Applied Soft Computing 10 (2010) 183–197

Contents lists available at ScienceDirect

Applied Soft Computing

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

An efficient hybrid approach based on PSO, ACO and *k*-means for cluster analysis

Taher Niknam [a,*], Babak Amiri [b]

[a] *Electronic and Electrical Department, Shiraz University of Technology, Modars Blvd. Shiraz, Iran*
[b] *Iran University of Science and Technology, Tehran, Iran*

ARTICLE INFO

Article history:
Received 12 October 2008
Received in revised form 8 June 2009
Accepted 2 July 2009
Available online 10 July 2009

Keywords:
Ant colony optimization (ACO)
Data clustering
Hybrid evolutionary optimization algorithm
k-means clustering
Fuzzy adaptive particle swarm optimization (FAPSO)

ABSTRACT

Clustering is a popular data analysis and data mining technique. A popular technique for clustering is based on *k*-means such that the data is partitioned into K clusters. However, the *k*-means algorithm highly depends on the initial state and converges to local optimum solution. This paper presents a new hybrid evolutionary algorithm to solve nonlinear partitional clustering problem. The proposed hybrid evolutionary algorithm is the combination of FAPSO (fuzzy adaptive particle swarm optimization), ACO (ant colony optimization) and *k*-means algorithms, called FAPSO-ACO–K, which can find better cluster partition. The performance of the proposed algorithm is evaluated through several benchmark data sets. The simulation results show that the performance of the proposed algorithm is better than other algorithms such as PSO, ACO, simulated annealing (SA), combination of PSO and SA (PSO–SA), combination of ACO and SA (ACO–SA), combination of PSO and ACO (PSO–ACO), genetic algorithm (GA), Tabu search (TS), honey bee mating optimization (HBMO) and *k*-means for partitional clustering problem.

1. Introduction

Data clustering describes the process of grouping data into classes or clusters such that the data in each cluster share a high degree of similarity while being very dissimilar to data from other clusters. Dissimilarities are assessed according to the attribute values describing the objects. Generally, distance measures are utilized. Data clustering has roots in a number of areas; including data mining, machine learning, biology, and statistics. Traditional clustering algorithms can be divided into two main categories: hierarchical and partitional [1–3]. This paper concentrates on the partitional clustering. *k*-Means clustering algorithm, which is developed three decades ago, is one of the most popular partitional clustering used in variety of domains. The *k*-means algorithm is defined over continuous data. The *k*-means algorithm gave better results only when the initial partitions were close to the final solution. In other words, the results of *k*-means highly depend on the initial state and reach to local optimal solution. In order to overcome this problem, a lot of studies have done in clustering [1–13]. For instance, Kao et al. have proposed a hybrid technique based on combining the *k*-means algorithm, Nelder–Mead simplex search, and PSO for cluster analysis [1]. Cao et al. have presented a hybrid algorithm according to the combination of GA, *k*-means and logarithmic regression expectation maximization [2]. Zalik has introduced a *k*-means algorithm that performs correct clustering without pre-assigning the exact number of clusters [3]. Krishna and Murty have presented an approach called genetic *k*-means algorithm for clustering analysis [4]. Mualik and Bandyopadhyay have proposed a genetic algorithm based method to solve the clustering problem and experiment on synthetic and real life datasets to evaluate the performance [5]. It defines a basic mutation operator specific to clustering called distance-based mutation. Fathian et al. have proposed the HBMO algorithm to solve the clustering problem [6]. A genetic algorithm that exchanges neighboring centers for *k*-means clustering has presented by Laszlo and Mukherjee [7]. Shelokar et al. have introduced an evolutionary algorithm based on ACO algorithm for clustering problem. Ng and Sung have proposed an approach based on TS for cluster analysis [7,8]. Niknam et al. have presented a hybrid evolutionary optimization algorithm based on the combination of ACO and SA to solve the clustering problem [11,12]. Niknam et al. have presented a hybrid evolutionary algorithm based on PSO and SA to find optimal cluster centers [13].

