Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

An efficient hybrid evolutionary optimization algorithm based on PSO and SA for clustering

Taher NIKNAM[†1], Babak AMIRI[2], Javad OLAMAEI[3], Ali AREFI[4]
([1]*Electronic and Electrical Engineering Department, Shiraz University of Technology, Shiraz, Iran*)
([2]*Information Technology Department, Islamic Azad University-Fars Science and Research Branch, Shiraz, Iran*)
([3]*Technical Engineering Department, Islamic Azad University-South Tehran Branch, Tehran, Iran*)
([4]*Power Engineering Group, Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran*)
[†]E-mail: niknam@sutech.ac.ir

Received Mar. 18, 2008; Revision accepted June 25, 2008; Crosschecked Dec. 26, 2008

Abstract: The *K*-means algorithm is one of the most popular techniques in clustering. Nevertheless, the performance of the *K*-means algorithm depends highly on initial cluster centers and converges to local minima. This paper proposes a hybrid evolutionary programming based clustering algorithm, called PSO-SA, by combining particle swarm optimization (PSO) and simulated annealing (SA). The basic idea is to search around the global solution by SA and to increase the information exchange among particles using a mutation operator to escape local optima. Three datasets, Iris, Wisconsin Breast Cancer, and Ripley's Glass, have been considered to show the effectiveness of the proposed clustering algorithm in providing optimal clusters. The simulation results show that the PSO-SA clustering algorithm not only has a better response but also converges more quickly than the *K*-means, PSO, and SA algorithms.

Key words: Simulated annealing (SA), Data clustering, Hybrid evolutionary optimization algorithm, *K*-means clustering, Particle swarm optimization (PSO)
doi:10.1631/jzus.A0820196 **Document code:** A **CLC number:** TP301.6

INTRODUCTION

Data clustering is a manner of grouping data into classes or clusters. In clustering, the data in each cluster assign a high degree of likeness while being very dissimilar to data from other clusters. Differences are evaluated according to the attributed values describing the objects. Generally, distance measures are used. Data clustering has been utilized in many different applications, such as data mining, machine learning, biology, and statistics (Tsai *et al*., 2004; Kao *et al*., 2008). Traditional clustering algorithms can be partitioned into two main categories: hierarchical and partitional clustering. In hierarchical clustering the data are not separated into a particular cluster in a single step. Hierarchical clustering does not need to specify the number of clusters, most of which are deterministic. On the other hand, partitional clustering attempts to directly partition the dataset into a set of disjoint clusters. The partitional clustering begins with a randomly chosen or user-defined clustering, and then optimizes the clustering according to some validity measurements (Fathian *et al*., 2007).

In this paper, we focus on partitional clustering, and in particular a common partitional clustering method called *K*-means clustering. Among clustering algorithms, the *K*-means clustering method is one of the most widely used and applied methods. The main idea of the *K*-means clustering is to specify *K* centroids, one for each cluster. All samples in the dataset are compared with each center by means of the Euclidean distance and assigned to the closest cluster center. The procedure is iterated until no sample is pending. In each stage, the center of each cluster is recalculated by using the average vector of the objects assigned to the cluster. The algorithm stops when the changes in the cluster centers from one stage to the next are close to zero or smaller than a pre-specified value. Every sample is allocated to only one cluster (Mingoti and Lima, 2006).

Unfortunately, the results of the *K*-means are very sensitive to the initial values of centers. A poor selection of centers may lead to a local optimum, which is quite inferior to the global optimum (Laszlo and Mukherjee, 2007). Recently, evolutionary algorithms such as the genetic algorithm (GA), tabu search (TS), and simulated annealing (SA) have been utilized to solve the clustering problem. However, most evolutionary methods such as GAs and TS are typically very slow at finding an optimal solution. Recently researchers have presented new evolutionary methods such as particle swarm algorithms to solve hard optimization problems, which not only have a better response but also converge very quickly in comparison with ordinary evolutionary methods (Eberhart and Shi, 2001). All studies confirm that the particle swarm optimization (PSO) should be taken into account as a powerful technique, which is efficient enough to handle various kinds of nonlinear optimization problems. Nevertheless, it may be trapped into local optima if the global best and local best positions are equal to the position of particle over a number of iterations (Niknam, 2006; Olamaei *et al*., 2008). To overcome this shortcoming this paper presents a novel hybrid evolutionary optimization method based on PSO and SA, called PSO-SA, for optimally clustering *N* objects into *K* clusters, which not only has a better response but also converges more quickly than ordinary evolutionary algorithms. The basic idea is to search around the global solution by SA and to increase the information exchange among particles using a mutation operator to escape local optima.

