. Due to these characteristics, chaos theory can be applied in optimization.

One of the simplest maps, the logistic map, was brought to the attention of scientists by May (1976). It appears in nonlinear dynamics of biological population evidencing chaotic behavior. The logistic map can be described by the following equation:

$$X_{(n+1)} = a \times X_{(n)} \times (1 - X_{(n)}) \tag{4}$$

In this equation, $X_{(n)}$ is the nth chaotic number where n denotes the iteration number. Obviously, $X_{(n)} \in (0, 1)$ under the conditions that the initial $X_{(0)} \notin (0, 1)$ and that $X_{(0)}$ {0.0, 0.25, 0.5, 0.75, 1.0}. In our experiments $a = 4$ has been used. As show in Fig. 1a, the chaotic sequence value X is bounded within (0, 1) when a equals 4. Fig. 1b shows the chaotic X value of a logistic map for 100 iterations where $X_{(0)} = 0.0001$.

2.3. Chaotic particle swarm optimization (CPSO)

In PSO, the main advantage of the chaotic optimization is the maintenance of population diversity in the problem of interest. According to studies conducted by Clerc and Kennedy (2002) and Trelea (2003), the parameters w, c_1, c_2, r_1 and r_2 are generally the key factors affecting the typical PSO convergence. Therefore, this paper incorporates chaotic mapping with ergodic, irregular, and stochastic properties in PSO to improve the global convergence. The use of chaotic sequences in PSO can facilitate the escape from local minima. The literature is rich in chaotic time-series sequences, such as logistic map, ten map, Lozi map, Ikeda map, Hénon map, and others (Caponetto, Fortuna, Fazzino, & Xibilia, 2003; Coelho & Mariani, 2009; May, 1976; Singh & Sinha, 2009). In this study, we combine the logistic map (May, 1976) with particle swarm optimization in a process we named chaotic particle swarm optimization (CPSO).

In CPSO, sequences generated by the logistic map (May, 1976) substitute the random parameters r_1 and r_2 of PSO. The parameters r_1 and r_2 are modified by the logistic map based on the following equation:

$$Cr_{(t+1)} = k \times Cr_{(t)} \times (1 - Cr_{(t)}) \tag{5}$$

In Eq. (5), $Cr_{(0)}$ is generated randomly for each independent run, with $Cr_{(0)}$ not being equal to {0, 0.25, 0.5, 0.75, 1} and k equal to 4. The driving parameter k of the logistic maps controls the behavior of $Cr_{(t)}$ (as t goes to infinity). The behavior of the logistic map for various values of the parameter k is shown in Fig. 1a. For low values of k ($k < 3$), Cr eventually converges to a single number. When $k = 3$, Cr oscillates between two values. This characteristic change in behavior is called bifurcation. For $k > 3$, Cr goes through further bifurcations, eventually resulting in chaotic behavior. In fact, the bifurcation diagram is itself a fractal (Kuo, 2005). The velocity update equation for CPSO can be formulated as:

$$v_{id}^{new} = w \times v_{id}^{old} + c_1 \times Cr \times (pbest_{id} - x_{id}^{old}) + c_2 \times (1 - Cr) \times (gbest_d - x_{id}^{old}) \tag{6}$$

In Eq. (5), Cr is a function based on the results of the logistic map with values between 0.0 and 1.0. The pseudo-code of CPSO is shown below.

CPSO pseudo-code

```
begin
  Randomly initialize particle swarm
  Randomly generate Cr(0)
  while (number of iterations, or the stopping criterion is not met)
    Evaluate fitness of particle swarm
    for n = 1 to number of particles
      Find pbest
      Find gbest
      for d = 1 to number of dimensions of particle
        update the Chaotic Cr value with Eq. (5)
        update the position of particles with Eqs. (6) and (2)
      next d
    next n
    update the inertia weight value with Eq. (3)
  next generation until stopping criterion
end
```

2.4. Accelerated chaotic particle swarm optimization (ACPSO)

In order to increase the particle distribution, we used a chaotic map to allow the PSO to find optimal solutions. We also increased the speed of convergence of PSO by incorporating an acceleration strategy in a process we call ACPSO. The acceleration strategy is similar to the K-means algorithm, but it does not perform a complete K-means algorithm. The strategy focuses on the part of the calculations where the particle cluster around the arithmetic average of the center; it calculates the arithmetic mean center. A part of the particles contained in the original cluster center are then replaced. We reduce the sum of the intra-cluster distance to find optimal solutions and thereby expedite the rate of convergence in CPSO.

