Information Sciences xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Integration of particle swarm optimization and genetic algorithm for dynamic clustering

R.J. Kuo [a,*], Y.J. Syu [b], Zhen-Yao Chen [c], F.C. Tien [d]

[a] Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan, ROC
[b] Vanguard International Semiconductor Corporation, Hsinchu, Taiwan, ROC
[c] Department of Business Administration, De Lin Institute of Technology, New Taipei City, Taiwan, ROC
[d] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei, Taiwan, ROC

ARTICLE INFO

Article history:
Received 16 April 2010
Received in revised form 21 July 2011
Accepted 7 January 2012
Available online xxxx

Keywords:
Cluster analysis
Dynamic clustering
Particle swarm optimization algorithm
Genetic algorithm

ABSTRACT

Although the algorithms for cluster analysis are continually improving, most clustering algorithms still need to set the number of clusters. Thus, this study proposes a novel dynamic clustering approach based on particle swarm optimization (PSO) and genetic algorithm (GA) (DCPG) algorithm. The proposed DCPG algorithm can automatically cluster data by examining the data without a pre-specified number of clusters. The computational results of four benchmark data sets indicate that the DCPG algorithm has better validity and stability than the dynamic clustering approach based on binary-PSO (DCPSO) and the dynamic clustering approach based on GA (DCGA) algorithms. Furthermore, the DCPG algorithm is applied to cluster the bills of material (BOM) for the Advantech Company in Taiwan. The clustering results can be used to categorize products which share the same materials into clusters.

1. Introduction

Cluster analysis is an important data processing procedure in data mining. Recognition of natural clusters in data not only helps us get an insight into the structure of the data, but also provides a base for constructing data prediction, or classification, model [36]. With the rapid development of scientific technology and increase of information, more and more data can be collected. To acquire available information from within enormous pools of data to facilitate decision making requires the use of data mining. Cluster analysis is an important technique for data mining. In recent years, clustering has been widely applied in various fields, such as graphic recognition, machine learning and market analysis. The main goal is to cluster similar data into a group and gather together clusters with high similarity. Through clustering, valuable information such as data distribution and characteristics can be acquired from enormous quantities of data, and hidden information can be extracted by reducing data complexity. Therefore, the results obtained from clustering can be used to effectively solve problems or help make decisions.

To date, development of clustering has focused on data processing speed and efficiency. Most clustering methods, like K-means, need to first set the number of clusters. How appropriate the number of clusters is may affect the clustering results. For example, a smaller number of clusters can help clearly identify the original data structure but key hidden information may not be obtained. On the other hand, using too many clusters can result in high homogeneity of the same cluster set, while the original data structure is ignored. Thus, this study proposes a dynamic clustering approach based on particle

* Corresponding author. Tel.: +886 2 27376328; fax: +886 2 27376344.
E-mail address: rjkuo@mail.ntust.edu.tw (R.J. Kuo).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.01.021

swarm optimization (PSO) and genetic algorithm (GA) (DCPG) algorithm. The DCPG algorithm can automatically determine the optimal number of clusters and simultaneously cluster the data set with minimal user interference.

For validation, this study utilizes the Iris, Wine, Glass, and Vowel benchmark data sets. The experimental results show that the proposed DCPG algorithm outperforms other algorithms. In addition, the DCPG algorithm was applied to data of the Advantech Company in Taiwan, an internationally well-known industrial personal computer (IPC) manufacturer to cluster the bills of materials (BOM) data for all machine types. Then, during the scheduling process, similar orders can be scheduled together in order to reduce the machine changeover time, as orders belonging to the same cluster can share more similar materials.

The rest of this paper is organized as follows. Section 2 gives general background related to this study, while the proposed clustering algorithm is presented in Section 3. Sections 4 and 5 provide simulation results and model evaluation results, respectively. Finally, concluding remarks are made in Section 6.

2. Background

This section briefly introduces cluster analysis, PSO, GA, and the hybrid of PSO and GA.

2.1. Cluster analysis

Data clustering is an exploratory or descriptive process for data analysis, where similar objects or data points are identified and grouped into a set of objects called a cluster [29,34]. The clustering problem is defined as the problem of classifying a collection of objects into a set of natural clusters without any priori knowledge. The main goal of cluster analysis, an unsupervised classification technique [2], is to analyze similarity of data and divide it into several cluster sets of similar data. In terms of internal homogeneity and external separation, data within a cluster has high similarity, but there is less similarity between clusters. In other words, within-cluster variance is smaller, and between-cluster variance is larger. Therefore, clustering is vitally important for data mining. During mining, the correlations and indications of data are unknown within very large amount of data. As a result, clustering is utilized to divide the data into several cluster sets for analysis and reduce data complexity. Generally, clustering has several steps [60]: (1) proper features are selected from data as the basis for clustering; (2) a suitable clustering algorithm is selected based on data type; and (3) the evaluation principle determines the clustering results, which are provided for experts to interpret.

Basically, clustering methods can be hierarchical or partitional [15,35]. Of the unsupervised clustering algorithms, the hierarchical method is most typical. In the hierarchical method, data form a hierarchical structure and relationships among cluster sets are displayed as a tree structure. These methods can be divided into agglomerative and divisive approaches. In contract, the current study focuses on a partitional clustering method. But the problem with partitional methods is that it is necessary to set the number of clusters before clustering, as in K-means [14] or K-medoids [24]. In addition, Gath and Geva [16] proposed an unsupervised clustering algorithm based on the combination of fuzzy C-means and fuzzy maximum likelihood estimation. Lorette et al. [38] proposed an algorithm based on fuzzy clustering to dynamically determine the number of clusters in a data set. Furthermore, SYNERACT [20], an alternative approach to ISODATA [3] combining K-means with hierarchical descending approaches, does not require specifying the number of clusters. Based on K-means, other unsupervised clustering algorithms have also been developed, such as X-means [47] and G-means [18].

Furthermore, an artificial neural network (ANN) can be applied for unsupervised clustering. ANNs are systems derived from models of neurophysiology. In general, they are collections of simple nonlinear computing elements whose inputs and outputs are linked to form networks. ANN-based clustering has been dominated by self-organizing feature map (SOM) and adaptive resonance theory (ART). There are also some important representative algorithms including SOM [28], ART1 [6], ART2 [7], and fuzzy ART [8]. Moreover, Lee and Antonsson [33] applied an evolution strategy to dynamically cluster data sets, using variable length genomes to search for both cluster centroids and the number of clusters. Consequently, Kuo et al. [30] presented an ant colony-based clustering method for case clustering. Omran et al. [44] proposed a new dynamic clustering approach based on binary-PSO (DCPSO) algorithm. Paterlinia and Krink [46] applied PSO for partitional clustering. In addition, Ouadfel et al. [45] presented a modified PSO algorithm for automatic image clustering.

According to these studies, there are many different kinds of clustering algorithms which can be applied. However, it is difficult to decide the number of clusters for unknown data. As a result, automatic clustering algorithm is developed here so that it is not necessary to set the number of clusters in advance. Such an algorithm can directly generate a suitable number of clusters. In addition, it is critical to have a suitable measure for the effectiveness of a clustering method. Though some measures have been proposed in the area of classification, like Wallace's information measure [57], few have been proposed for clustering algorithms. The most frequent applied measures are cohesion, within-cluster variance, separation, and between-cluster variance [54]. The Silhouette coefficient can also be used [50].

2.2. Particle swarm optimization (PSO) algorithm

By observing human decision-making behavior, Boyd and Richerson [5] found that humans make decisions based on two kinds of information, their own experiences and other people's experience, and they proposed individual learning and the transmission of culture, namely, the PSO concept. Next, Eberhart and Kennedy [12] observed birds or fish looking for food and proposed PSO to describe the behavior of a swarm of birds that did not initially know, where to find food. When an individual bird knew the direction to food, it transmitted that information to its group, thus correcting the direction of the other birds and allowing them to advance toward the food.

In PSO [12], each feasible solution is called a particle and each particle corresponds to a fitness value via an objective function. The fitness value is determined by the objective function; each particle has its own velocity (V_{id}) and position (X_{id}), and is corrected by individual and group flying experience. Thus, each particle advances towards a personal best (P_{id}) and a global best (P_{gd}). At the very start, an algorithm has to randomly generate the initial velocity (V_{id}) and position (X_{id}) of each particle. Each particle can memorize its own optimal fitness value, called its personal best, when searching for the optimal solution. This mechanism is a cognition model. In addition, personal best values are further compared to determine the best one, which is called the global best (P_{gd}). Meanwhile, the velocity and position of each particle are corrected by the personal best and global best, referred to as a social model. The optimal solution is determined by iterative calculation.