In the following, the cluster analysis problem is discussed in Section 2. In Sections 3 and 4 the basic principles are introduced of the PSO and SA algorithms, respectively. The implementation of the PSO-SA algorithm in clustering is illustrated in Section 5. In Section 6 the performance of the PSO-SA algorithm is demonstrated and compared with that of the original SA, PSO and *K*-means for different datasets.

CLUSTER ANALYSIS PROBLEM

The *K*-means algorithm, utilized for many clustering tasks (MacQueen, 1967), attempts to find the cluster centers, C_1, C_2, ..., C_K, in such a way that the sum of the squared distances (this sum of squared distances is termed the 'objective function', f) of each data point (X_i) to its nearest cluster center (C_k) is minimized, as shown in Eq.(3), where d is a distance function. Typically, d is chosen as the Euclidean distance derived from the Minkowski metric Eqs.(1) and (2).

$$d(\boldsymbol{x},\boldsymbol{y})=\left(\sum_{i=1}^{m}|x_i-y_i|^r\right)^{1/r}, \tag{1}$$

$$d(\boldsymbol{x},\boldsymbol{y})=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2}, \tag{2}$$

$$f(\boldsymbol{X},\boldsymbol{C})=\sum_{i=1}^{N}\left(\min_{k=1,2,\dots,K} d(X_i,C_k)\right)^2. \tag{3}$$

The steps of the *K*-means algorithm are described as follows:

Step 1: Initialize K center locations C_1, C_2, ..., C_K.

Step 2: Assign each X_i to its nearest cluster center C_k.

Step 3: Update each cluster center C_k as the means of all X_i that have been assigned as closest to it.

Step 4: Calculate the objective function Eq.(3).

Step 5: If the value of f converges, then return (C_1, C_2, ..., C_K); otherwise, go to Step 2.

ORIGINAL PSO ALGORITHM

PSO is an evolutionary optimization technique introduced by Kennedy and Eberhart (1995). Since then it has been widely used to solve a wide range of optimization problems (Angeline, 1998; Shi and Eberhart, 1998; Miranda and Fonseca, 2002; Naka *et al*., 2003; Gaing, 2003; 2004; Hu *et al*., 2004; Esmin *et al*., 2005; Chang *et al*., 2005; Chu *et al*., 2006; Cai *et al*., 2007). The PSO concept is based on social behavior of bird flocking and fish schooling. In the PSO algorithm, the particles move around in the multi-dimensional search space. The positions of individual particles are adjusted according to their previous best positions and the neighborhood best or the global best. PSO is the only evolutionary algorithm that does not implement the survival of the fittest. The reason is that all particles in PSO are kept as members of the population during the course of the searching process. Therefore, we can conclude that the PSO algorithm can be utilized to a wide range of continuous optimization problems (Sung and Jin, 2000; Ng and Wong, 2002).

The searching procedure based on this concept can be described by

$$\begin{cases} V_i^{(t+1)} = \omega V_i^{(t)} + c_1 rand_1(\circ)(Pbest_i - X_i^{(t)}) \\ \qquad + c_2 rand_2(\circ)(Gbest - X_i^{(t)}), \\ X_i^{(t+1)} = X_i^{(t)} + V_i^{(t+1)}, \end{cases} \tag{4}$$

where i=1, 2, …, N_{Swarm} is the index of each particle; t is the iteration number; $rand_1(\circ)$ and $rand_2(\circ)$ are random numbers between 0 and 1; *Pbest* is the best previous solution of the ith particle that is recorded; *Gbest* is the best particle among the whole population; N_{Swarm} is the number of the swarms; constants c_1 and c_2 are weighting factors, which pull each particle towards *Pbest* and *Gbest* positions, respectively. Low values of c_1 and c_2 permit particles to travel far from the goal region before being togged back. On the other hand, high values result in rapid forward or backward movements, from the goal region. Hence, the values of c_1 and c_2 are often set to 2.0 according to early experiences. Angeline (1998) proposed the parameter ω into the PSO equation to improve its performance. The appropriate selection of inertia weight ω in Eq.(4) provides a balance between the global and local positions. As originally developed, ω often decreases linearly from about 0.9 to 0.4 during a run. Generally ω is defined by

$$\omega^{(t+1)} = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} \cdot t, \tag{5}$$

where $t_{\max}$ and t are the maximum number of iterations and the current iteration number, respectively. $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum of the inertia weights, respectively. Fig.1 presents a graphical depiction of the basic idea of the particle swarm optimizer.