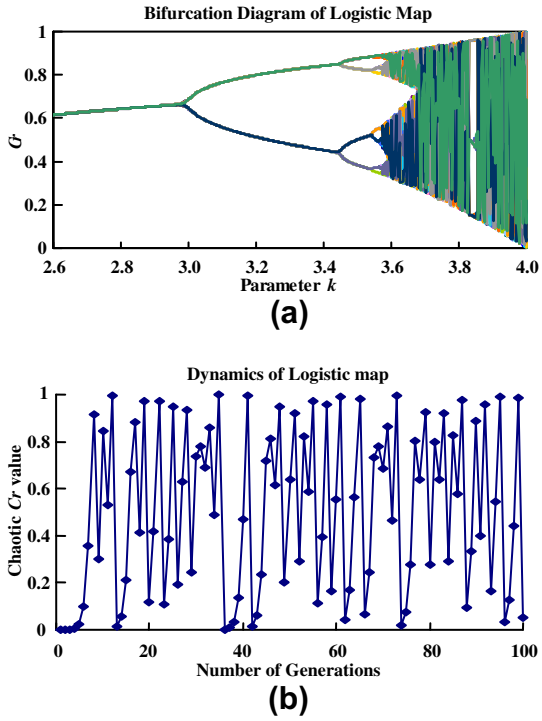

Fig. 1. Bifurcation diagram and dynamics of logistic map.

2.4.1. Particle encoding

Each particle contained in the cluster centers represents a feasible solution for a clustering problem. The dimensions of each particle are equal to the dimension of the data vector multiplied by the number of clusters.

2.4.2. Initial population

ACPSO randomly generates $3N$ particles. The parameter N is based on different data sets and carries out changes. It is defined in Eq. (7), where d is the dimension of data set and k is the anticipated number of clusters. All particles in the solution space are randomly generated with an individual position and velocity

$$N = k \times d \tag{7}$$

2.4.3. Fitness evaluation

The fitness value is the sum of the intra-cluster distances. When the sum of the distances is small, the clustering results are considered excellent. This sum of distances has a profound impact on the error rate. The fitness value of each particle can be computed with the fitness function, where k and n are the numbers of the clusters and the data sets, respectively. Z_i is cluster center i and X_j is data point j

$$\text{fitness} = \sum \|X_j - Z_i\|, \quad i = 1, \ldots, k, \; j = 1, \ldots, n \tag{8}$$

2.4.4. Acceleration strategy

As mentioned above, the acceleration strategy is similar to a K-means algorithm. It focuses on the results of the calculation of the arithmetic average of the centers, and replaces a part of the particles contained in the original cluster center. In the initial population, one-third of the particles is used to accelerate the convergence rate of the particles. The distances between data vectors within a cluster and the center of the cluster are defined in Eq. (9). The acceleration strategy recalculates the cluster center vectors using Eq. (10) and yields mean centers. The mean clusters then replace the original centers. This is the new position of the particle. z_j denotes the center vector of cluster j, x_p denotes the pth data vector, the d subscript represents the number of features of each center vector, n_j is the number of data vectors in cluster j, and C_j is the subset of data vectors that form cluster j

$$D(x_p \cdot z_j) = \sqrt{\sum_{i=1}^{d} (x_{pi} - z_{ji})^2} \tag{9}$$

$$z_j = \frac{1}{n_j} \sum_{\forall x_p \in C_j} x_p \tag{10}$$

2.4.5. ACPSO procedure

The data clustering procedures of the proposed ACPSO algorithm can be summarized as follows:

Step (1) Initial population: Each particle is randomly generated. All particles in the solution space are randomly generated with an individual position and velocity. Initialize population of particle with random position x ($x \in \{x_1, x_2, \ldots, x_n\}$) and the velocity for the ith particle can be written as v ($v \in \{v_1, v_2, \ldots, v_n\}$), where n is the number of particles. Every particle is contained the center position of each cluster.
Step (2) Acceleration strategy: In an initial step, one-third of the particles are used to accelerate the convergence rate of the particles.
Step (2.1) Select the top one-third of the particles
Step (2.2) Group the data vectors for one-third of the particles. Assign the vector to the cluster with the closest centroid vector, where the distance to the centroid is determined by Eq. (9).
Step (2.3) Recalculate the cluster centroid vectors, using Eq. (10).
Step (2.4) Get the new centroid vectors and replace particle position.
Step (3) Group the data vectors for every particle: The data vectors are grouped into k clusters on the basis of the Euclidean distance as the similarity measure. Each particle maintains a matrix $x_i = (C_1, C_2, \ldots, C_j, \ldots, C_k)$, where C_j represents the ith cluster centroid vector and k is the number of clusters. For each data vector, assign the vector to the cluster with the closest centroid vector, where the distance to the centroid is determined by Eq. (9).
Step (4) Fitness evaluation: the fitness function of all particles is calculated. The fitness value of each particle is defined in Eq. (8).
Step (5) Update *pbest* and *gbest*: At each iteration, every individual compares its current fitness value to its own *pbest* and global best solution *gbest*. The *pbest* and *gbest* values are updated if the new values are better than the old ones.
Step (6) Velocity and position update: The particles move through the search space at each ite-ration. In CPSO, sequences generated by the logistic map (May, 1976) substitute the random parameters r_1 and r_2 of PSO.
Step (7) Repeat steps 3–6 until the termination condition is met.

2.4.6. Parameter settings

The parameter N is based on the data sets and carries out changes. It is defined as $k \times d$, where k is the anticipated number of clusters and d is the data set dimension, respectively. The values of the different parameters of the ACPSO algorithm in this study are as follows: number of iteration = $10N$, population size = $3N$, V_{max} and V_{min} were set as maximum and minimum values from each dimension of our data set, the initial inertia weight w was 0.5 + (*rand*/2), *rand* was a randomly generated number between 0

and 1, c_1 and c_2 were 2 (Kao et al., 2008), and one-third of the particles were used to accelerate the convergence rate of the particles. The fraction one-third of the particles has been decided upon after several experimental trials.