The PSO algorithm is one of the modern evolutionary algorithms. This algorithm was first proposed by Kennedy and Eberhart. PSO was developed through simulation of a simplified social system, and has been found to be robust in solving continuous nonlinear optimization problems. The PSO algorithm can produce high-quality solutions within shorter calculation time and more stable convergence characteristics than other stochastic methods [12–17]. However, the performance of the traditional PSO

* Corresponding author.
E-mail addresses: niknam@sutech.ac.ir, taher_nik@yahoo.com (T. Niknam), amiri_babak@ind.iust.ac.ir (B. Amiri).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.07.001

significantly depends on its parameters, and it often suffers from the problem of being trapped in local optima. Also the final outputs have some stochastic characteristics. In order to avoid these problems, this paper presents a new hybrid evolutionary optimization algorithm based on combining the fuzzy adaptive particle swarm optimization (FAPSO) and ACO algorithms, called FAPSO–ACO. In the algorithm, the inertia weight and learning factors of PSO are dynamically adjusted using fuzzy IF/THEN rules. The algorithm incorporates intelligent decision-making structure of ACO algorithm into the original FAPSO where the global best position is unique for every particle. The proposed algorithm uses randomly selection procedure of ACO algorithm to assign different global best positions to every distinct agent.

In this paper, in order to overcome the *k*-means shortcomings, the hybrid evolutionary algorithm is used to solve the clustering problem. To use the advantages of the *k*-means algorithm in the proposed algorithm, the output of hybrid FAPSO–ACO algorithm is considered as the initial state of *k*-means. Through experiments, it is shown that the FAPSO–ACO–K algorithm efficiently finds accurate clusters in several datasets.

The main contribution of this paper is presentation of a new hybrid evolutionary algorithm based on the combination of FAPSO and ACO algorithm to solve the clustering problem.

The rest of this paper is organized as follows. In Section 2, the cluster analysis problem is discussed. In Sections 3 and 4, the basic principles of the PSO and ACO algorithms are introduced, respectively. In Section 5, the proposed hybrid evolutionary algorithm is presented. The application of the FAPSO–ACO–K algorithm in clustering is shown in Section 6. In Section 7, the feasibility of the FAPSO–ACO–K is demonstrated and compared with the PSO–ACO, PSO, ACO, PSO–SA, ACO–SA, HBMO, SA, GA, TS and *k*-means for different data sets. In Section 8 a case of an internet bookstore has been analyzed. Finally, Section 9 includes the conclusion.

2. Cluster analysis problem

Data clustering, which is an NP-complete problem of finding groups in heterogeneous data by minimizing some measure of dissimilarity, is one of the fundamental tools in data mining, machine learning and pattern classification solutions [10]. Clustering in *N*-dimensional Euclidean space R^N is the process of partitioning a given set of *n* points into a number, say *k*, of groups (or, clusters) based on some similarity (distance) metric in clustering procedure is Euclidean distance, which derived from the Minkowski metric (Eqs. (1) and (2))

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

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

Let the set of *n* points $\{X_1, X_2, \ldots, X_n\}$ be represented by the set *S* and the *K* clusters be represented by $C_1, C_2, \ldots, C_K$. Then:

$$C_i \neq \phi \quad \text{for} \quad i = 1, \ldots, K,$$
$$C_i \cap C_j = \phi \quad \text{for} \quad i = 1, \ldots, K, \;\; j = 1, \ldots, K, \;\; \text{and} \, i \neq j$$
$$\text{and} \bigcup_{i=1}^{K} C_i = S.$$

In this study, we will also use Euclidian metric as a distance metric. The existing clustering algorithms can be simply classified into the following two categories: hierarchical clustering and partitional clustering. The most class of popular class of partitional clustering methods is the center based clustering algorithms [11]. The *k*-means algorithms, is one of the most widely used center based clustering algorithms. To find *K* centers, the problem is defined as an optimization (minimization) of a performance function, *Perf*(*X*, *C*), defined on both the data items and the center locations. A popular performance function for measuring goodness of the k clustering is the total within-cluster variance or the total mean-square quantization error (MSE), Eq. (3) [11]

$$Perf(X,C) = \sum_{i=1}^{N} Min\{||X_i - C_l||^2 | l = 1, \ldots, K\} \tag{3}$$

The steps of the *k*-means algorithm are as follows [4]:

Step 1: Choose *K* cluster centers $C_1, C_2, \ldots, C_k$ randomly from *n* points $\{X_1, X_2, \ldots, X_n\}$.
Step 2: Assign point X_i, $i = 1, 2, \ldots, n$ to cluster C_j, $j \in \{1, 2, \ldots, K\}$ if $||X_i - C_j|| < ||X_i - C_p||$, $p = 1, 2, \ldots, K$, and $j \neq p$.
Step 3: Compute new cluster centers $C_1^*, C_2^*, \ldots, C_K^*$ as follows:

$$C_i^* = \frac{1}{n} \sum_{x_j \in C_i} X_j, \quad i = 1, 2, \ldots, K,$$

where n_i is the number of elements belonging to cluster C_i.
Step 4: If termination criteria satisfied, stop otherwise continues from step 2.

Note that in case the process close not terminates at step 4 normally, then it executed for a mutation fixed number of iterations.

3. Original PSO and FAPSO algorithms

3.1. Original PSO

PSO is a population-based stochastic search algorithm. It was first introduced by Kennedy and Eberhart. Since then, it has been widely used to solve a broad range of optimization problems [13–17]. The algorithm was presented as simulating animals' social activities, e.g. insects, birds, etc. It attempts to mimic the natural process of group communication to share individual knowledge when such swarms flock, migrate, or hunt. If one member sees a desirable path to go, the rest of this swarm will follow quickly. In PSO, this behavior of animals is imitated by particles with certain positions and velocities in a searching space, wherein the population is called a swarm, and each member of the swarm is called a particle. Starting with a randomly initialized population, each particle in PSO flies through the searching space and remembers the best position it has seen. Members of a swarm communicate good positions to each other and dynamically adjust their own position and velocity based on these good positions. The velocity adjustment is based upon the historical behaviors of the particles themselves as well as their neighbors. In this way, the particles tend to fly towards better and better searching areas over the searching process. The searching procedure based on this concept can be described by (4)

$$V_i^{(t+1)} = \omega \cdot V_i^{(t)} + c_1 \cdot rand_1(\cdot) \cdot (Pbest_i - X_i^{(t)}) + c_2 \cdot rand_2(\cdot) \cdot (Gbest - X_i^{(t)})$$
$$X_i^{(t+1)} = X_i^{(t)} + V_i^{(t+1)}$$
$$X_i^t = [x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,K}^t]_{1\times K}$$
$$Pbest_i = [pbest_{i,1}^t, pbest_{i,2}^t, \ldots, pbest_{i,K}^t]_{1\times K}$$
$$Gbest = [gbest_1^t, gbest_2^t, \ldots, gbest_K^t]_{1\times K}$$
$$\omega^{(t+1)} = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} \times t$$

In these equations, $i = 1, 2, \ldots, N_{\text{Swarm}}$ is the index of each particle, *t* is the iteration number, $rand_1(\cdot)$ and $rand_2(\cdot)$ are random numbers between 0 and 1. $Pbest_i$ is the best previous experience of the *i*th particle that is recorded. *Gbest* is the best particle among the entire population. N_{Swarm} is the number of the swarms. Constants c_1 and c_2 are the weighting factors of the stochastic acceleration terms, which pull each particle towards $Pbest_i$ and $Gbest$ positions. $t_{\max}$ *is* the maximum number of iterations. $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum of the inertia weights, respectively. K is the number of variables.

As indicated in (4), there are three tuning parameters; ω, c_1, and c_2 that each of them has a great impact on the algorithm performance. The inertia weight ω controls the exploration properties of the algorithm. The learning factors c_1 and c_2 determine the impact of the personal best $Pbest_i$ and the global best $Gbest$, respectively. If $c_1 > c_2$, the particle has the tendency to converge to the best position found by itself $Pbest_i$ rather than the best position found by the population $Gbest$, and vice versa. Most implementations use a setting with $c_1 = c_2 = 2$ [12–17].

3.2. FAPSO

From experience, it is known that [15]:

(i) when the best fitness is found at the end of the run, low inertia weight and high learning factors are often preferred;
(ii) when the best fitness is stayed at one value for a long time, the number of generations for unchanged best fitness is large. The inertia weight should be increased and learning factors should be decreased.