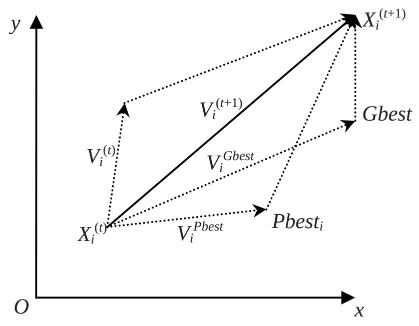

Fig.1 Concept of a search by PSO

To apply the PSO algorithm for clustering, the following steps should be taken (Olamaei *et al.*, 2008):

Step 1: The initial population and initial velocity for each particle should be generated randomly.

Step 2: The objective function is to be evaluated for each individual.

Step 3: The individual that has the minimum objective function should be selected as the global position.

Step 4: The ith individual is selected.

Step 5: The best local position (*Pbest*) is selected for the ith individual.

Step 6: The modified velocity for the ith individual needs to be calculated based on the local and global positions and Eq.(4).

Step 7: The modified position for the ith individual should be calculated based on Eq.(4) and then checked with its limit.

Step 8: If all individuals are selected, go to the next step; otherwise, i=i+1, and go to Step 4.

Step 9: If the current iteration number reaches the predetermined maximum iteration number, the search procedure is stopped; otherwise, go to Step 2.

The last *Gbest* is the solution to the problem.

SIMULATED ANNEALING

SA is an optimization technique that has been successfully used for solving a wide range of optimization problems. This optimization technique acts on the basis of a condensed material behavior at low temperatures, which in fact simulates the annealing process in nature like freezing and crystallizing liquid or cooling and annealing metal. This procedure consists of slow cooling and allowing material to redistribute atoms as they lose mobility. The procedure will continue till a minimum energy state is reached (Meer, 2007; Saber *et al*., 2007; Christober Asir Rajan and Mohan, 2007; Niknam *et al*., 2008a; 2008b; Wu *et al*., 2008).

In SA process, at first material is placed in a heat bath. Next step is to melt and disarrange the material physically by raising the temperature gradually. Material particles will orientate in the direction that is equal to its difference from a high energy level. In a further step, heat bath will be cooled slowly by decreasing the temperature gradually. This will allow the particles to align themselves in a regular crystalline lattice structure. Final structure reached is similar to whatever at first was. Notice that gradual decreasing of temperature is important to reach equilibrium after each temperature drop for a stable low energy structure; otherwise, the structure will be irregular, and this will defect solidifying mechanism and final structure will be highly unstable.

The best point of the SA technique is that, it can be used to a variety of optimization problems without the need to change the basic structure of the computation and also the specialist knowledge.

In the SA algorithm, like most of its family (meta heuristics that are based on gradual local improvement), the algorithm starts with a randomly chosen none optimal configuration using a suitable mechanism and calculates its relative cost to improve the configuration. By reduction of differential cost ΔX_k, a new configuration will be selected and the process is repeated until a termination criterion is met.

The negative point of meta heuristic algorithms is that they can be easily trapped in local minima. The SA technique solves this problem by allowing ‘uphill’ moves based on a model of the annealing process in the physical world (as mentioned before). During the cooling process the system can escape local minima by moving to a thermal equilibrium of a higher energy potential based on the probabilistic distribution w of entropy S, i.e.,

$$S = k \ln w, \tag{6}$$

where k is Boltzmann’s constant and w is the probability that the system exists in a state that relates to all the possible states that it could be in. Thus, given entropy’s relation to energy E and temperature T, the reduction rate would be

$$\mathrm{d}S = \frac{\mathrm{d}E}{T}, \tag{7}$$

which is actually the probabilistic expression of w related to the energy distribution of the temperature T as follows:

$$w \propto \exp(-E/(kT)). \tag{8}$$

Eq.(8) is known as a Boltzmann probability distribution and is used to select the uphill moves that may help the optimization procedure escape from local minima.