3. Illustrative example

This section provides an example that illustrates the process details, in particular the steps regarding the acceleration strategy (Steps 2.1–2.4) of the proposed ACPSO clustering algorithm. We demonstrate the differences between ACPSO with acceleration strategy and CPSO without acceleration strategy. We assume a two-dimensional data set containing 10 data points. The data points 1–10 used in this example are (1.0, 9.0), (1.5, 7.0), (3.0, 8.0), (3.0, 2.0), (3.5, 3.0), (4.0, 2.5), (6.0, 5.0), (7.0, 4.0), (7.0, 8.0), and (8.0, 6.0). These data points are assumed to be divided into three clusters, i.e., $k = 3$, $d = 2$, where k is the anticipated number of clusters and d is the dimension of data set. The goal is to minimize the intra-cluster distance of the data set. The parameter $N = k \times d = 2 \times 3 = 6$. The ACPSO algorithm consists of the following seven major steps:

Step (1) Initial population. In this step, ACPSO is started on the initial population. ACPSO initializes a randomly generated population for each particle's velocity and location, the particle dimension is $N = 6$ and the population size is $3N = 18$. The assumed position of the 1st randomly generated particle is $x_1 = (2, 6, 4, 1, 8, 5)$.
Step (2) Implementation of acceleration strategy. We selected 1/3 of the particles (i.e., the first six particles) to implement the acceleration strategy. In step 2.2, all data vectors are grouped into three clusters surrounding the centers (2, 6), (4, 1), (8, 5) by Eq. (9). The results show that Data 1, 2, 3 belong to cluster center (2, 6), Data 4, 5, 6 belong to cluster center (4, 1), and Data 7, 8, 9, 10 belong to cluster center (8, 5). In step 2.3, each arithmetic mean center is updated by Eq. (10), i.e., the updated center of cluster 1 is $\left(\frac{\text{Data1}.x+\text{Data2}.x+\text{Data3}.x}{3}, \frac{\text{Data1}.y+\text{Data2}.y+\text{Data3}.y}{3}\right) = \left(\frac{1+1.5+3}{3}, \frac{9+7+8}{3}\right) = (1.83, 8)$.
Following similar procedures, the updated centers of cluster 2, 3 are (3.5, 2.5) and (7, 5.75), respectively. In step 2.4, these new cluster center vectors are then combined to replace the original particle, i.e., the position of the first particle is (1.83, 8, 3.5, 2.5, 7, 5.75) after this step.
Step (3) Grouping the data vectors for each particle. In step 3, each of the 18 particles assigns these ten data points into three clusters according to centers defined by particle's position, which results in a total of 18 different clustering configurations.
Step (4) Calculation of the fitness value of each particle. In step 4, Eq. (8) is used to calculate the fitness function for each particle. The fitness value is the sum of the intra-cluster distances of all clusters. Take the first particle for example. The intra-cluster distances of the three clusters, are 3.52, 1.71, 6.28 respectively. The fitness value is thus 3.52 + 1.71 + 6.28 = 11.51.

Intra-cluster distance of Cluster 1:

$$\sqrt{(1.83-1)^2+(8-9)^2}+\sqrt{(1.83-1.5)^2+(8-7)^2}+\sqrt{(1.83-3)^2+(8-8)^2}=3.52$$

Intra-cluster distance of Cluster 2:

$$\sqrt{(3.5-3)^2+(2.5-2)^2}+\sqrt{(3.5-3.5)^2+(2.5-3)^2}+\sqrt{(3.5-4)^2+(2.5-2.5)^2}=1.71$$

Intra-cluster distance of Cluster 3:

$$\sqrt{(7-6)^2+(5.75-5)^2}+\sqrt{(7-7)^2+(5.75-4)^2}+\sqrt{(7-7)^2+(5.75-8)^2}+\sqrt{(7-8)^2+(5.75-6)^2}=6.28$$

Step (5) Update pbest and gbest.
In step 5, a comparison is performed to update *pbest* and *gbest*. The *pbest* and *gbest* values are updated if the new values are better than the old ones.
Step (6) Velocity and position update.
This step uses Eqs. (2) and (6) to update the 18 particle positions and velocities.
Step (7) Repetition of steps 3-6 until the stopping criteria is satisfied.
Steps 3–6 are repeated when the number of iterations exceeds $10N = 60$. If the stopping criteria are satisfied, the best particle is output.

4. Experimental results and discussion

4.1. Data sets

To validate our method, six experimental data sets, named Vowel, Iris, Crude Oil, Contraceptive Method Choice (CMC), Cancer, and Wine were used. These data sets cover examples of data with a low, medium and high number of dimensions. All data sets are available at ftp://ftp.ics.uci.edu/pub/machine-learning-databases/. Table 1 summarizes the characteristics of these data sets. Given a data set with three features that is to be grouped into two clusters, the number of parameters to be optimized in order to find the two optimal cluster center vectors is equal to the product of the number of clusters and the number of features, i.e., $N = k \times d = 2 \times 3 = 6$.