The earliest velocity update rule of PSO was the maximum velocity (V_{max}) method [12]. In this method, some parameters and their meaning are introduced: c_1 is the learning factor of the cognition model, representing individual knowledge of the searching space; c_2 is the learning factor of the social model, indicating the information sharing and cooperation between one particle and another. These two parameters are used to control flying speed and are important for velocity and position updates. Generally $c_1 = c_2 = 2$, but Suganthan [52] indicated that it is good for the search efficiency of the particle when $c_1 \neq c_2$. Besides, $rand_1$ and $rand_2$ are the random variables between 0 and 1. In such case, these random variables are used in the velocity update equation. Therefore, the velocity update equation (Eq.) is shown as follows:

$$V_{id}^{new} = w \times V_{id}^{old} + c_1 \times rand_{1d} \times \left(P_{id} - X_{id}^{old}\right) + c_2 \times rand_{2d} \times \left(P_{gd} - X_{id}^{old}\right) \tag{1}$$

$$X_{id}^{new} = X_{id}^{old} + V_{id}^{new} \tag{2}$$

$$\text{if } V_{id} > V_{max} \text{ then } V_{id} = V_{max} \tag{3}$$

$$\text{else if } V_{id} < -V_{max} \text{ then } V_{id} = -V_{max} \tag{4}$$

In this method, each particle is limited by the V_{max} set by the user to control the flight velocity of the particle. When V_{max} is very large, the velocity of the global search can be increased. On the other hand, a small V_{max} is effective in searching for the regional solution. Now, many improved velocity update rules have been developed, including the inertia weight (W) method [51], constriction factor method [10], guaranteed convergence method [4], improvement social model [62], global–local best value based PSO method, and GLbest-PSO method [1]. In addition,Montes de Oca et al. [42] proposed a novel PSO algorithm combining a number of algorithmic components that showed distinct advantages for optimization speed and reliability in their experimental study. This algorithm is called Frankenstein's PSO.

In addition to these methods, Kennedy and Eberhart [26] proposed the binary-PSO (BPSO), which is applied to optimization problems requiring discrete or binary variables for solutions. The representative problem of this type is scheduling. A novel optimization algorithm, based on a modified binary-PSO with mutation (MBPSOM) combined with a support vector machine (SVM), was proposed to select the fault feature variables for fault diagnosis [58]. Chuang et al. [9] presented improved BPSO for feature selection using gene expression data. In addition, Xie et al. [59] suggested the distributive-PSO (DPSO) algorithm. The DPSO algorithm is used to solve problems in which it is easy for the PSO to fall into a local value from which it cannot escape. In particle searching, the DPSO algorithm is utilized to randomly select particles for mutation. Liu et al. [37] proposed PSO with mutation based on similarity, in which similarity and collectivity are introduced. Most recently, Nani and Lumini [43] used the Nearest Neighbor approach for prototype reduction using PSO and the found the Nearest Neighbor approach to be good. In order to achieve a dynamic clustering, where the optimum number of clusters is also determined within the process, the so-called multi-dimensional PSO (MDPSO) method extends the native structure of PSO particles in such a way that they can make inter-dimensional pass with a dedicated dimensional PSO process [27]. In addition, Ouadfel et al. [45] presented an automatic clustering using a modified PSO (ACMPSO) algorithm. Also, Martinez et al. [40] proposed a swarm intelligence feature selection algorithm based on the initialization and update of only a subset of particles in the swarm.

2.3. Genetic algorithm (GA)

In addition to the PSO algorithm, the GA is also often used. GA is based on the idea of the "survival of the fittest in natural selection" proposed by Darwin. Proposed by Holland [19], this algorithm simulates the biological genetic evolution process. That is, GA consists of selection, reproduction, crossover, and mutation operators. Excellent parent generation passes genes down to offspring via this mechanism to quickly find optimal solutions during searching. GAs have been applied to many applications. For instance, Maravall et al. [39] proposed a hybrid model of GAs with reinforcement learning for robotic controllers.

In summary, PSO and GA algorithms have many common points, such as optimization based on random searches, random production of initial solutions and use of fitness values as evaluation indicators.

2.4. Hybrid of PSO and GA algorithms

The difference between PSO and GA is that PSO lacks crossover and mutation and more easily to falls into local optimal solutions. But PSO can memorize the global best and affect the movement of other particles, resulting in quick convergence and falling into a local optimal solution. In addition, the GA shares information through chromosomes. Some studies have suggested that hybrids of PSO and GA algorithms can obtain better solutions discussed as follows.

Robinson et al. [49] suggested the PSO–GA algorithm, which used the optimal solution of PSO as initial parent of GA. Du et al. [11] put forward the GA–PSO and applied it to an ANN. They substituted an operator of GA into PSO and updated the position to achieve three advantages when using GA–PSO: increased searching capability, ability to escape from local optimal solutions, and as well as high accuracy. The aforementioned two methods use GA after PSO, and another hybrid method features the simultaneous operation of PSO and GA for production of the next iterative individual. Next, Kao and Zahara [23] suggested a revision to the GA–PSO algorithm, dividing the randomly generated parents into the one with better fitness value and the one with worse fitness value. GA is carried out for one with better fitness value, and PSO is performed for the one with worse fitness value. Then, the result calculated from GA is used to adjust PSO individuals for update action. The GA–PSO algorithm integrates the individual evolution of GA and self improvement of PSO algorithm.

In addition to combining PSO in with the operators of GA, elitist selection can be adopted to increase algorithm efficiency. Accordingly, Juang [22] proposed the hybrid of GA and PSO (HGAPSO) algorithm, which discards particles with worse fitness value and calls the particles with better fitness value as elitist. This method can increase the influence of the particles with better fitness value, and lead to faster convergence. Consequently, Kuo et al. [31] proposed a hybrid of particle swarm and genetic algorithm based optimization (HPSGO) algorithm to improve performance of a radial basis function neural network (RBFnn). In the case of a limitation of the local optimal solution, GA can be utilized to change chromosomes and escape the local optimal solution and quickly attain the global optimal solution. The HPSGO algorithm was applied to sales forecasting problems, showing that the hybrid algorithm is superior to PSO or GA. Unler et al. [56] also presented a hybrid filter-wrapper feature subset selection algorithm based on minimum redundancy maximum relevance and PSO.

3. Methodology

The proposed DCPG algorithm in this study is referred to as the DCPSO algorithm [44] and it has crossover and mutation operators of GA added. We expect that the hybrid algorithm can increase global search capabilities and escape from the local optimal solution. The DCPSO algorithm [44] uses binary-PSO for calculation. PSO has memory and can make corrections based on current optimal solutions with quick convergence velocity. However, when the optimal particle is a local optimal solution, the global optimal solution cannot be found. Thus, this study added GA to overcome this drawback. Thus a parent is generated through PSO, and that parent goes through crossover and mutation operators in GA to generate another parent. Finally, the next iterative parent is generated by elitist selection. This selection process lasts until conditions are met, and then K-means is utilized for correction of the particle centroid. This will obtain better results than using only PSO or GA. Fig. 1 illustrates the flowchart for the proposed DCPG algorithm.

Basically, a pool of cluster centroids, M, is randomly chosen from Z, the set of data points, for the proposed DCPG algorithm. The swarm of particles, S, is then randomly initiated. The length of particle will be equal to number of cluster centroids, M. If the bit is 1, then the corresponding point is the centroid; otherwise, it is not selected as centroid. Then, PSO and GA are applied to find the better cluster centroids, Ma, from M from a specified number of iterations. Then K-means is employed to fine-tune the result. Then, M is set to $M_a \cup M_b$, where M_b is the randomly chosen set of centroids from Z. But, the number of M is fixed. The algorithm is then repeated using the new M. If the specified number of iterations is satisfied, then the process stops and M_a will be the final solution. The evolutionary steps for the DCPG algorithm are as follows.

Step 1: Set up the values of parameters, including the population size (number of particles), inertia weight (W), maximum velocity (V_{max}), learning factors (c_1, c_2), the maximal number of clusters (N_c), crossover rate and mutation rate.

Step 2: Generate randomly N_c centers of clusters in four benchmark data sets.

Step 3: Generate randomly the initial position (X_{id}) and initial velocity (V_{id}) of individual particle in population through DCPSO algorithm [44], where

$$X_{id} = (x_{i1}, \ldots, x_{ik}, \ldots, x_{iN_C}), \tag{5}$$

in which, $x_{ik} \sim U(0,1)$: $x_{ik} = 1$, which indicates that the ith particle belongs to the kth cluster; $x_{ik} = 0$, which indicates that the ith particle does not belong to the kth cluster and N_c is in the range of $[0,1]$.

Step 4: Calculate the fitness values of all particles to measure their performance. The fitness value calculation for each particle is based on the following steps:

(1) Partition data based on the centroids shown in the particle by assigning each data point to the closest (in terms of the Euclidean distance) cluster.