The general procedure for the SA algorithm can be summarized as follows:

Step 1: Select an initial line configuration C and an initial temperature T.

Step 2: Find another solution, namely C_{next}, by modifying the last answer C.

Step 3: Calculate the energy differential $\Delta E = f(C_{\text{next}}) - f(C)$.

Step 4: If $\Delta E<0$, go to Step 9.

Step 5: Generate a random number, namely R, between 0 and 1.0.

Step 6: If $R<\exp(-\Delta E/T)$, go to Step 9.

Step 7: Repeat Steps 2~6 for a number of predetermined steps for the given temperature.

Step 8: If no new configuration C_{next} is accepted, then go to Step 10.

Step 9: Decrease the temperature T, replace C with C_{next}, and go to Step 2.

Step 10: Reheat the environment by setting T to a higher value.

Step 11: Repeat Steps 1~10 until no further improvement is obtained.

APPLICATION OF PSO-SA TO CLUSTERING

In this study, a new hybrid evolutionary algorithm is proposed to incorporate the SA algorithm into the original PSO. The algorithm is called PSO-SA and is applied to clustering. In the proposed algorithm, we use SA as a local search around *Gbest*; in other words, *Gbest* changes in each iteration. The control variables are the cluster centers. To implement the PSO-SA algorithm in the clustering problem, the following steps should be taken and repeated.

Step 1: Generate the initial population and initial velocity.

The initial population $\boldsymbol{P}$ and initial velocity $\boldsymbol{V}$ for each particle are randomly generated as follows:

$$\boldsymbol{P} = [\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_{N_{\text{Swarm}}}]^{\text{T}}, \tag{9}$$

$$\boldsymbol{V} = [\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_{N_{\text{Swarm}}}]^{\text{T}}, \tag{10}$$

where

$$\begin{aligned}
\boldsymbol{C}_i &= [\boldsymbol{C}_1', \boldsymbol{C}_2', \ldots, \boldsymbol{C}_K'],\ i = 1, 2, \ldots, N_{\text{Swarm}},\\
\boldsymbol{C}_j' &= [c_1, c_2, \ldots, c_d],\quad c_i^{\min} < c_i < c_i^{\max},\\
\boldsymbol{V}_i &= [\boldsymbol{V}_{\text{c}1}, \boldsymbol{V}_{\text{c}2}, \ldots, \boldsymbol{V}_{\text{c}K}],\ i = 1, 2, \ldots, N_{\text{Swarm}},\\
\boldsymbol{V}_{\text{c}j} &= [v_1, v_2, \ldots, v_d],\quad v_i^{\min} < v_i < v_i^{\max},
\end{aligned}$$

$\boldsymbol{C}_j'$ is the *j*th cluster center for the *i*th individual; $\boldsymbol{V}_{\text{c}j}$ is the velocity of the *j*th cluster center for the *i*th individual; $\boldsymbol{V}_i$ and $\boldsymbol{C}_i$ are velocity and position of the *i*th individual, respectively; *d* is the dimension of each cluster center; $v_i^{\max}$ and $v_i^{\min}$ are the maximum and minimum values of velocity of each point belonging to the *j*th cluster center, respectively; $c_i^{\max}$ and $c_i^{\min}$ (each center feature) are the maximum and minimum values of each point belonging to the *j*th cluster center, respectively.

Step 2: Calculate the objective function value for each individual.

Step 3: Sort the initial population based on the objective function values.

The initial population is ascending, based on the value of the objective function.

Step 4: Select the best global position.

The individual that has the minimum objective function is selected as the best global position (*Gbest*).

Step 5: Apply SA to search around the global solution *Gbest*. Swap *Gbest*, if the solution obtained by SA is better than it.

Step 6: Select the best local position for each individual.

Step 7: Select the *i*th individual and move it according to the following rules:

$$\begin{cases}
\boldsymbol{V}_i^{(t+1)} = \omega \boldsymbol{V}_i^{(t)} + c_1 rand_1(\circ)(Pbest_i - \boldsymbol{C}_i^{(t)}) \\
\qquad + c_2 rand_2(\circ)(Gbest - \boldsymbol{C}_i^{(t)}), \\
\boldsymbol{C}_i^{(t+1)} = \boldsymbol{C}_i^{(t)} + \boldsymbol{V}_i^{(t+1)}.
\end{cases} \tag{11}$$

The modified position for the *i*th individual is checked with its limit.