The six real-life data sets are briefly described below.

(1) The Vowel data set consists of 871 Indian Telugu vowel sounds. The data set has three features corresponding to the first, second, and third vowel frequencies and six overlapping classes, δ (72 objects), a (89 objects), i (172 objects), u (151 objects), e (207 objects), o (180 objects).
(2) Fisher's iris data set consists of three different species of iris flowers: Iris setosa, Iris virginica, and Iris versicolour. For each species, 50 samples with four features each (sepal length, sepal width, petal length, and petal width) were collected.
(3) The Crude Oil data set consists of 56 objects characterized the five features such as vanadium, iron, beryllium, saturated hydrocarbon, and aromatic hydrocarbon content. There are three crude-oil samples from three zones of sandstone (Wilhelm has 7 objects, Sub-Mulnia has 11 objects, and Upper has 38 objects).
(4) The Contraceptive Method Choice (CMC) data set is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples were married women who either were not pregnant or did not know if they were pregnant at the time interviews were conducted. The problem consisted of predicting the choice of the current contraceptive method (no contraception has 629 objects, long-term methods have 334 objects, and short-term methods have 510 objects) of a woman based on her demographic and socioeconomic characteristics.
(5) The Wisconsin breast cancer data set consists of 683 objects characterized by nine features: clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. There are two categories in the data: malignant (444 objects) and benign (239 objects).

Table 1
Characteristics of the considered data sets.

Name of data set	Number of classes (k)	Number of features (d)	Size of data set (n)	Size of individual classes in parentheses
Vowel	6	3	871	(72, 89, 172, 151, 207, 180)
Iris	3	4	150	(50, 50, 50)
Crude Oil	3	5	56	(7, 11, 38)
CMC	3	9	1473	(629, 334, 510)
Cancer	2	9	683	(444, 239)
Wine	3	13	178	(59, 71, 48)

(6) The Wine data set consists of 178 objects characterized by 13 features: alcohol, malic acid, ash content, alcalinity of ash, concentration of magnesium, total phenols, flavanoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, and OD280/OD315 of diluted wines and pralines. The results were obtained by chemical analysis of wines produced in the same region of Italy but derived from three different cultivars. The quantities of objects in the three categories of the data are: class 1 (59 objects), class 2 (71 objects), and class 3 (48 objects).

4.2. Results

In order to demonstrate the performance of ACPSO, we compared the ACPSO results to results obtained with the following methods: K-means, PSO, NM-PSO, K-PSO, K-NM-PSO, and CPSO. The quality of the respective clustering was also compared, where quality is measured by the following two criteria, sum of the intra-cluster distances and error rate.

1. Sum of the intra-cluster distances: the distances between data vectors within a cluster and the centroid of the cluster, as defined in Eq. (9). A higher quality of clustering is achieved if the sum is relatively small.
2. Error rate: the number of misplaced points divided by the total number of points, as shown in Eq. (11):

$$\text{error rate} = \left(\frac{\sum_{i=1}^{n}(\text{if } A_i = B_i) \text{ then 0 else 1})}{n}\right) \times 100 \qquad (11)$$

where n denotes the total number of points. A_i and B_i denote the data sets of which the ith point is a member before and after clustering, respectively. Table 2 serves as an example, in which two data points (6, 3) and (5, 4) out of clusters 1 and 2 are misplaced and the error rate is 2/5, i.e., 40%.

The algorithms were implemented using Java. For each run, $10 \times N$ iterations were carried out on each of the six data sets for every algorithm when solving an N-dimensional problem. The criterion adopted was $10 \times N$, as it has been used with great success in terms of effectiveness in many previous experiments (Kao et al., 2008).

Table 2
Error rate calculations.

i	Data point	A_i	B_i	Not misplaced (0)/misplaced (1)
1	(2, 6)	2	2	0
2	(6, 3)	2	1	1
3	(1, 7)	1	1	0
4	(5, 4)	2	1	1
5	(8, 7)	1	1	0
Number of misplaced point: 2				

Table 3 summarizes the intra-cluster distances obtained from the seven clustering algorithms for the data sets in Table 1. The values reported are averages over 20 simulations from the sums of intra-cluster distances. In addition to the best fitness solutions, the standard deviations are given in parentheses to demonstrate the spanning range of values that the algorithms produced. The summary of the test results of data sets Vowel, Iris and Crude Oil indicates that PSO outperforms the GA method, independent of whether the average intra-cluster distance or best intra-cluster distance is measured. After hybridization with the K-means methods, K-PSO still leads KGA, except for a tiny slight loss in average intra-cluster distance performance for data set Vowel. As can be seen from these results, PSO offers better optimized solutions than GA with or without integration of the K-means method. For all experimental data sets, CPSO outperformed PSO and NM-PSO, as born out by a smaller difference between the averages and a smaller standard deviation. For the Crude Oil, CMC, Cancer and Wine data sets, the averages and standard deviation of CPSO are smaller than the ones for K-PSO and K-NM-PSO. K-PSO is a hybrid of the K-means and PSO algorithm, and K-NM-PSO is a hybrid of the K-means, Nelder–Mead simplex search (Nelder & Mead, 1965) and PSO. In the Vowel, Crude Oil, CMC and Cancer data sets, the best solution obtained with CPSO is smaller than the one obtained with K-PSO. In the Vowel, CMC and Cancer data sets, the best solution of CPSO is smaller than the best solution of K-NM-PSO. For all experimental data sets, ACPSO outperformed the other five methods. Please note that in terms of the best distance, PSO, NM-PSO, K-PSO, K-NM-PSO, and CPSO all have a larger standard deviation than ACPSO, even though they may achieve a global optimum. This means that PSO, NM-PSO, K-PSO, K-NM-PSO, and CPSO are weaker search tools for global optima than ACPSO if all algorithms are executed just once. It follows that ACPSO is more efficient in finding the global optimum solution than the other five methods.