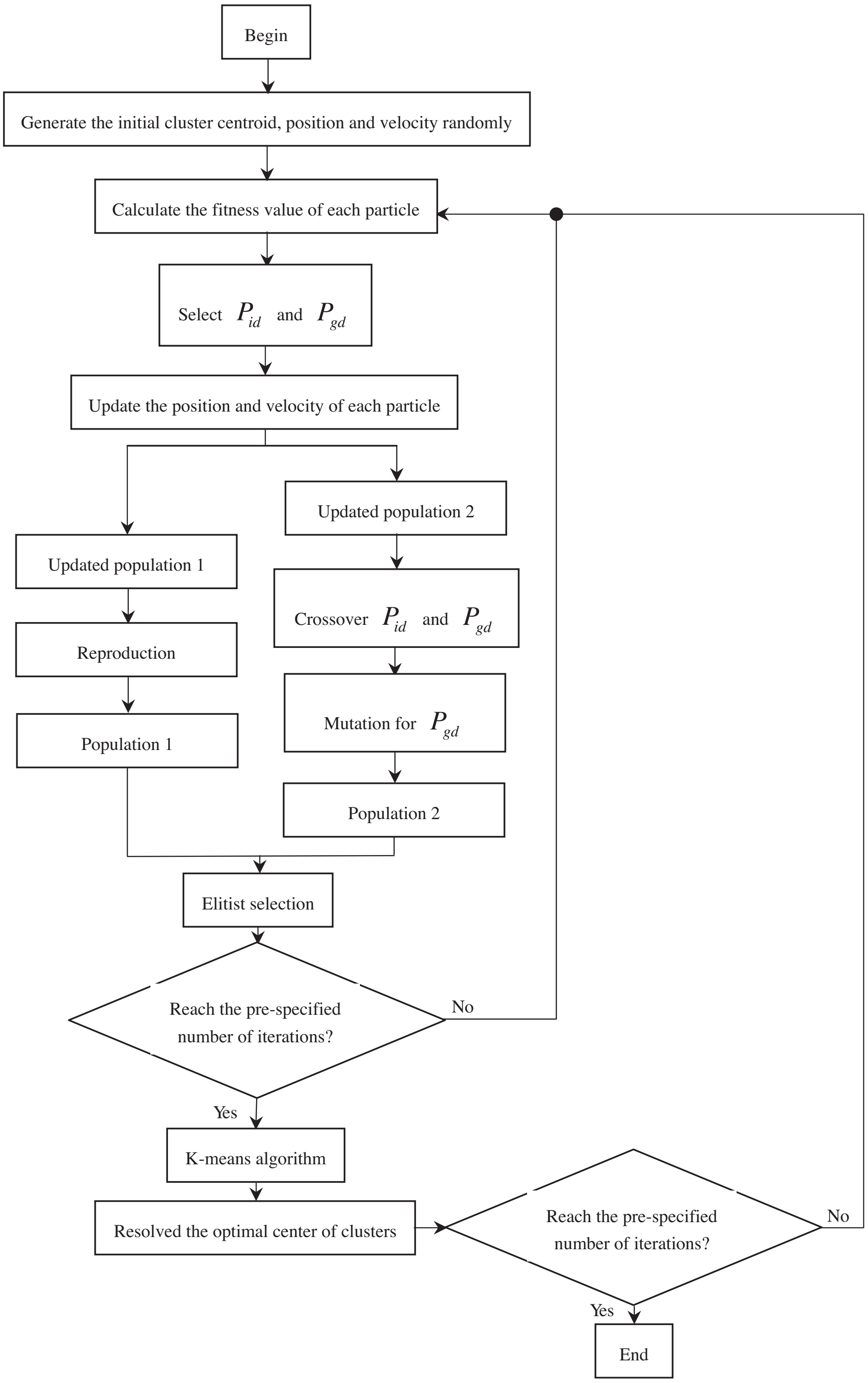

Begin
Generate the initial cluster centroid, position and velocity randomly
Calculate the fitness value of each particle
Select P_{id} and P_{gd}
Update the position and velocity of each particle
Updated population 2
Updated population 1
Crossover P_{id} and P_{gd}
Reproduction
Mutation for P_{gd}
Population 1
Population 2
Elitist selection
Reach the pre-specified number of iterations?
No
Yes
K-means algorithm
Resolved the optimal center of clusters
Reach the pre-specified number of iterations?
No
Yes
End

Fig. 1. Flowchart for the proposed DCPG algorithm.

(2) Calculate the fitness value of each particle. The idea is to calculate the sum of Euclidian distance between each data point to its corresponding centroid for each cluster. Then sum them to obtain the fitness value for the particle. The form is shown as follows:

$$fitness\ value = \sum_{k=1}^{N_c} \left(\sum_{\forall Z \in n_{ik}} \|Z - M_i\| \right) \tag{6}$$

Step 5: Select P_{id} and P_{gd}.

Step 6: Adjust the velocity (V_{id}) and position (X_{id}) of the individual particle through adopting the updating rule of DCPSO algorithm [44] as follows:

$$V_{id}^{new} = w \times V_{id}^{old} + c_1 \times rand_{1d} \times (P_{id} - X_{id}^{old}) + c_2 \times rand_{2d} \times \left(P_{gd} - X_{id}^{old}\right), \tag{7}$$

in which

$$sigmoid(V_{id}^{new}) = \frac{1}{1 + e^{-V_{id}^{new}}} \tag{8}$$

$$X_{id}^{new} = \begin{cases} 1, & if\ rand() < sigmoid(V_{id}^{new}) \\ 0, & else \end{cases} \tag{9}$$

Step 7: Perform (1) and (2) for the updated parent in Step 6:
(1) Copy all particles to generate population 1.
(2) Perform two-point crossover for P_{id} and P_{gd} in Step 5 and mutation for P_{gd}. This can generate the population 2.

Step 8: Combine populations 1 and 2 and calculate the fitness value of each particle.

Step 9: Perform elitist selection for populations 1 and 2 to generate the next iterative population.

Step 10: Return to Step 4 until the pre-specified number of iterations is satisfied.

Step 11: Adjust M_{jk} by applying the K-means clustering algorithm. Where M_{jk} is the number of clusters which was firstly grouped into N_c clusters then refined by K-means algorithm.

Step 12: Randomly reinitialize M_{rk} from the set of data points and set $M = M_{rk} \cup M_{jk}$. In which, M is the union set of M_{rk} and M_{jk}. Moreover, M_{rk} is the refined point of center for $(N_c - x_{ik})$ clusters.

Step 13: The DCPG algorithm will not stop returning to Step 4 until a pre-specified number of iterations is satisfied.

The PSO algorithm uses a population (swarm) of solutions (particles), which follows certain stochastic rules in order to emulate the behavior of biological organisms. The rules are derived from the observation that the behavior of the members of a group is often influenced by the group leaders while sharing knowledge between members attains an evolutionary advantage [17]. Moreover, the GA is appropriate for large-sized, nonlinear space problems in which a solution is unpredictable. Relying on multi-point search and algorithmic features, it is not easy to fall into local optimal solutions but it can converge to a universal optimal solution [32].

4. Simulation results

Section 3 has presented the DCPG algorithm. Next, four benchmark data sets were used in cluster analysis and the results from the DCPG algorithm are compared among DCPSO [44], ACMPSO [45], and DCGA algorithm to verify the accuracy of the DCPG algorithm. In addition, the stop condition for algorithm execution is the pre-specified number of iterations. The inner and outer loops will run 100 and 5 iterations, respectively.

4.1. Data sets

The benchmark data sets used in this experiment were derived from information (http://www.ics.uci.edu/~mlearn/databases) provided by the Department of Information and Computer Science at UCI. The reference data sets included Iris, Wine, Glass, and Vowel data sets.

4.2. Data preprocessing

This study employed distance as the measurement index. Features of the data sets used had great differences, and great variation may lead to deviation in distance calculation. If the original data were used directly for clustering, the clustering results would have been affected. Therefore, the original data was normalized, so that in the same data set, different features were measured by the same standard. A conversion procedure was used to find out the maximum and minimum values from the original data matrix, and the difference between them was extreme. Features of each original piece of data were deducted by the minimum value of the feature, and then divided by the extreme difference, thus completing the normalization.

4.3. Experimental results and analysis

For the DCPG in this study, it is not necessary to set the number of clusters. However, this kind of dynamic clustering has a drawback in that the number of clusters is randomly affected by the initial centroids selected. For this study, in addition to the two indicators mentioned in Section 3, VI value aand Index value, the average number of clusters and actual number of classified data sets were determined. Table 1 lists the parameter setup.

A parameter for the maximum number of clusters must initially be set, after which, the optimal solution can be found by dynamic clustering. Zhang et al. [61] suggested that the maximum number of clusters should not exceed the square root of the number of data. Thus, the maximum number of clusters was set to 13 for Iris and Wine, 15 for Glass, and 30 for Vowel.

4.3.1. Algorithm evaluation

An alternative approach to measure cluster quality is based on how well connected the objects in the cluster are to one another [34]. Traditional clustering emphasizes clustering data with high similarity in the same cluster set and makes every effort to find the shortest distance between data and the cluster centroid. Kaufman and Rousseeuw [24] suggested that Sum of Euclidean Distance (SED) is better than MSE for measuring cluster analysis results. In our dynamic clustering, the distances within each cluster and the distances between clusters were considered in the measurement index. Consequently, the index modified by Turi [55] was adopted, as follows:

$$VI = (c \times N(0,1) + 1) \times \frac{intra}{inter}, \tag{10}$$

where $(c \times N(0,1) + 1)$ is regarded as a punishment value to avoid having too few clusters; c is a constant set to 30; and $N(0,1)$ stands for the Gaussian function of the number of clusters. Turi [55] indicated that dynamic clustering results falling in the interval of [2, maximum number of clusters]. The punishment term can prevent simpler data from always being grouped into the two clusters, enabling data to move to a suitable number of clusters.