Step 8: If all individuals are selected, go to the next step; otherwise, *i*=*i*+1, and go back to Step 7.

Step 9: Check the termination criteria.

If the current iteration number reaches the predetermined maximum iteration number, stop the search procedure; otherwise, replace the initial population with the new population of swarms and then go back to Step 3.

The last *Gbest* is the solution to the problem. The flowchart of PSO-SA is shown in Fig.2.

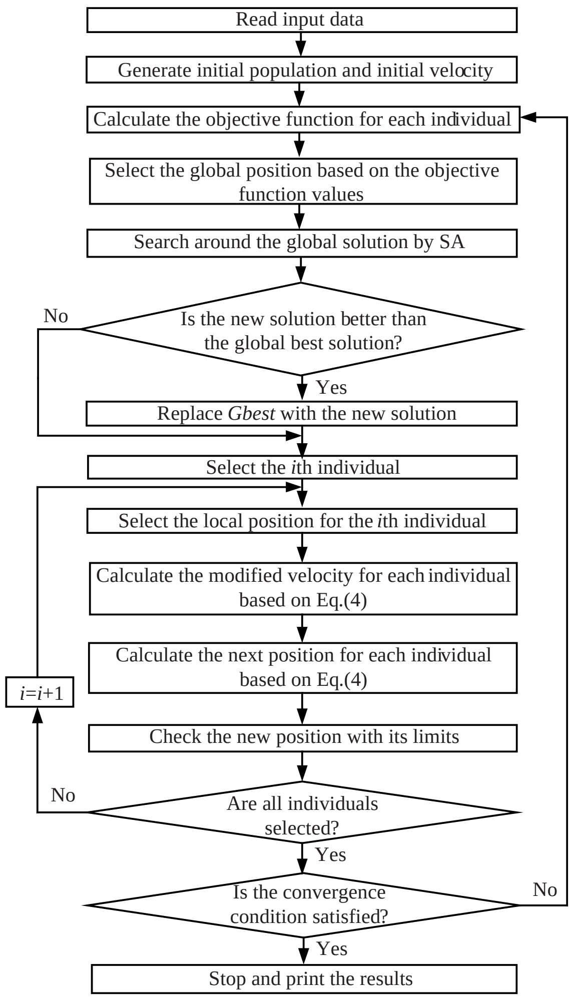
Read input data
Generate initial population and initial velocity
Calculate the objective function for each individual
Select the global position based on the objective function values
Search around the global solution by SA
Is the new solution better than the global best solution?
No
Yes
Replace Gbest with the new solution
Select the ith individual
Select the local position for the ith individual
Calculate the modified velocity for each individual based on Eq.(4)
Calculate the next position for each individual based on Eq.(4)
i=i+1
Check the new position with its limits
Are all individuals selected?
No
Yes
Is the convergence condition satisfied?
No
Yes
Stop and print the results

Fig.2 Flowchart of PSO-SA

EXPERIMENTAL RESULTS

We assess and compare the performance of the POS-SA algorithm with the original PSO, SA and *K*-means algorithms on three real-life datasets, i.e., Iris, Wisconsin Breast Cancer, and Ripley's Glass.

Iris data (N=150, d=4, K=3) include 150 random samples of flowers from the Iris species setosa, versicolor, and virginica collected by Anderson (1935). There exist 50 observations for sepal length, sepal width, petal length, and petal width in cm from each species. The Iris data were utilized by Fisher (1936) in his initiation of the linear-discriminant-function technique.

Wisconsin Breast Cancer (N=683, d=9, K=2) consists of 683 objects characterized by 9 features: clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. In the Wisconsin Breast Cancer there exist two categories: malignant (444 objects) and benign (239 objects).

Ripley's Glass (N=214, d=9, K=6) includes 6 different types of glass: building windows float processed (70 objects), building windows non-float processed (76 objects), vehicle windows float processed (17 objects), containers (13 objects), tableware (9 objects), and headlamps (29 objects). Each glass type was analyzed for 9 features, which are refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron.

The parameters required for implementation of the PSO-SA algorithm are T, c_1, c_2, $\omega_{\min}$, and $\omega_{\max}$. In this study the proper values for the aforementioned parameters are T=100, c_1=c_2=2, N_{Swarm}=10~15, $\omega_{\min}$=0.4, and $\omega_{\max}$=0.9, determined by 10 runs of the algorithm.