Table 4 shows the mean error rates, standard deviations, and the best solution of the error rates from the 20 simulation runs. For all data sets except the Vowel, Crude Oil and Wine data sets, CPSO exhibited a significantly smaller mean and standard deviation compared to K-means, PSO, NM-PSO and K-PSO. For the Iris and Cancer data sets, the averages and standard deviation of CPSO are smaller than the averages and standard deviation of K-NM-PSO, and for the CMC data set, the average is equal the average of K-PSO and K-NM- PSO. For all the real-life data sets except Crude Oil, ACPSO exhibited a significantly smaller mean and standard deviation compared to K-means, PSO, NM-PSO, K-PSO, K-NM-PSO, and CPSO. Again, ACPSO is superior to the other six methods with respect to the intra-cluster distance. However, it does not compare favorably to the other methods for the Vowel, Iris, Crude Oil, and CMC data sets in terms of the best error rate. Although ACPSO in the Crude Oil data set does not obtain the best error rate, the intra-cluster distance is the smallest (Table 3). It should be noted that the intra-cluster distance is not proportional to the error rate (Kao et al., 2008). The actual data distribution was not regular and therefore a smaller intra-cluster distance does not necessarily indicate a lower error rate.

Summaries for the comparison of algorithms unequipped with methods for local search and K-means, i.e. PSO, CPSO and ACPSO, are given below. According to the experimental results, the CPSO algorithm outperforms PSO through the introduction of chaos theory via chaotic mapping, which results in unlimited possibilities for randomization from 0 to 1 (Alatas et al., 2009). As a consequential benefit, particles are less likely to roam around in the proximity of previous positions due to the more uniform distribution of positions. The chaotic mapping improves PSO by giving particles a higher probability of leaving local optima due to acyclic and ergodic behavior (Chuanwen & Bompard, 2005; Xiang et al., 2007). Compared to CPSO, ACPSO further improves the performance

Table 3
Comparison of intra-cluster distances for the nine clustering algorithms.

Data set	Criteria	K-means	GA	KGA	PSO	NM-PSO	K-PSO	K-NM-PSO	CPSO	ACPSO
Vowel	Average	159242.87	390088.24	149368.45	168477.00	151983.91	149375.70	149141.40	151337	**149051.84**
	(Std)	(916)	N/A	N/A	(3715.73)	(4386.43)	(155.56)	(120.38)	(3491.43)	(67.27)
	Best	149422.26	383484.15	149356.01	163882.00	149240.02	149206.10	149005.00	148996.5	148970.84
Iris	Average	106.05	135.40	97.10	103.51	100.72	96.76	96.67	96.90	**96.66**
	(Std)	(14.11)	N/A	N/A	(9.69)	(5.82)	(0.07)	(0.008)	(0.303)	(0.001)
	Best	97.33	124.13	97.10	96.66	96.66	96.66	96.66	96.66	96.66
Crude Oil	Average	287.36	308.16	278.97	285.51	277.59	277.77	277.29	277.24	**277.24**
	(Std)	(25.41)	N/A	N/A	(10.31)	(0.37)	(0.33)	(0.095)	(0.038)	(0.04)
	Best	279.20	297.05	278.97	279.07	277.19	277.45	277.15	277.21	277.21
CMC	Average	5693.60	N/A	N/A	5734.20	5563.40	5532.90	5532.70	5532.23	**5532.20**
	(Std)	(473.14)	N/A	N/A	(289.00)	(30.27)	(0.09)	(0.23)	(0.04)	(0.01)
	Best	5542.20	N/A	N/A	5538.50	5537.30	5532.88	5532.40	5532.19	5532.19
Cancer	Average	2988.30	N/A	N/A	3334.60	2977.70	2965.80	2964.70	2964.49	**2964.42**
	(Std)	(0.46)	N/A	N/A	(357.66)	(13.73)	(1.63)	(0.15)	(0.12)	(0.03)
	Best	2987	N/A	N/A	2976.30	2965.59	2964.50	2964.50	2964.40	2964.39
Wine	Average	18061.00	N/A	N/A	16311.00	16303.00	16294.00	16293.00	16292.90	**16292.31**
	(Std)	(793.21)	N/A	N/A	(22.98)	(4.28)	(1.70)	(0.46)	(0.78)	(0.03)
	Best	16555.68	N/A	N/A	16294.00	16292.00	16292.00	16292.00	16292.19	16292.18

The results of GA can be found in Murthy and Chowdhury (1996), the results of KGA can be found in Bandyopadhyay and Maulik (2002). The results of K-means, PSO, NM-PSO, K-PSO, K-NM-PSO can be found in Kao et al. (2008). N/A: data not available. Highest values are indicated in bold type.