In this experiment, since Iris and Wine have few clusters, $N(0,1)$ is adopted to calculate the *VI* index. As Glass and Vowel have more clusters, $N(2,1)$ is adopted in place of $N(0,1)$ to calculate the *VI* index. Intra, which is shown as follows, is the average intra distance:

$$intra = \frac{1}{N_p}\sum_{k=1}^{K}\sum_{u\in C_k}\|u - m_k\|^2. \tag{11}$$

Its purpose is to calculate the intensity of intra-clusters. The approach is to calculate the Euclidian distance of the data point and the center of cluster, sum up all the shortest distance of each data point and the center of cluster, and then divide by the total data tuples (N_p). If the *intra* value is smaller (larger), the clustering efficiency for algorithm is considered better (worse). Lastly, the *inter* is the distance between two clusters and the formula is shown as Eq. (12):

$$\begin{aligned} &inter = \min\{\|m_k - m_{kk}\|^2\} \\ &\forall\ k = 1,2,\ldots,K-1 \text{ and } kk = k+1,\ldots,K. \end{aligned} \tag{12}$$

This part, the inter, focuses on the minimal distance between clusters. The distance from each cluster centroid to the centroid of other clusters is calculated to find the minimum value. Calculations show that the distance between clusters is very far, though the distance has a minimum value. Currently, few clustering methods consider the distance between clusters, but all hope that the clustering result has a small distance within and large distance between. Thus, two requirements are considered for calculating the VI value through the ratio of intra to inter. The VI value is the training measurement index of an algorithm, and it is not affected by the number of clusters, and so whether the clustering result is good or bad cannot be directly judged according to VI value. Thus, the sum of squares within and the sum of squares between are calculated, and the result from the division of both serves as the index to determine the number of clusters. The equations of the sum of squares within (SSW), sum of squares between (SSB), and index are as follows:

$$SSW = \sum\sum(X_{ij} - \overline{X}_{kj})^2, \tag{13}$$

Table 1
The setting of parameter values for the proposed DCPG algorithm.

The description of parameter	Value
The total number of training (epochs)	500 (100 × 5)
Population size	20
PSO algorithm part:	
Inertia weight (*W*)	0.72
Learning factor (c_1, c_2)	$c_1 = c_2 = 1.49$
The maximum velocity of each particle (V_{max})	3
GA algorithm part:	
Crossover rate (P_c)	1
Mutation rate (P_m)	0.05

where $\overline{X}_{kj}$ is the center of cluster.

$$SSB = \sum N_k(\overline{X}_{kj} - \overline{X}_j)^2, \tag{14}$$

where N_k is the data tuples of cluster, $\overline{X}_{kj}$ is the center of cluster, and $\overline{X}_j$ is the average of all data.

$$Index = \frac{SSW}{SSB}. \tag{15}$$

In order to validate the proposed DCPG algorithm, it is compared to DCPSO, ACMPSO and DCGA algorithms. For the DCPSO algorithm [44], a pool of cluster centroids (M) is randomly chosen from the sets of data points (Z). The swarm of particles (S) is then randomly initialized. Binary-PSO (BPSO) [44] is then applied to find the best set of cluster centroids (M_b) from M. The K-means algorithm is applied to M_b in order to refine the chosen centroids. M is then set to $M_b \cup M_r$, which is a randomly chosen set of centroids from Z. In which, M_r is the refined point of center for ($N_c - x_{ik}$) clusters. The DCPG algorithm is then repeated using the new M. When the termination criteria are satisfied, M_b will be the optimum set of cluster centroids. For the DCGA algorithm, the initial chromosome generation is similar to that of DCPSO algorithm, and the crossover and mutation operators are implemented sequentially. The K-means algorithm is then applied to the chromosomes. The DCPG algorithm is then repeated using the new chromosomes. If the termination criteria are met, the best chromosome will be the optimum set of cluster centroids.

4.3.2. Algorithm result analysis

In this study, tests of each clustering algorithms were performed 30 times for four benchmark data sets, and the average values of the tested results of the 30 tests were used as experimental results. Through the VI and *Index* values as the measurement indexes, the smaller (greater) the two values were, the better (worse) clustering results of the algorithm were.

For verification, comparisons were made to the *Index* values and the number of clusters. Due to the random generation of the initial solution, evaluation was based on the average values of the results from the 30 tests and standard deviation (SD). The VI value indicates that the training process of the algorithm reaches convergence, and the *Index* value is the evaluation index based on the final clustering result. Regarding the number of clusters, the closer to the benchmark data set the number was, the better the clustering result of the algorithm obtained. Tables 2–4 illustrate the comparison results, in which, the Z value is the value of Z distribution, while the P value is the probability value of it being less than the Z value.

The known number of clusters is three for the Iris data set, the test result of 30 times for the DCPG algorithm was just 3 and it showed a stable situation, implying the smaller standard deviation. Off all the algorithms, ACMPSO is the worst. The VI values among four algorithms are similar for the Iris data set, but results obtained by the DCPG algorithm indicate that the VI value is more stable. In addition, the index of the DCGA algorithm is a little smaller than that of the DCPG algorithm. Based on SD, the DCPG algorithm has smaller SD, which means that the DCPG algorithm is better and steadier for clustering. Furthermore, the DCPSO algorithm is weaker than the other two algorithms, and its SD is not stable. But the DCPG algorithm achieves the merits of both GA and PSO algorithms through integration.

Table 2
Number of clusters for all algorithms.

Data set evaluation	Algorithm (number of clusters ±SD)			
	DCPG	DCPSO	ACMPSO	DCGA
Iris	3 ± 0	3.06667 ± 0.24944	3.55563 ± 0.13596	3.03333 ± 0.17951
Wine	3.76667 ± 0.42295	3.63333 ± 0.60461	3.69833 ± 0.54978	3.93333 ± 0.62893
Glass	5.93333 ± 0.77172	5.43333 ± 0.61554	5.65738 ± 0.92384	5.8 ± 1.46969
Vowel	11.66667 ± 2.57337	8.3 ± 2.01908	10.33338 ± 2.83143	15 ± 3.72380

Table 3
VI and *Index* values for all algorithms.

Data set evaluation		Algorithm			
		DCPG	DCPSO	ACMPSO	DCGA
Iris	(VI value, SD)	(0.35869, 0.01435)	(0.37155, 0.02273)	(0.36632, 0.02002)	(0.37419, 0.02469)
	(Index value, SD)	(0.20631, 0.00220)	(0.20898, 0.02952)	(0.20753, 0.01669)	(0.20607, 0.00659)
Wine	(VI value, SD)	(0.58139, 0.02386)	(0.59656, 0.03555)	(0.59817, 0.03425)	(0.61523, 0.04582)
	(Index value, SD)	(0.96284, 0.05652)	(0.96922, 0.07855)	(0.97079, 0.12541)	(0.97873, 0.19404)
Glass	(VI value, SD)	(0.46991, 0.04837)	(0.53900, 0.08497)	(0.49146, 0.06368)	(0.44367, 0.04233)
	(Index value, SD)	(0.57861, 0.10041)	(0.63563, 0.11193)	(0.62757, 0.10624)	(0.67689, 0.11256)
Vowel	(VI value, SD)	(0.78194, 0.03316)	(0.74063, 0.04225)	(0.76481, 0.03749)	(0.78806, 0.03434)
	(Index value, SD)	(0.71919, 0.11894)	(0.93013, 0.16526)	(0.77151, 0.14329)	(0.61257, 0.16789)

Table 4
Comparison of the VI and Index values between two algorithms.

Data set evaluation		Algorithms		
		DCPG & DCPSO	DCPG & ACMPSO	DCPG & DCGA
Iris	VI value: (*Z* value, *P* value)	(−2.233, 0.026)	(−2.354, 0.019)	(−2.189, 0.029)
	Index value: (*Z* value, *P* value)	(−0.62, 0.535)	(−0.83, 0.043)	(−0.98, 0.327)
Wine	VI value: (*Z* value, *P* value)	(−1.804, 0.071)	(−2.637, 0.048)	(−3.046, 0.002)
	Index value: (*Z* value, *P* value)	(−0.392, 0.695)	(−0.669, 0.491)	(−0.917, 0.359)
Glass	VI value: (*Z* value, *P* value)	(−3.46, 0.001)	(−2.714, 0.075)	(−2.099, 0.036)
	Index value: (*Z* value, *P* value)	(−1.919, 0.056)	(−0.478, 0.086)	(−3.711, 0)
Vowel	VI value: (*Z* value, *P* value)	(−3.43, 0.001)	(−2.682, 0.081)	(−0.724, 0.469)
	Index value: (*Z* value, *P* value)	(−4.465, 0)	(−5.253, 0)	(−3.814, 0)

Of the UCI data sets, the Wine data set is divided into three clusters. Through the four algorithms in this study, the Wine data set was divided into three, four or five clusters. The number of clusters using DCPG algorithm was three or four. Thus, the result was steadier, but four clusters were quite common. There were sometimes five clusters using the DCPSO or DCGA algorithm, and the change was greater. In VI value comparison, the DCPG algorithm was superior to other algorithms. It was found that the stability and solution of the DCPSO algorithm were inferior to those of the DCPG algorithm. In contract, the DCGA algorithm showed poor performance in the Wine data set, regardless of the VI or average value.