The algorithms were implemented using MATLAB 7.1 on a computer with a Pentium IV, 2.8 GHz CPU and 512 GB RAM. Tables 1~5 present a comparison among the results of PSO-SA, SA, PSO, and *K*-means for 10 random tails using the mentioned three datasets.

For Iris data it is revealed that the PSO-SA clustering algorithm obtains a best value of 96.65 while the best solutions of PSO, SA, and *K*-means are 96.73, 97.05, and 97.24, respectively. The PSO-SA has obtained the best solution in all runs, while the PSO, SA, and *K*-means algorithms have obtained the global solution in one run, one run and two runs respectively. As shown in Table 1 the best and worst solutions for the PSO-SA are the same. The worst solutions of the PSO, SA, and *K*-means algorithms are 102.23, 101.58, and 104.79, respectively.

Table 1 Simulation results for 10 random trials using the Iris data

Initial configuration	Number of iterations			
	K-means	PSO	SA	PSO-SA
1	101.01	97.06	98.75	96.65
2	104.79	99.08	97.05	96.65
3	97.81	102.23	98.98	96.65
4	104.44	96.73	100.72	96.65
5	101.35	97.61	100.93	96.65
6	97.24	98.05	101.58	96.65
7	101.26	98.02	101.19	96.65
8	100.92	100.80	98.71	96.65
9	97.24	98.52	100.03	96.65
10	97.38	98.02	100.60	96.65

For Wisconsin Breast Cancer data (Table 2), the PSO-SA algorithm achieves the best value of 2964.38 in 90% of total runs. The *K*-means algorithm obtains the best value of 2987.19 only once. The optimum values obtained by PSO and SA algorithms are 2973.50 and 2980.49, respectively, which are obtained in 20% and 10% of all runs respectively. The worst solutions of PSO-SA, PSO, SA and *K*-means algorithms are 2964.50, 3318.88, 3242.01 and 2988.24, respectively. The simulation results show that the worst value obtained by the proposed algorithm is less than those obtained by the other algorithms.

Table 2 Simulation results for 10 random trials using the Wisconsin Breast Cancer data

Initial configuration	Number of iterations			
	K-means	PSO	SA	PSO-SA
1	2988.24	2981.99	3080.18	2964.38
2	2987.19	2973.50	2980.49	2964.38
3	2988.13	3094.16	3242.01	2964.38
4	2988.00	3147.64	2991.81	2964.50
5	2987.58	3000.27	3146.59	2964.38
6	2987.81	3034.15	3129.57	2964.38
7	2988.24	2973.50	3057.32	2964.38
8	2987.83	3318.88	2987.44	2964.38
9	2987.32	2989.68	2989.33	2964.38
10	2987.46	2986.60	3155.81	2964.38

The simulation results for the clustering problem using the Ripley's Glass data are shown in Table 3. For this problem, the PSO-SA, PSO, SA and *K*-means methods find the optimum solutions of 199.57, 270.57, 274.45 and 215.47, respectively. The PSO-SA reaches the optimum solution nine times while the PSO, SA and *K*-means algorithms converge to the best solution two, one and one time(s), respectively. The PSO-SA algorithm attains the worst solution value of 200.01, which is significantly less than those of the others.

Table 3 Simulation results for 10 random trials using the Ripley's glass data

Initial configuration	Number of iterations			
	K-means	PSO	SA	PSO-SA
1	227.61	275.05	280.39	199.57
2	242.77	281.11	279.27	199.57
3	237.91	279.48	277.16	199.57
4	218.11	270.57	286.08	199.57
5	246.71	283.52	278.08	199.57
6	255.38	272.71	278.50	199.57
7	240.58	277.58	282.89	199.57
8	239.53	270.57	274.45	200.01
9	215.47	274.79	282.00	199.57
10	230.70	271.72	284.70	199.57

In terms of the number of function evaluations (Table 5), *K*-means requires the least number of function evaluations in all the datasets. However, the results of *K*-means are not very acceptable. For the Iris dataset, the number of function evaluations of PSO-SA, PSO, SA and *K*-means are 2523, 4963, 4941 and 120, respectively. The results reveal that the number of function evaluations of PSO-SA is less than those of other evolutionary algorithms. Based on the obtained simulation results, we can conclude that the changes in the number of fitness function evaluations of the proposed algorithm are less than those of the other evolutionary algorithms in all cases (Table 4).