Table 4
Comparison of error rates for the seven clustering algorithms.

Data set	Criteria	K-means (%)	PSO (%)	NM-PSO (%)	K-PSO (%)	K-NM-PSO (%)	CPSO (%)	ACPSO (%)
Vowel	Average	44.26	44.65	41.96	42.24	41.94	42.23	**41.69**
	(Std)	(2.15)	(2.55)	(0.98)	(0.95)	(0.95)	(1.82)	(0.31)
	Best	42.02	41.45	40.07	40.64	40.64	37.54	41.10
Iris	Average	17.80	12.53	11.13	10.20	10.07	10.00	**9.80**
	(Std)	(10.72)	(5.38)	(3.02)	(0.32)	(0.21)	(0.00)	(0.84)
	Best	10.67	10.00	8.00	10.00	10.00	10.00	8.00
Crude Oil	Average	24.46	24.64	24.29	24.29	23.93	26.52	26.25
	(Std)	(1.21)	(1.73)	(0.75)	(0.92)	(0.72)	(0.65)	(0.84)
	Best	23.21	23.21	23.21	23.21	23.21	25.00	25.00
CMC	Average	54.49	54.41	54.47	**54.38**	**54.38**	**54.38**	**54.38**
	(Std)	(0.04)	(0.13)	(0.06)	(0.00)	(0.054)	(0.015)	(0.00)
	Best	54.45	54.24	54.38	54.38	54.31	54.38	54.38
Cancer	Average	4.08	5.11	4.28	3.66	3.66	**3.51**	**3.51**
	(Std)	(0.46)	(1.32)	(1.10)	(0.00)	(0.00)	(9.11E−16)	(0.00)
	Best	3.95	3.66	3.66	3.66	3.66	3.51	3.51
Wine	Average	31.12	28.71	28.48	28.48	28.37	28.62	**28.23**
	(Std)	(0.71)	(0.27)	(0.27)	(0.40)	(0.27)	(0.43)	(0.25)
	Best	29.78	28.09	28.09	28.09	28.09	28.09	28.09

The results of K-means, PSO, NM-PSO, K-PSO, K-NM-PSO can be found in Kao et al. (2008)). Highest values are indicated in bold type.

through the employed acceleration strategy. This method provides a faster rate of convergence by resetting the cluster centers of the particles, and thus enables ACPSO to perform better than CPSO in the search of optimal solution.

The population size of PSO and K-PSO was $5N$, for NM-PSO and K-NM-PSO it was $3N + 1$, and for CPSO and ACPSO $3N$. The population size of CPSO and ACPSO was thus smaller than the population size of the other algorithms. This results in a lower computational cost for CPSO and ACPSO. In a direct performance comparison, ACPSO proved to be superior to K-NM-PSO.

Table 5 lists the number of objective function evaluations required by the five methods after $10 \times N$ iterations. As an average, the K-means algorithm needed the fewest function evaluations on all data sets, but its results are less than satisfactory, as seen in Tables 3 and 4. This algorithm tends to get trapped in a local optimum. CPSO and ACPSO uses fewer function evaluations than PSO, NM-PSO, K-PSO, and K-NM-PSO, and produces better outcomes than these methods. All the evidence gathered in the simulations illustrates that ACPSO converges to global optima with fewer function evaluations and a smaller error rate than the other algorithms, which naturally leads to the conclusion that ACPSO is a viable and robust technique for data clustering.

Fig. 2 provides more insight into the convergence behavior of the PSO, CPSO, K-PSO and ACPSO algorithms. It respectively illustrates the trends of convergence of the algorithms for the Vowel, Iris, Crude Oil, CMC, Cancer and Wine data sets. For all data sets, the PSO algorithm exhibits a fast but premature convergence to a local optimum. K-PSO and CPSO converge near the global optimum. For the Vowel data set, K-PSO converges in about 29 iterations to the near global optimum, and CPSO converges in about 176 iterations to the near global optimum ACPSO converges in about 172 iterations to the global optimum and correctly classifies this data set into six clusters. PSO classifies this data set with a 44.65% error rate, CPSO classifies this data set with a 42.23% error rate, K-PSO classifies this data set with a 42.24% error rate, and ACPSO classifies this data set with a 41.69% error rate. For the Wine

Table 5
Number of function evaluations of each clustering algorithm.