The known number of clusters is 6 for the Glass data set. By using the four dynamic clustering algorithms, the Glass data set was divided into 5, 6 and 7 clusters. It was divided into 8 in one of the DCPG algorithm tests and 13 in one of the DCGA algorithm tests. As a result, the three algorithms had greater SD for the average number of clusters than the DCPSO algorithm. As for the VI value, the DCGA algorithm failed to reach the known number of clusters in only one test, but the average value was not affected. This means that the test still obtained convergence. Based on the *Index* value, the DCPG algorithm had better clustering results among other algorithms. Because the *Index* value using the DCPG algorithm was smaller and steadier than other algorithms, and the distance within cluster was the smallest and the distance between two clusters was the largest.

The known number of clusters was 11 for the Vowel data set. This set was the most complicated among the data sets studied. The average number of clusters using the DCPG algorithm was close to 11, and the greatest results were 8 and 13 clusters. Only a few results were beyond this range. The results of the DCPSO algorithm were smaller. Most were 6, 10, and 11 clusters, occurring only one time. The results of the ACMPSO algorithm were worse than those of the DCPG algorithm but better than those of the DCPSO and DCGA algorithms. The clusters of the DCGA algorithm were greater. Most of them exceeded 11 clusters, and the average value was 15 clusters. In the comparison of VI value, the DCPSO algorithm convergence value was smaller than the DCPG algorithm, but the stability of the two algorithms was similar. In addition, the DCGA algorithm had a smaller index, but its average number of clusters was up to 15, as mentioned above. However, the *Index* value was the ratio of distance within to distance between, and the *Index* value was smaller, the larger number of clusters was. Thus, the DCGA algorithm had the smallest *Index* value. Comparing all algorithms, the DCPG algorithm had the best clustering results. Therefore, this study set the value of parameters most suitable for the DCPG algorithm to achieve better clustering result.

To further assess the proposed DCPG algorithm, the Mann–Whitney U test was employed. The significant level was set as 90%. Thus, if the *P*-value was less than 0.05, then the difference was significant. For the VI value, the DCPG algorithm was significantly different from other algorithms for some data sets. For the Wine data set, the DCPG algorithm was significantly different from DCGA and ACMPSO algorithms, rather than the DCPSO algorithm. In addition, for the Glass data sets, the DCPG algorithm was significantly different only from DCPSO and DCGA algorithms, rather than the ACMPSO algorithm. Regarding the Vowel data set, the DCPG algorithm was significantly different only from the DCPSO algorithm. For the *Index* value, the DCPG algorithm was significantly different from other algorithms on the Vowel data set. This result shows that the DCPG algorithm is more suitable for data with more clusters. This kind of phenomenon is similar for the Glass data set. Basically, the proposed DCPG algorithm was not worse than other algorithms according to the statistical tests.

4.3.3. Convergence analysis for algorithm

Figs. 2–5 show the convergence of DCPG, DCPSO, ACMPSO, and DCGA algorithms in Iris, Wine, Glass and Vowel data sets. In Fig. 2, DCPG has quicker convergence than the other algorithms. Due to the simpler Iris data set, all algorithms had good convergence. The DCGA algorithm convergence fluctuates, because the cluster centroid was corrected by the K-means algorithm after GA training in the DCGA algorithm to reselect cluster centroid. Thus the initial cluster centroid is not the only factor that should be taken into account in clustering. During calculation, other data points also had the opportunity to be selected as cluster centroid. There were 100 GA training iterations in this study, and the correction action above was carried out after 100 iterations. After correction, VI value will fluctuate. The reason is that after correction by K-means algorithm, the intra-distance will be reduced. However, the inter value from the VI value was the minimum value of the distance to the cluster centroid. When inter became smaller, the VI value increased. This situation never occurred in DCPSO

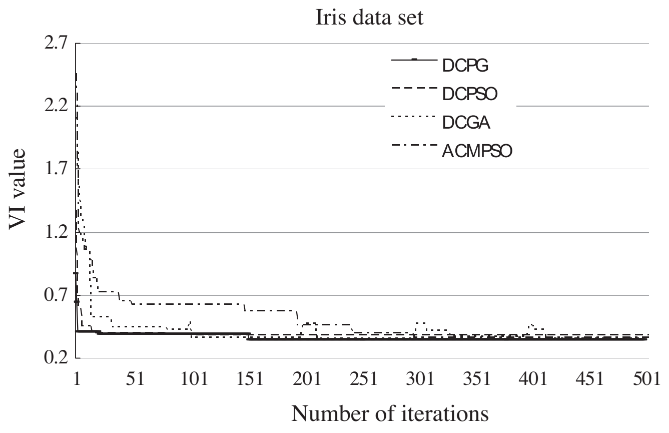


Fig. 2. Convergence of Iris data set using different algorithms.

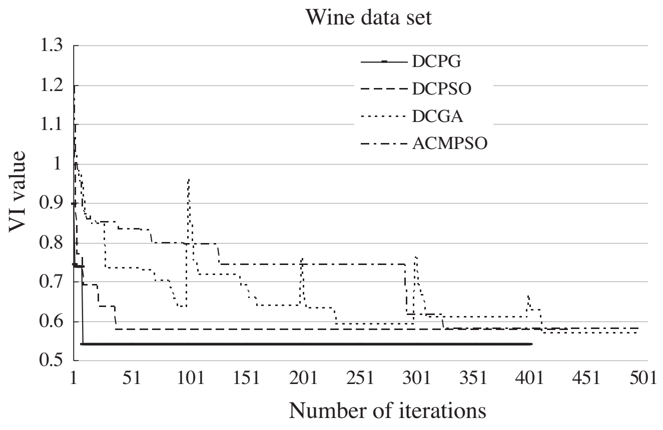


Fig. 3. Convergence of Wine data set using different algorithms.

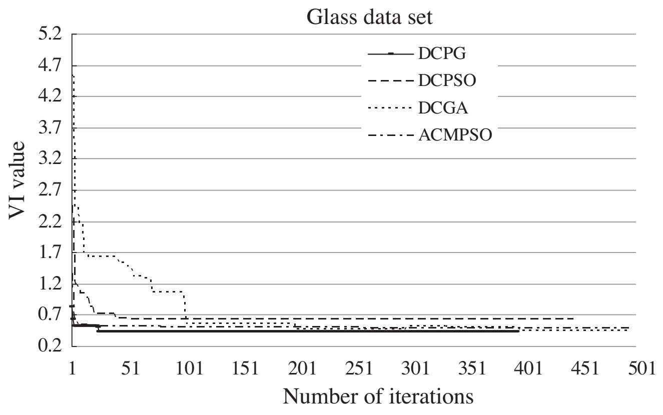


Fig. 4. Convergence of Glass data set using different algorithms.

and DCPG algorithms since PSO algorithm has a memory mechanism. When the iteration result was superior to the last iteration, replacement action was carried out, so there was no increase in the VI value.

Fig. 3 shows an optimal solution is converged in the DCPG algorithm after training 20 times, and for the DCPSO algorithm, 50 times was enough for convergence. As a result, the training had stopped before 500 iterations were reached. It is clear that the DCPG algorithm had the best performance in the Wine data set, and the performance of the DCPSO and DCGA algorithms was similar. The DCGA algorithm had greater fluctuation, but showed a downward trend in total, and convergence finally occurred.

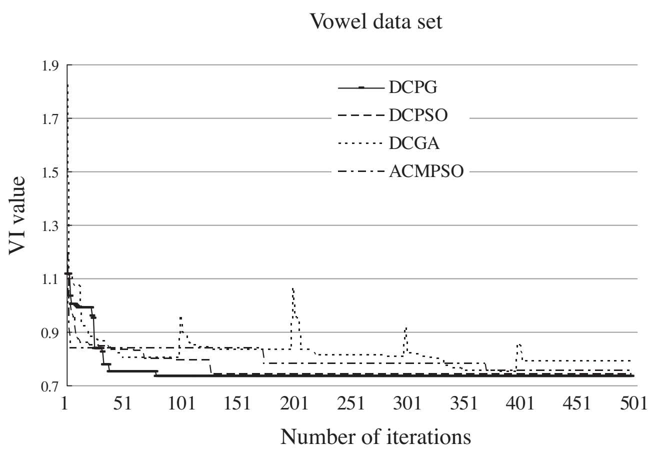


Fig. 5. Convergence of Vowel data set using different algorithms.

As shown in Fig. 4, the result for the DCGA algorithm had no fluctuation in the Glass data set, and it was smaller than the DCPSO algorithm. Table 2 shows that the average result for the DCPSO algorithm was worse than the DCGA algorithm, but the DCPSO algorithm had faster convergence. The DCPG algorithm was characterized by fast convergence, and a better convergence result than the other algorithms.

According to Fig. 5, it is clear that the DCPG algorithm was superior to other algorithms. As for the Vowel data set with higher complexity, training around 100 times allowed convergence to obtain the optimal solution. In addition, the DCPSO algorithm converged to an approximate solution, but it took more time. And the DCGA algorithm caused an increase of convergence value due to mutation operator, and this is the disadvantage of GA.

In summary, the proposed DCPG algorithm can always provide the best results of all algorithms. Regarding the computational time, though the DCPG algorithm is more complicated, implying longer computational time, it needs less iteration to find the optimal solution. Thus, the computational time will be similar. For instance, the computational time of the DCPSO and DCPG algorithms for the Iris data set is 13.868 s and 10.031 s, respectively.