Table 4 Number of function evaluations for the three datasets

Dataset	Number of function evaluations			
	K-means	PSO	SA	PSO-SA
Iris	120	4963	4941	2523
Wisconsin Breast Cancer	180	16290	17471	3082
Ripley's Glass	630	197973	198294	15928

Table 5 illustrates the CPU time of the proposed algorithm in comparison with others. As shown in Table 5, the PSO-SA algorithm converges quicker than other algorithms.

Table 5 CPU time of the four algorithms

Dataset	CPU time (s)			
	K-means	PSO	SA	PSO-SA
Iris	~5	~36	~95	~31
Wisconsin Breast Cancer	~8	~43	~112	~32
Ripley's Glass	~9	~68	~129	~34

The simulation results demonstrate that the proposed hybrid evolutionary algorithm converges to the global optimum almost every time, and with fewer function evaluations and leads naturally to the conclusion that the PSO-SA algorithm is a viable and robust technique for data clustering.

CONCLUSION

We proposed a hybrid evolutionary algorithm based clustering algorithm, called PSO-SA. The PSO-SA algorithm was used to search for the cluster centers. This algorithm minimizes the objective function of the clustering problem. When the number of clusters is known a priori, the PSO-SA algorithm can find the cluster centers. In order to demonstrate the effectiveness of the PSO-SA clustering algorithm in finding optimal clusters, three datasets—Iris, Wisconsin Breast Cancer, and Ripley's Glass—have been considered. The datasets consist of the numbers of dimensions ranging from 4 to 9 and the numbers of clusters ranging from 2 to 6. The simulation results of the proposed algorithm were compared with those of the *K*-means, original PSO, and SA algorithms. The results reveal that the PSO-SA clustering algorithm provides a performance that is significantly better than that of the *K*-means algorithm.

References

Anderson, E., 1935. The irises of the Gaspé peninsula. *Bull. Am. Iris Soc.*, **59**:2-5.

Angeline, P.J., 1998. Using Selection to Improve Particle Swarm Optimization. IEEE Int. Conf. on Computational Intelligence, p.84-89.

Cai, X., Cui, Z., Zeng, J., Tan, Y., 2007. Performancedependent adaptive particle swarm optimization. *Int. J. Innov. Comput. Inf. Control*, **3**(6):1697-1706.

Chang, J.F., Chu, S.C., Roddick, J.F., Pan, S.J., 2005. A parallel particle swarm optimization algorithm with communication strategies. *J. Inf. Sci. Eng.*, **21**(4):809-818.

Christober Asir Rajan, C., Mohan, M.R., 2007. An evolutionary programming based simulated annealing method for solving the unit commitment problem. *Int. J. Electr. Power Energy Syst.*, **29**(7):540-550. [doi:10.1016/j.ijepes.2006.12.001]

Chu, S.C., Tsai, P., Pan, J.S., 2006. Parallel Particle Swarm Optimization Algorithms with Adaptive Simulated Annealing. Studies in Computational Intelligence Book Series. Springer Berlin Heidelberg, **31**:261-279.

Eberhart, R., Shi, Y., 2001. Particle Swarm Optimization: Development, Application and Resources. IEEE Congress on Evolutionary Computation, **1**:81-86.

Esmin, A.A..A., Lambert-Torres, G., Zambroni de Souza, A.C., 2005. A hybrid particle swarm optimization applied to loss power minimization. *IEEE Trans. Power Syst.*, **20**(2):859-866. [doi:10.1109/TPWRS.2005.846049]

Fathian, M., Amiri, B., Maroosi, A., 2007. Application of honey-bee mating optimization algorithm on clustering. *Appl. Math. Comput.*, **190**(2):1502-1513. [doi:10.1016/j.amc.2007.02.029]

Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. *Ann. Eug.*, **7**:179-188.

Gaing, Z.L., 2003. Particle swarm optimization to solving the economic dispatch considering the generator constraints. *IEEE Trans. Power Syst.*, **18**(3):1187-1195. [doi:10.1109/TPWRS.2003.814889]

Gaing, Z.L., 2004. A particle swarm optimization approach for optimum design of PID controller in AVR system. *IEEE Trans. Power Syst.*, **19**(2):384-391.