Data set	K-means	PSO	NM-PSO	K-PSO	K-NM-PSO	CPSO	ACPSO
Vowel	180	16,290	10,501	15,133	**9291**	9774	9774
Iris	120	7260	4836	6906	4556	**4356**	**4356**
Crude Oil	150	11,325	7394	10,807	7057	**6795**	**6795**
CMC	270	36,585	23,027	34,843	**21,597**	21, 951	21,951
Cancer	180	16,290	10,485	15,756	10,149	**9774**	**9774**
Wine	390	73,245	47,309	74,305	46,459	**45,747**	**45,747**
Average	215	26,833	17,259	26,292	16,519	**16,400**	**16,400**

The results of K-means, PSO, NM-PSO, K-PSO, K-NM-PSO can be found in Kao et al. (2008). Highest values are indicated in bold type.

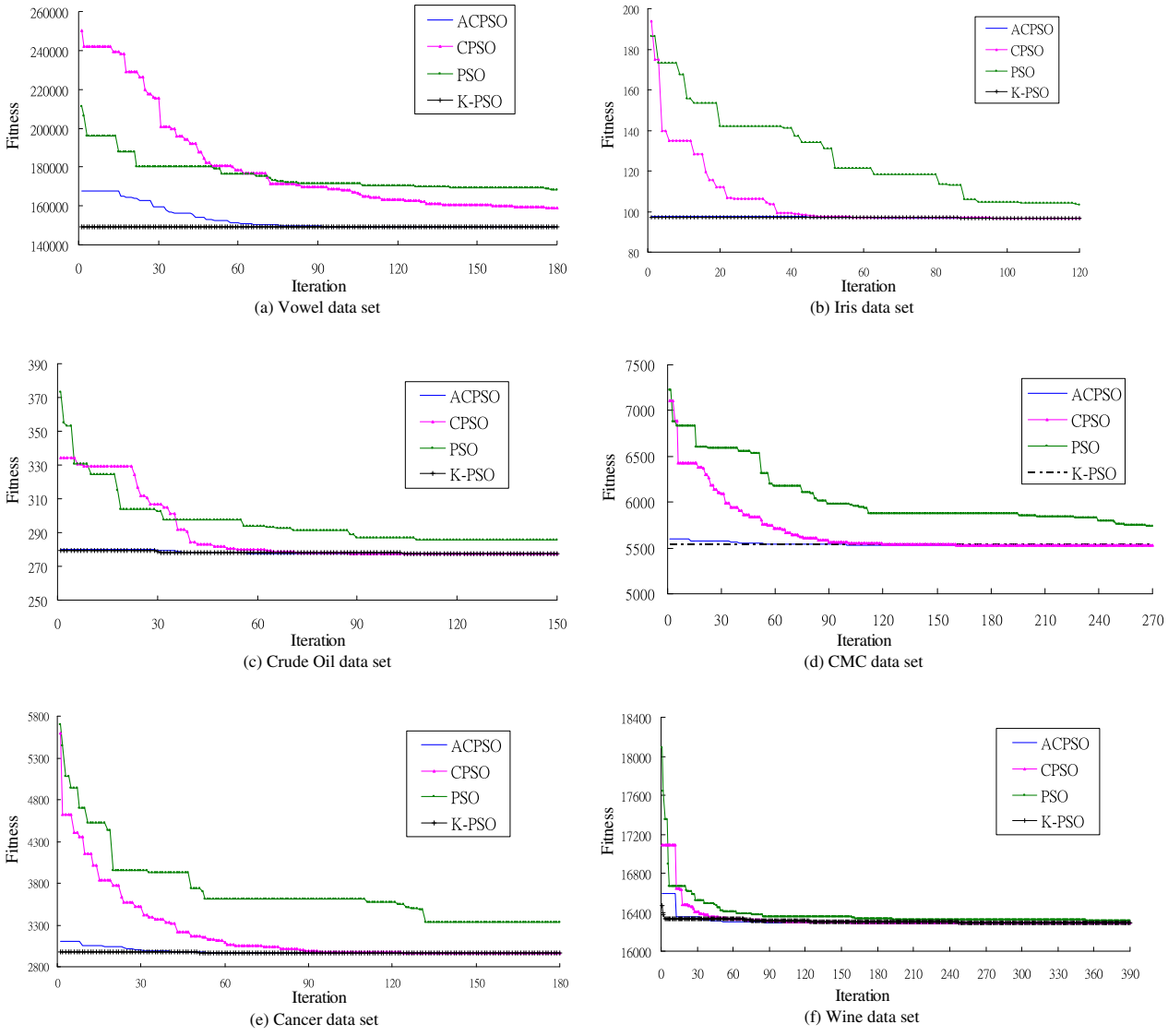

Fig. 2. Convergence of four algorithms for six data sets: (a) Vowel data set; (b) Iris data set; (c) Crude Oil data set; (d) CMC data set; (e) Cancer data set; (f) Wine data set. For all data sets, the PSO algorithm exhibits a fast but premature convergence to a local optimum. K-PSO and CPSO converge near the global optimum, and ACPSO converges to the global optimum.

data set, K-PSO converges in about 285 iterations to the near global optimum, and CPSO converges in about 385 iterations to the near global optimum ACPSO converges in about 325 iterations to the global optimum and correctly classifies this data set into three clusters. PSO classifies this data set with a 28.71% error rate, CPSO classifies this data set with a 28.62% error rate, K-PSO classifies this data set with a 28.48% error rate and ACPSO classifies this data set with a 28.23% error rate. Fig. 2 shows that ACPSO initially converges faster than PSO. The fitness value and error ratio of the latter iterations of ACPSO are also better than the ones of CPSO and PSO. Although K-PSO initially converges faster than ACPSO, the final fitness value and error ratio are inferior to ACPSO. The above statements prove that ACPSO does not only converge rapidly, but also obtains better solutions than the other algorithms.