According to this knowledge, a fuzzy system is utilized to tune the inertia weight and learning factors with the best fitness (*BF*) and the number of generations for the best unchanged fitness (*NU*) as the input variables, and the inertia weight (ω) and learning factors (c_1 and c_2) as the output variables.

The *BF* value determines the performance of the best candidate solution found so far. The optimization problems have different ranges of the *BF* values. To use a FAPSO, which is applicable to a various range of problems, the ranges of the *BF* and *NU* values are normalized into [0, 1.0]. The *BF* values can be normalized using the following formula:

$$NBF = \frac{BF - BF_{\min}}{BF_{\max} - BF_{\min}} \tag{5}$$

where $BF_{\max}$ and $BF_{\min}$ are the maximum and minimum values of *BF* value.

NU values are normalized in a similar way. Other converting methods are possible as well. The bound values for ω, c_1, and c_2 are: $0.2 \le \omega \le 1.2$, $1 \le c_1 \le 2$ and $1 \le c_2 \le 2$.

For fuzzification of every input and output, the membership functions shown in Fig. 1 are used.

In Fig. 1 PS (positive small), PM (positive medium), PB (positive big) and PR (positive bigger) are the linguistic values for the inputs and outputs.

The Mamdani-type fuzzy rule is used to formulate the conditional statements that comprise fuzzy logic. For example

$$\begin{aligned} R_i:\ & \text{IF (NBF is PB) and (NU is PM)}, \\ & \text{THEN } (\omega \text{ is PB}),\ (c_1 \text{ is PM}) \text{ and } (c_2 \text{ is PM}) \end{aligned}$$

The fuzzy rules in Tables 1–3 [15] are used to adjust the inertia weight (ω) and learning factors (c_1 and c_2), respectively. Each rule represents a mapping from the input space to the output space.

To obtain a deterministic control action, a defuzzification strategy is required. In this paper, the centroid method has been used.

4. ACO algorithm

Dorigo and his colleague's first proposed ACO as a multi-agent approach to solve difficult combinatorial optimization problems like the traveling salesman problem (TSP) and the quadratic assignment problem (QAP) [11,12,18]. A number of studies based on ACO have been presented that deal with the classification task of data mining [18–21,8]. Shelokar et al. [8] presented an ant colony optimization, methodology for optimally clustering N objects into K clusters. The algorithm employs distributed agents who mimic the way real ants find a shortest path from their nest to food source and back. Mullen et al. reviewed the ant colony

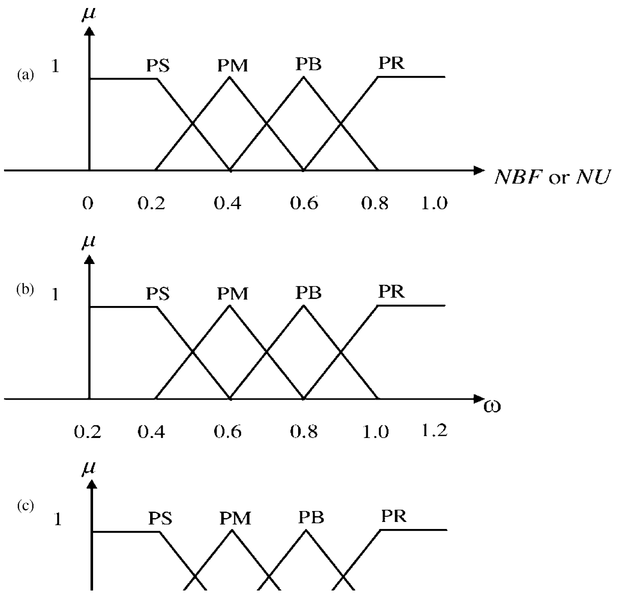

Fig. 1. Membership functions of inputs and outputs (a) *NBF* or NU, (b) ω, and (c) c_1 and c_2.

Table 1
Fuzzy rules for the inertia weight.