4.4. Parameters Setup

To achieve robust parameter design, this study employs the Taguchi method to determine the parameter combination. Taguchi methods are statistical methods to improve the quality of manufactured goods. By using the Taguchi method, the number of experiments can be reduced by referring to orthogonal arrays. In this study, four parameters including learning factors c_1 and c_2, inertial weight W, crossover rate and mutation rate are considered. Each parameter has three levels according to references. Taguchi's orthogonal arrays are employed to implement the experimental design for Glass dataset. Thus, the L_9 orthogonal array of four control factors and three levels suggests the possible combinations, as illustrated in Table 5. Each experiment is implemented ten times and the corresponding average is calculated. The average S/N (signal to noise ratio) values are shown in Table 6, while the average S/N values for each factor under different levels are illustrated in Table 7, which indicates that learning factor has more influence on the results. Also, the best parameter combination shown in "best level" of Table 7 is as follows:

(1) Inertial weight: decreased from 0.9 to 0.4.
(2) Learning factor: c_1 is increased from 0.35 to 2.4 and c_2 is decreased from 2.4 to 0.35.
(3) Crossover rate: 60%.
(4) Mutation rate: 1%.

Thus, these parameter values will be used for the practical application, order clustering, in the next section.

Table 5
Different levels for each parameter.

Inertial weight (W)	0.72 ($W1$)[41]	Decreased from 0.9 to 0.4 ($W2$)[13]	Decreased from 1.2 to 0.4 ($W3$) [32]
Learning factor (c_1, c_2)	$c_1 = 1.49$, $c_2 = 1.49$ (L1)[25]	$c_1 = 2$, $c_2 = 1.49$ (L2) [52]	c_1: Increased from 0.35 to 2.4, c_2: Decreased from 2.4 to 0.35 (L3) [48]
Crossover rate (P_c)	1 (P_c1)	0.8 (P_c2)	0.6 (P_c3)
Mutation rate (P_m)	0.05 (P_m1)	0.01 (P_m2)	0.1(P_m3)

Table 6
L_9 orthogonal table and results.

No.	Inertial weight (W)	Learning factor (c_1, c_2)	Crossover rate (P_c)	Mutation rate (P_m)	S/N value
1	$W1$ (0.72)	L1 (1.49, 1.49)	P_c1 (1)	P_m1 (0.05)	−4.91437
2	$W1$ (0.72)	L2 (2, 1.49)	P_c2 (0.8)	P_m2 (0.01)	−4.32689
3	$W1$ (0.72)	L3 ([0.35, 2.4], [2.4, 0.35])	P_c3 (0.6)	P_m3 (0.1)	−4.30419
4	$W2$ ([0.9, 0.4])	L1 (1.49, 1.49)	P_c2 (0.8)	P_m3 (0.1)	−4.35939
5	$W2$ ([0.9, 0.4])	L2 (2, 1.49)	P_c3 (0.6)	P_m1 (0.05)	−3.5688
6	$W2$ ([0.9, 0.4])	L3 ([0.35, 2.4], [2.4, 0.35])	P_c1 (1)	P_m2 (0.01)	−3.76857
7	$W3$ ([1.2, 0.4])	L1 (1.49, 1.49)	P_c3 (0.6)	P_m2 (0.01)	−4.82313
8	$W3$ ([1.2, 0.4])	L2 (2, 1.49)	P_c1 (1)	P_m3 (0.1)	−4.89088
9	$W3$ ([1.2, 0.4])	L3 ([0.35, 2.4], [2.4, 0.35])	P_c2 (0.8)	P_m1 (0.05)	−4.61935

Table 7
S/N response values under different levels for different factors.

Evaluation item	Inertial weight (W)	Learning factor (c_1, c_2)	Crossover rate (P_c)	Mutation rate (P_m)
Max–min	0.878858	0.468262	0.292559	0.211958
Ranking	2	1	3	4
Best level	Level 2	Level 3	Level 3	Level 2

5. Model evaluation results

This section takes the production of IPCs by the Advantech Company in Taiwan as an example for demonstration. This company produces small quantities of products of different types. Consequently, it is easy for the surface mount technology (SMT) production system to encounter a bottleneck during line changeover. Before line changeover, manual supply of materials is required, and this operation is rather time-consuming. If the line changeover is not finished before the next order, the machine is left unused, and the utilization rate of production is lower. Thus, this study utilized the cluster analysis algorithm to classify products with similarity into the same cluster based on all BOM data and characteristics. In addition, the shared materials of the products in the cluster were also found. As a result, engineers could fix the bin level of SMT machines for shared materials. Moreover, production management engineers could arrange the products in the same clusters together for production and select suitable work order combinations to simplify manual supply, thus, tying materials together to shorten SMT changeover time.

5.1. Case study

The most important procedure in the assembly of IPCs is the assembly of printed circuit boards. SMT is applied to assemble printed circuit board components to increase functional density and place more components on the same area to reduce cost [21]. This technology uses automatic equipment to place surface mount components (SMC) such as transistors, resistances, diodes, etc., on the paste stencil circuit board, and fix the components on the board by heating. The production process can be divided into three steps: paste stencil printing, component placement, and solder reflow. In addition, cleaning depends on the situation of the printed circuit board. Component placement is a time-consuming step which requires accuracy.

5.2. Experimental procedure

The data used for this demonstration was derived from the product type and material report provided by Advantech. Since there is an enormous amount of data, the reported data must be converted into a data type suitable for cluster analysis before clustering. However, the characteristic dimension of the product type is quite high. Therefore, the principal component analysis (PCA) is first carried out for data to reduce the dimension of product type. After the PCA procedure, the data can be used for cluster analysis and the results computed by DCPG, DCPSO, ACMPSO, and DCGA algorithms are evaluated so as to find the optimal clustering algorithm. This algorithm can be utilized to determine the optimal number of clusters. The empirical analysis flowchart is illustrated in Fig. 6.

5.3. Source and preprocessing of empirical data

The empirical data were derived from the material reports provided by Advantech Company from January 1999 to May 2005. There are 2826 types of products and 2516 types of materials. Based on the relationships between product types and materials shown in Fig. 7, the reports were transformed into 0 or 1, forming a 2826 $\times$ 2516 feature matrix of products.

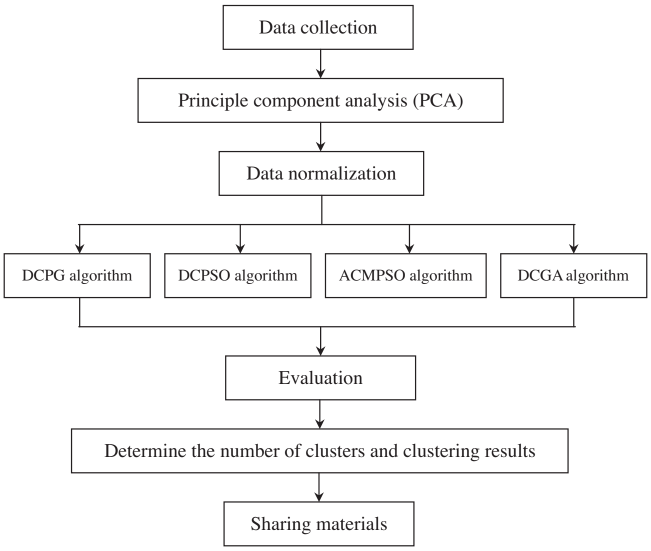
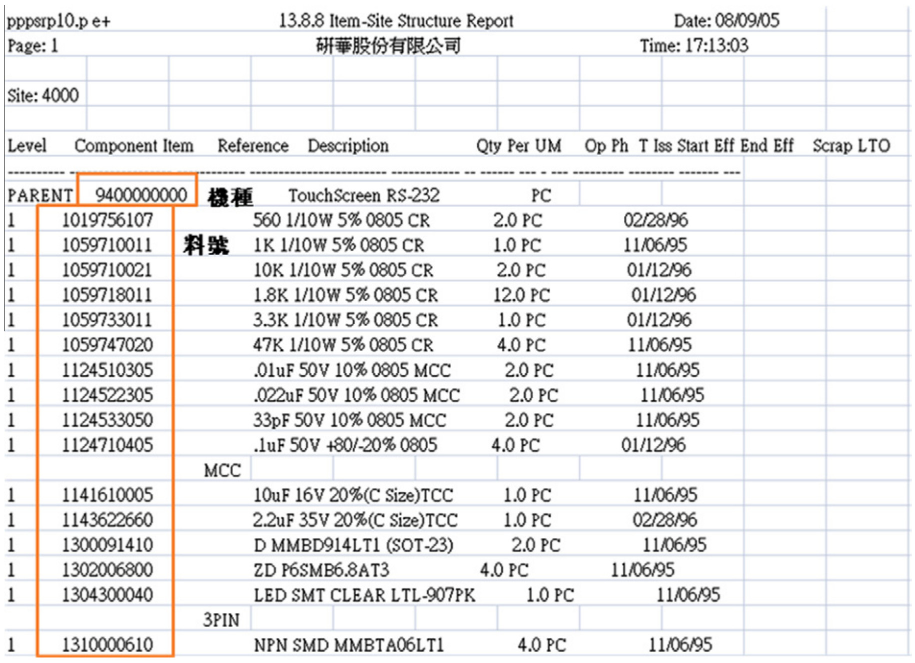