Hu, X., Shi, Y., Eberhart, R., 2004. Recent Advances in Particle Swarm. IEEE Congress on Evolutionary Computation, **1**:90-97.

Kao, Y.T., Zahara, E., Kao, I.W., 2008. A hybridized approach to data clustering. *Exp. Syst. Appl.*, **34**(3):1754-1762. [doi:10.1016/j.eswa.2007.01.028]

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization. IEEE Int. Conf. on Neural Networks, **4**:1942-1948.

Laszlo, M., Mukherjee, S., 2007. A genetic algorithm that exchanges neighboring centers for *k*-means clustering. *Pattern Recog. Lett.*, **28**(16):2359-2366. [doi:10.1016/j.patrec.2007.08.006]

MacQueen, J.B., 1967. Some Methods for Classification and Analysis of Multivariate Observations. Proc. 5th Berkeley Symp. on Mathematical Statistics and Probability, **1**:281-297.

Meer, K., 2007. Simulated annealing versus metropolis for a TSP instance. *Inf. Processing Lett.*, **104**(6):216-219. [doi:10.1016/j.ipl.2007.06.016]

Mingoti, S.A., Lima, J.O., 2006. Comparing SOM neural network with fuzzy *c*-means, *k*-means and traditional hierarchical clustering algorithms. *Eur. J. Oper. Res.*, **174**(3):1742-1759. [doi:10.1016/j.ejor.2005.03.039]

Miranda, V., Fonseca, N., 2002. EPSO—Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems. IEEE/PES Transmission and Distribution Conf. and Exhibition: Asia Pacific, **2**:745-750. [doi:10.1109/TDC.2002.1177567]

Naka, S., Genji, T., Yura, T., Fukuyama, Y., 2003. A hybrid particle swarm optimization for distribution state estimation. *IEEE Trans. Power Syst.*, **18**(1):60-68. [doi:10.1109/TPWRS.2002.807051]

Ng, M.K., Wang, J.C., 2002. Clustering categorical data sets using tabu search techniques. *Pattern Recog.*, **35**(12):2783-2790. [doi:10.1016/S0031-3203(02)00021-3]

Niknam, T., 2006. An Approach Based on Particle Swarm Optimization for Optimal Operation of Distribution Network Considering Distributed Generators. IEEE 32nd Annual Conf. on Industrial Electronics, p.633-637. [doi:10.1109/IECON.2006.347222]

Niknam, T., Bahmani Firouzi, B., Nayeripour, M., 2008a. An efficient hybrid evolutionary algorithm for cluster analysis. *World Appl. Sci. J.*, **4**(2):300-307.

Niknam, T., Olamaie, J., Amiri, B., 2008b. A hybrid evolutionary algorithm based on ACO and SA for cluster analysis. *J. Appl. Sci.*, **8**(15):2695-2702. [doi:10.3923/jas.2008.2695.2702]

Olamaei, J., Niknam, T., Gharehpetian, G., 2008. Application of particle swarm optimization for distribution feeder reconfiguration considering distributed generators. *Appl. Math. Comput.*, **201**(1-2):575-586. [doi:10.1016/j.amc.2007.12.053]

Saber, A.Y., Senjyu, T., Yona, A., Urasaki, N., Funabashi, T., 2007. Fuzzy unit commitment solution—a novel twofold simulated annealing approach. *Electr. Power Syst. Res.*, **77**(12):1699-1712. [doi:10.1016/j.epsr.2006.12.002]

Shi, Y., Eberhart, R., 1998. A Modified Particle Swarm Optimizer. Proc. IEEE World Congress on Computational Intelligence, p.69-73. [doi:10.1109/ICEC.1998.699146]

Sung, C.S., Jin, H.W., 2000. A tabu-search-based heuristic for clustering. *Pattern Recog.*, **33**(5):849-858. [doi:10.1016/S0031-3203(99)00090-4]

Tsai, C.F., Tsai, C.W., Wu, H.C., Yang, T., 2004. ACODF: a novel data clustering approach for data mining in large databases. *J. Syst. Software*, **73**(1):133-145. [doi:10.1016/S0164-1212(03)00216-4]

Wu, T.H., Chang, C.C., Chung, S.H., 2008. A simulated annealing algorithm for manufacturing cell formation problems. *Exp. Syst. Appl.*, **34**(3):1609-1617. [doi:10.1016/j.eswa.2007.01.012]