5. Conclusions

In this paper, we employed the ACPSO algorithm to clustering data vectors for six data sets. ACPSO uses minimum intra-cluster distances as a metric, and searches for robust data cluster centers in an *N*-dimensional Euclidean space. Under the same metric, PSO, NM-PSO, K-PSO, and K-NM-PSO need more iteration to arrive at a global optimum than ACPSO. The K-means algorithm has a tendency of getting stuck in a local optimum, depending on the choice of the initial cluster centers. Although the method introduced utilizes neither k-means nor local search, the results obtained are better than the results of other literature hybrid algorithms. The experimental results indicate that ACPSO reaches a minimal error rate faster than the other methods, and thus reduces the computational cost. The ACPSO algorithm developed in this paper can be applied when the number of clusters is known a priori and the clusters are clearly defined.

Acknowledgement

This work was partly supported by the National Science Council in Taiwan under Grants NSC96-2221-E-214-050-MY3, NSC96-2622-E-214-004-CC3, and NSC97-2622-E-151-008-CC2.

References

Alatas, B., Akin, E., & Ozer, A. B. (2009). Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons & Fractals, 40*, 1715–1734.

AlRashidi, M. R., & El-Hawary, M. E. (2009). A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation, 13*, 913–918.

Anderberg, M. R. (1973). *Cluster analysis for applications*. New York: Academic Press.

Bandyopadhyay, S., & Maulik, U. (2002). An evolutionary technique based on K-Means algorithm for optimal clustering in R^N. *Information Science, 146*, 221–237.

Caponetto, R., Fortuna, L., Fazzino, S., & Xibilia, M. G. (2003). Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation, 7*, 289–304.

Chuanwen, J., & Bompard, E. (2005). A self-adaptive chaotic particle swarm algorithm for short term hydroelectric system scheduling in deregulated environment. *Energy Conversion and Management, 46*, 2689–2696.

Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*, 58–73.

Coelho, L. d. S., & Mariani, V. C. (2009). A novel chaotic particle swarm optimization approach using Hénon map and implicit filtering local search for economic load dispatch. *Chaos, Solitons & Fractals, 39*, 510–518.

dos Santos Coelho, L., & Herrera, B. M. (2007). Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system. *IEEE Transactions on Industrial Electronics, 54*, 3234–3245.

Han, J., Kamber, M., & Tung, A. K. H. (2001). *Spatial clustering methods in data mining: A survey*. London: Taylor & Francis.

He, Q., & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence, 20*, 89–99.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys, 31*, 264–323.

Kao, Y.-T., Zahara, E., & Kao, I. W. (2008). A hybridized approach to data clustering. *Expert Systems with Applications, 34*, 1754–1762.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *IEEE international joint conference on neural network* (Vol. 4, pp. 1942–1948).

Kuo, D. (2005). Chaos and its computing paradigm. *IEEE Potentials, 24*, 13–15.

Liu, J., Li, Z., Hu, X., & Chen, Y. (2009). Biclustering of microarray data with MOSPO based on crowding distance. *BMC Bioinformatics, 10*, S9.

Liu, Y., Qin, Z., Shi, Z., & Lu, J. (2007). Center particle swarm optimization. *Neurocomputing, 70*, 672–679.

Lu, H., Zhang, H. M., & Ma, L. H. (2006). A new optimization algorithm based on chaos. *Journal of Zhejiang University Science A, 7*, 539–542.

Maimon, O. Z., & Rokach, L. (2005). *Data mining and knowledge discovery handbook*. New York: Springer.

May, R. M. (1976). Simple mathematical models with very complicated dynamics. *Nature, 261*, 459–467.

Murthy, C. A., & Chowdhury, N. (1996). In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters, 17*, 825–832.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal, 7*, 308–313.

Omran, M., Engelbrecht, A. P., & Salman, A. (2005). Particle swarm optimization method for image clustering. *International Journal on Pattern Recognition and Artificial Intelligence, 19*, 297–322.

Paterlini, S., & Krink, T. (2006). Differential evolution and particle swarm optimisation in partitional clustering. *Computational Statistics and Data Analysis, 50*, 1220–1247.

Schuster, H. G., & Just, W. (2005). *Deterministic chaos: An introduction*. Weinheim: Wiley-VCH Verlag GmbH.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *The 1998 IEEE international conference on evolutionary computation proceedings* (pp. 69–73).

Singh, N., & Sinha, A. (2009). Chaos based multiple image encryption using multiple canonical transforms. *Optics & Laser Technology, 42*, 724–731.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters, 85*, 317–325.

Xiang, T., Liao, X., & Wong, K. (2007). An improved particle swarm optimization algorithm combined with piecewise linear chaotic map. *Applied Mathematics and Computation, 190*, 1637–1645.