Fig. 6. Flowchart for empirical analysis in this study.



pppsrp10.p e+ 13.8.8 Item-Site Structure Report Date: 08/09/05
Page: 1 研華股份有限公司 Time: 17:13:03

Site: 4000

Level	Component Item	Reference	Description	Qty Per UM	Op Ph T Iss Start Eff End Eff	Scrap LTO
PARENT	9400000000		TouchScreen RS-232	PC		
1	1019756107		560 1/10W 5% 0805 CR	2.0 PC	02/28/96	
1	1059710011		1K 1/10W 5% 0805 CR	1.0 PC	11/06/95	
1	1059710021		10K 1/10W 5% 0805 CR	2.0 PC	01/12/96	
1	1059718011		1.8K 1/10W 5% 0805 CR	12.0 PC	01/12/96	
1	1059733011		3.3K 1/10W 5% 0805 CR	1.0 PC	01/12/96	
1	1059747020		47K 1/10W 5% 0805 CR	4.0 PC	11/06/95	
1	1124510305		.01uF 50V 10% 0805 MCC	2.0 PC	11/06/95	
1	1124522305		.022uF 50V 10% 0805 MCC	2.0 PC	11/06/95	
1	1124533050		33pF 50V 10% 0805 MCC	2.0 PC	11/06/95	
1	1124710405		.1uF 50V +80/-20% 0805	4.0 PC	01/12/96	
		MCC				
1	1141610005		10uF 16V 20%(C Size)TCC	1.0 PC	11/06/95	
1	1143622660		2.2uF 35V 20%(C Size)TCC	1.0 PC	02/28/96	
1	1300091410		D MMBD914LT1 (SOT-23)	2.0 PC	11/06/95	
1	1302006800		ZD P6SMB6.8AT3	4.0 PC	11/06/95	
1	1304300040		LED SMT CLEAR LTL-907PK	1.0 PC	11/06/95	
		3PIN				
1	1310000610		NPN SMD MMBTA06LT1	4.0 PC	11/06/95	

Fig. 7. Machine types and materials report.

For cluster analysis, processing 2516 feature values (dimensions) is rather time-consuming. In order to save calculation time and still maintain the features of product types, the PCA procedure was conducted for the feature matrix of product types to extract the principal component factors (PCFs) and to use fewer variables for interpreting data variation. In factor selection, components whose Eigen-values were greater than 1 were retained. Finally, we obtained 200 PCFs, and the percentage of cumulative explanation variation reached 81.98%. Thus, the feature matrix of product types was reduced from 2826×2516 to 2826×200, and the feature matrix of product types was turned into numerical values.

5.4. Empirical results and analysis

The 2826×200 data after PCA and reduction of dimension were tested 10 times using DCPG, DCPSO, ACMPSO and DCGA algorithms. We set the maximum number of clusters to be 53, and the number of particles as 20. The DCPG algorithm param-

Table 8
The *Index* values for all algorithms.

Evaluation	Algorithm			
	DCPG	DCPSO	ACMPSO	DCGA
Number of clusters ±SD	25.7 ± 3.06757	18.3 ± 3.22645	23.2 ± 7.16982	23.3 ± 10.55509
Index value ±SD	6.6 ± 0.96623	11.2 ± 3.06168	10.5 ± 7.08313	13.4 ± 12.49171

Table 9
Empirical analysis results in this study.

Max. number of clusters	Number of clusters	The *Index* value
15	6	49.6048
20	7	32.47995
25	8	28.6942
30	9	24.18523
35	15	9.636351
40	15	11.80893
45	25	6.733402
50	22	6.397107
55	27	5.295779
60	31	4.689919
65	34	4.577252
70	34	4.180515
75	37	4.313175
80	38	3.933179

eters were as follows: c_1 was increased from 0.35 to 2.4, c_2 was decreased from 2.4 to 0.35; Inertia weight was decreased from 0.9 to 0.4; Crossover rate was set to 60%, and Mutation rate was 1%. These setups were obtained from the Taguchi approach [53], as shown in subsection 4.4. Regarding the previously stated evaluation criterion, the VI value is the ratio of average distance within cluster to minimum distance between two clusters. Moreover, the VI value cannot interpret the clustering result. Consequently, the *Index* value was adapted for evaluation. The average *Index* value of results from the 10 time tests is summarized in Table 8.

As shown in Table 8, DCPG and DCPSO algorithms divided the data from Advantech into 25.7 and 18.3 clusters, and SD was ±3 clusters, so the clustering results were stable. However, the ACMPSO and DCGA algorithms divided the data into 23.2 and 23.3 clusters, and SD was up to 7 and 10 clusters. This means the clustering results of ACMPSO and DCGA algorithms are not stable for this practical case with complexity. In respect to the *Index* value, the average value for the proposed DCPG algorithm was the smallest, the 10 time tests showed the clustering results were steadier, followed by the ACMPSO and DCPSO algorithms. Next, the DCGA algorithm displayed unstable clustering results and thus its *Index* value was also undesirable.

This case aimed to utilize a dynamic clustering algorithm to find shared materials for products in the same cluster according to the relationships between product types and materials. If there are too few clusters, more products are classified into the same cluster and the amount of shared materials is reduced. On the contrary, there are too many clusters, changeover efficiency cannot be improved. The dynamic clustering algorithm allows the number of clusters to range from 2 clusters to the set maximum number of clusters. This study found the suitable number of clusters based on the change of maximum

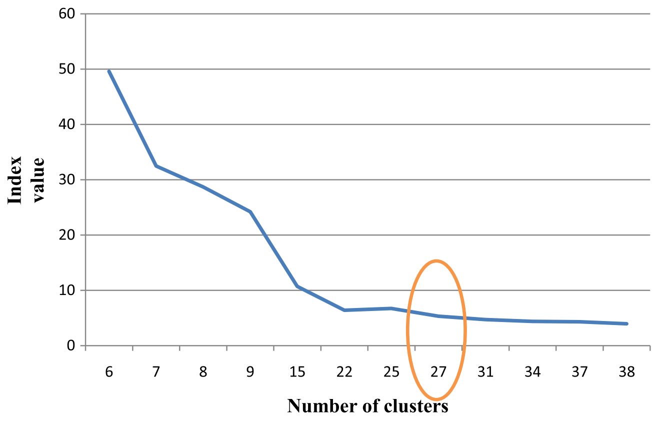


Fig. 8. Convergence curve of the *Index* value.

Table 10
The production time with and without clustering.

Scheduling method	Evaluation item	Binding time (h)	Production time (h)	Idle time (h)
FCFS	Before improvement	23.9	34.3	2.8
	After improvement	18.5	31.5	0
	Improved percentage (%)	22.6	8.2	100
SPT	Before improvement	23.9	34.9	5.6
	After improvement	19.2	31.7	0
	Improve percentage (%)	19.7	9.2	100

number of clusters. In the experiment, the maximum number of clusters was set to 15, and each time 5 clusters were added, up to 80 clusters. The *Index* value was used to evaluate the result of clustering. Moreover, the experimental results are summarized in Table 9 and the convergence curve of the *Index* value is illustrated in Fig. 8. If the clustering results are the same, the *Index* value is expressed by average value.

According to Fig. 8, when the number of clusters was 22, the *Index* value tended to be smooth. By further observation, when the result was 25, the *Index* value change significantly. However, when the result was 27, the *Index* value dropped slowly. In the first phase of cluster analysis, there were 25 clusters for the BOM from Advantech Company. Furthermore, the results obtained can be used for improving the scheduling process. This can result in smaller changeover time.

In order to further verify that scheduling similar orders together can reduce the production time, two simple scheduling methods, first-come first-served (FCFS) and shortest processing time (SPT), are employed to prove this assumption. In total, there are eleven orders used and the computational results are listed in Table 10. This reveals that production time can be decreased by 8.2% and 9.2% for FCFS and SPT, respectively.

6. Conclusions

This study proposes the DCPG algorithm to solve the problem of setting the number of clusters in advance and find out the suitable number of clusters based on the features of data. Four benchmark data sets with different numbers of clusters, dimensions, and types were used to verify that the DCPG algorithm can obtain the right number of clusters and achieve better clustering results. Moreover, the DCPG algorithm was compared to DCPSO, ACMPSO, and DCGA algorithms. Based on PSO and GA algorithms, the DCPG algorithm obtained fast convergence and better clustering results. Using the DCPG algorithm, it is not easy to fall into the local optimal solution. Moreover, this is the most stable of all algorithms. As for the empirical study, the DCPG algorithm was applied to cluster analysis for the BOM provided by Advantech Company. The proposed DCPG algorithm found a suitable number of clusters, clustering results, product clusters, and shared materials. For an SMT production line, the clustering results can allow process engineers to fix shared materials, and production engineers can arrange the products in the same cluster together for production, reducing the time needed for binding materials and SMT changeover. In the future, it may be promising to employ different binary version of PSO or different soft computing algorithms, such as an artificial immune system (AIS) algorithm.

Acknowledgement

This study was financially supported by National Science Council of Taiwan Government under contract NSC96-2416-H-011-018-MY3. Its support is great appreciated.

References

[1] M.S. Arumugam, M.V.C. Rao, On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems, Applied Soft Computing 8 (1) (2008) 324–336.
[2] H. Alizadeh, B. Minaei-Bidgoli, S.K. Amirgholipour, A new method for improving the performance of *k* nearest neighbor using clustering technique, Journal of Convergence Information Technology 4 (2) (2009) 84–92.
[3] G. Ball, D. Hall, A clustering technique for summarizing multivariate data, Behavioral Science 12 (1967) 153–155.
[4] F.V.D. Bergh, A.P. Engelbrecht, A new locally convergent particle swarm optimizer, in: Proceedings of IEEE International Conference on Systems, Man and Cybernetics 3, 2002, pp. 96–101.
[5] R. Boyd, P.J. Richerson, Culture and the Evolutionary Process, University of Chicago Press, Chicago, 1985.
[6] G. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, Computer Vision Graphics and Image Processing 37 (1987) 54–115.
[7] G. Carpenter, S. Grossberg, ART2: self-organization of stable category recognition codes for analog input patterns, Applied Optics 26 (23) (1987) 4919–4930.
[8] G. Carpenter, S. Grossberg, D. Rosen, Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system, Neural Networks 4 (1991) 759–771.
[9] L.Y. Chuang, H.W. Chang, C.J. Tu, C.H. Yang, Improved binary PSO for feature selection using gene expression data, Computational Biology and Chemistry 32 (1) (2008) 29–38.
[10] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: Proceedings of IEEE Congress on Evolutionary Computation (ICEC), Washington, DC, 1999, pp. 1951–1957.

[11] S. Du, W. Li, K. Cao, A learning algorithm of artificial neural network based on GA–PSO, in: Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China, 2006, pp. 3633–3637.
[12] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the 6th International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[13] R. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: IEEE International Conference on Neural Networks, 2001, pp.1942–1948.
[14] E. Forgy, Cluster analysis of multivariate data: efficiency vs. interpretability of classification, Biometrics 21 (1965) 768–780.
[15] H. Frigui, R. Krishnapuram, A robust competitive clustering algorithm with applications in computer vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (5) (1999) 450–465.
[16] I. Gath, A. Geva, Unsupervised optimal fuzzy clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (7) (1989) 773–781.
[17] J.Y. Goulermas, X.J. Zeng, P. Liatsis, J.F. Ralph, Generalized regression neural networks with multiple-bandwidth sharing and hybrid optimization, IEEE Transactions on Systems, Man, and Cybernetics: Part B. Cybernetics 37 (6) (2007) 1434–1445.
[18] G. Hamerly, C. Elkan, Learning the *K* in K-means, in: Proceedings of 7th Annual Conference on Neural Information Processing Systems (NIPS), 2003, pp. 281–288.
[19] J.H. Holland, Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, Ann Arbor, MI, 1975.
[20] K. Huang, A synergistic automatic clustering technique (Syneract) for multispectral image analysis, Photogrammetric Engineering Remote Sensing 1 (1) (2002) 33–40.
[21] S. Huston, Surface mount technology market forecasting, Journal of Industrial Technology 17 (1) (2000) 2–7.
[22] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Transactions on Systems, Man and Cybernetics: Part B 34 (2) (2004) 997–1006.
[23] Y.T. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Applied Soft Computing 8 (2008) 849–857.
[24] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, New York, 1990.
[25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. 4, 1995, pp.1942–1948.
[26] J. Kennedy, R. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceeding of IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, NJ, 1997, pp. 4104–4109.
[27] S. Kiranyaz, T. Ince, A. Yildirim, M. Gabbouj, Fractional particle swarm optimization in multi-dimensional search space, IEEE Transactions on Systems Man and Cybernetics: Part B 40 (2) (2010) 298–319.
[28] T. Kohonen, The self-organizing map, Proceedings of IEEE 78 (9) (1990) 1464–1480.
[29] S. Kotsiantis, P. Pintelas, Recent advances in clustering: a brief survey, WSEAS Trans. Information Science & Applications 1 (1) (2004) 73–81.
[30] R.J. Kuo, H.S. Wang, T.L. Hu, S.H. Chou, Application of ant K-means on clustering analysis, Computers & Mathematics with Applications 50 (2005) 1709–1724.
[31] R.J. Kuo, T.L. Hu, Z.Y. Chen, Application of radial basis function neural network for sales forecasting, in: Proceedings of the 2009 International Asia Conference on Informatics in Control, Automation and Robotics (CAR'09), Bangkok, Thailand, 2009, pp. 325–328.
[32] R.J. Kuo, L.M. Lin, Application of a hybrid of genetic algorithm and particle swarm optimization algorithm for order clustering, Decision Support Systems 49 (2010) 451–462.
[33] C.Y. Lee, E.K. Antonsson, Dynamic partitional clustering using evolution strategies, in: Proceedings of the 3rd Asia-Pacific Conference on Simulated Evolution and Learning, Nagoya, Japan, 2000.
[34] J.S. Lee, S. Olafsson, Data clustering by minimizing disconnectivity, Information Sciences 181 (2011) 732–746.
[35] Y. Leung, J. Zhang, Z. Xu, Clustering by space–space filtering, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12) (2000) 1396–1410.
[36] J. Li, H. Fu, Molecular dynamics-like data clustering approach, Pattern Recognition 44 (2011) 1721–1737.
[37] B. Liu, L. Wang, Y.H. Jin, An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers, Computers & Operations Research 33 (10) (2006) 2960–2971.
[38] A. Lorette, X. Descombes, J. Zerubia, Fully unsupervised fuzzy clustering with entropy criterion, International Conference on Pattern Recognition 3 (2000) 3998–4001.
[39] D. Maravall, J. de Lope, J.A. Martin, Hybridizing evolutionary computation and reinforcement learning for the design of almost universal controllers for autonomous robots, Neurocomputing 72 (4-6) (2009) 887–894.
[40] E. Martinez, M.M. Alvarez, V. Trevino, Compact cancer biomarkers discovery using a swarm intelligence feature selection algorithm, Computational Biology and Chemistry 34 (1) (2010) 244–250.
[41] D.W.V.D. Merwe, A.P. Engelbrecht, Data clustering using particle swarm optimization, in: The 2003 Congress on Evolutionary Computation, 2003, pp. 215–220.
[42] M.A. Montes de Oca, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, Evolutionary Computation, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1120–1132.
[43] L. Nani, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing 72 (4–6) (2009) 1092–1097.
[44] M.G.H. Omran, A. Salman, A.P. Engelbrecht, Dynamic clustering using particle swarm optimization with application in image segmentation, Pattern Analysis & Applications 8 (4) (2006) 332–344.
[45] S. Ouadfel, M. Batouche, A. Taleb-Ahmed, A modified particle swarm optimization algorithm for automatic image clustering, in: Proceedings of the 1st International Symposium on Modeling and Implementing Complex Systems, Constantine, Algeria, 2010.
[46] S. Paterlinia, T. Krink, Differential evolution and particle swarm optimization in partitional clustering, Computational Statistics & Data Analysis 50 (5) (2006) 1220–1247.
[47] D. Pelleg, A. Moore, X-means: extending K-means with efficient estimation of the number of clusters, in: Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000, pp. 727–734.
[48] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 240–255.
[49] J. Robinson, S, Sinton, R.S. Yahya, Particle swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horn antenna, in: IEEE Antennas and Propagation Society International Symposium, San Antonio 1, 2002, pp. 314–317.
[50] J. Sander, M. Ester, H.P. Kriegel, X. Xu, Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications, Data Mining and Knowledge Discovery 2 (2) (1998) 169–194.
[51] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE Press, Piscataway, NJ, 1998, pp. 69–73.
[52] P.N. Suganthan, Particle swarm optimizer with neighborhood operator, in: Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, NJ, 1999, pp 1958–1962.
[53] G. Taguchi, S. Chowdhury, Y. Wu, Taguchi's Quality Engineering Handbook, Wiley, Hoboken, NJ, 2005.
[54] P.N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Pearson Education, Inc., 2006.
[55] R.H. Turi, Clustering-based color image segmentation. PhD Thesis, Monash University, Australia, 2001.
[56] A. Unler, A. Murat, R.B. Chinnam, mr^{2}PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification. Information Sciences (08 June 2010), 2010, doi:10.1016/j.ins.2010.05.037.
[57] C.S. Wallace, D.M. Boulton, An information measure for classification, Computer Journal 11 (2) (1968) 185–194.
[58] L. Wang, J.S. Yu, Fault feature selection based on modified binary PSO with mutation and its application in chemical process fault diagnosis, Advances in Natural Computation, Lecture Notes in Computer Science 3612 (2005) 832–840.

[59] X.F. Xie, W.J. Zhang, Z.L. Yang, A dissipative particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computing (CEC 2002), Honolulu, Hawaii, USA, 2002, pp. 1666–1670.
[60] R. Xu, D. Wunsch, Survey of clustering algorithm, IEEE Transactions on Neural Networks 16 (3) (2005) 645–678.
[61] D. Zhang, X. Liu, Z. Guan, A dynamic clustering algorithm based on PSO and Its application in fuzzy identification, in: Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06), 2006, pp 232–235.
[62] B. Zhao, C.X. Guo, B.R. Bai, Y.J. Cao, An improved particle swarm optimization algorithm for unit commitment, International Journal of Electrical Power & Energy Systems 28 (7) (2006) 482–490.