ω		NU			
		PS	PM	PB	PR
NBF	PS	PS	PM	PB	PB
	PM	PM	PM	PB	PR
	PB	PB	PB	PB	PR
	PR	PB	PB	PR	PR

Table 2
Fuzzy rules for learning factor c_1.

c_1		NU			
		PS	PM	PB	PR
NBF	PS	PR	PB	PB	PB
	PM	PB	PM	PM	PS
	PB	PB	PM	PS	PS
	PR	PM	PM	PS	PS

Table 3
Fuzzy rules for learning factor c_2.

c_2		NU			
		PS	PM	PB	PR
NBF	PS	PR	PB	PM	PM
	PM	PB	PM	PS	PS
	PB	PM	PM	PS	PS
	PR	PM	PS	PS	PS

optimization in several problems including clustering problem [18]. Holden and Freitas used the ACO algorithm in web page classification [19]. Parpinelli et al. applied the ACO algorithm in data mining problem [21].

Ants are insects which live together. Since they are blind animals, they find the shortest path from nest to food with the aid of pheromone. The pheromone is the chemical material deposited by ants, which serves as critical communication media among ants, thereby guiding the determination of the next movement. On the other hand, ants find the shortest path based on intensity of pheromone deposited on different paths. Generally, intensity of pheromone and the length of the path are used to simulate ant system. In ACO algorithm, the probability with which an ant q chooses to go from city i to city j is

$$p_{ij}^{q}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\gamma_2}[1/L_{il}]^{\gamma_1}}{\sum_{l\in N_i^q}[\tau_{il}(t)]^{\gamma_2}[1/L_{il}]^{\gamma_1}} & \text{if } j\in N_i^q\\ 0 & \text{otherwise}\end{cases}\tag{6}$$

where τ_{ij} and L_{ij} are the intensity of pheromone and the length of the path between cities j and i, respectively. γ_1 and γ_2 are the control parameters for determining the weight of the trail intensity and the length of the path, respectively. N_i^q is the set of neighbors of city i for the qth ant. After selecting the next path, the trail intensity of pheromone is updated as

$$\begin{aligned}&\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t)\\&\Delta\tau_{ij}(t)=\begin{cases}\dfrac{1}{Lm} & \text{if } (i,j)\in \text{global}-\text{best}-\text{tour}\\ 0 & \text{otherwise}\end{cases}\end{aligned}\tag{7}$$

In the above equation, $0<\rho\leq 1$ is the pheromone trial evaporation rate. $\Delta\tau_{ij}$ is the amount of pheromone trail added toτ_{ij} by ants. Q is a constant parameter. Lm is the length of the global best tour.

To improve the performance of ACO algorithm, Q-learning can be used in conjunction with ACO as follows.

4.1. ACO and Q-learning

Q-learning falls within the category of reinforcement learning, which is a subset of machine learning to which one could also relate the concept of ant algorithms to. Reinforcement learning involves agents learning by trial and error which actions are best to take in their current environment in order to achieve their goals. In a training phase, each time an agent performs an action in its environment; it may receive a reward or penalty reflecting the desirability of the outcome of the action performed. The goal of the agent is then to choose sequences of actions that maximize the cumulative reward. More specifically, Q-learning involves learning an action-value function, which measures the utility of taking a given action in a given state within the environment. At each time-step, t, an agent in state s_i takes an action which takes it to a new state s_{i+1}. The agent then receives a reward r depending on the new state. The Q-values for each state-action pair are updated at each time-step until convergence between successive Q-values approaches zero, using the following equation:

$$Q_n(s_t,a)\leftarrow(1-\alpha_n)Q_{n-1}(s_t,a)+\alpha_n[r_t+\gamma\max_{a'}Q_{n-1}(s_{t+1},a')]\ \text{and}\ \alpha_n=\frac{1}{1+s_n(s_t,a)}\tag{8}$$

where γ is the discount factor, a' is the action that maximizes Q, and $visits(s_t,a)$ is the total number of times the given state-action pair have previously been visited. An algorithm inspired by the original AS, called Ant-Q, was developed by Dorigo and Gambardella [22,23]. This algorithm has many similarities with the Q-learning algorithm, but also a few key differences; mainly that Ant-Q, unlike typical Q-learning algorithms, involves using multiple agents. These agents communicate, exchanging information in the form of AQ-values. As with AS, the Ant-Q algorithm was developed originally for the classic benchmark problem TSP. For TSP, $AQ(r,s)$ is the Ant-Q value associated with the path (r,s) between cities. $HE(r,s)$ is a heuristic value associated to path (r,s), which for TSP is the inverse of distance. k is an agent whose task it is to complete a closed tour of all cities, and associated with each agent k there is a list, $J_k(r)$, of all cities still to be visited, where r is the current city. This list acts as a kind of memory, and is another important difference between Ant-Q and Q-learning. The state transition rule for an agent k in city r is as follows:

$$s=\begin{cases}\operatorname{argmax}_{u\in J_k(r)}\{[AQ(r,u)]^{\delta}\cdot[HE(r,u)]^{\beta}\} & \text{if } q\leq q_0,\\ S & \text{otherwise},\end{cases}\tag{9}$$

where α and β are parameters which weigh the relative importance of the learned AQ-values and the heuristic values, q is a uniform probability randomly chosen value in $[0,1]$, q_0 $(0\leq q_0\leq 1)$ is a parameter such that the higher q_0 the smaller the probability to make a random choice, and S is a random variable selected according to a probability distribution given by the function of the $AQ(r,u)$'s and $HE(r,u)$'s, with $u\in J_r(r)$. The update rule for the AQ-values is as follows:

$$\begin{aligned}AQ(r,s)\leftarrow&(1-\alpha)\cdot AQ(r,s)+\alpha(\Delta AQ(r,s)+\gamma\\&\cdot Max_{z\in J_k(s)}AQ(s,z)),\end{aligned}\tag{10}$$

where α and γ are the learning step and discount factor, respectively. This update rule is the same as in Q-learning except that the set of available actions in state s, i.e. the set $J_k(s)$, is a function of the previous history of agent k. This approach adapts the idea of ant algorithms to that of Q-learning, and it is important to note that Ant-Q does not make use of artificial pheromones. A different approach has been developed in Refs. [23,24], which adapts the idea of Q-learning to that of ant algorithms, by introducing the use of artificial pheromones into multi-agent Q-learning. The pheromone Q-learning (Phe-Q) algorithm uses the same Q-value update function as in Eq. (13), but with an additional factor to be maximized, called the belief factor. The belief factor is a function of the synthetic pheromone concentration on the trial and reflects the extent to which an agent will take into account the information laid down by other agents from the same cooperating group. The belief factor is the ratio between the sum of actual pheromone concentrations in the current plus surrounding states, and the sum of the maximum possible pheromone concentration in the current plus surrounding states, and is given by

$$B(s,a)=\frac{\sum_{s\in N_a}\phi(s)}{\sum_{s\in N_a}\phi_{\max}(\sigma)}\tag{11}$$

where $\phi(s)$ is the pheromone concentration at state s in the environment, and N_a is the set of surrounding states for a chosen action a. With the addition of the belief factor the Q-learning update function then becomes

$$\begin{aligned}Q_n(s_t,a)\leftarrow&(1-\alpha_n)Q_{n-1}(s_t,a)+\alpha_n\{r_t+\gamma\max_{a'}[Q_{n-1}(s_{t+1},a')\\&+\xi B(s_{t+1},a')]\}\end{aligned}\tag{12}$$

where the parameter ξ is a sigmoid function of time $epochs\geq 0$, such that it increases with the number of agents who successfully complete the given task. In this example, where a key feature of the ant algorithms has been coupled with another established machine learning technique, improvements in the performance compared to the same algorithm without the additional ant algorithm feature have been shown [24,25]. This is a clear example of how 'hybrid' algorithms, bringing elements of different machine learning techniques together, can produce superior performing algorithms.

5. Hybrid FAPSO–ACO–K algorithm

As mentioned in the previous sections, the studies conducted by researchers confirm that the PSO method 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 particle's position over a number of iterations. Recently, numerous ideas have been used to alleviate this drawback by combining other global optimization algorithms such as GA, evolutionary programming (EP) or SA with the PSO [11–17]. In these approaches, new generation members are produced at each iteration by using evolutionary algorithm and then PSO's movement rule is applied to these new members providing better opportunity of exploring new places.

In the PSO algorithm, the *Gbest* value stored in PSO's memory has an important role in the steer of other particles. If the *Gbest* value does not change after some iteration, other particles gradually get close to the *Gbest* position. The ability of the best agent to search local area is also reduced since the condition $\omega < 1$ implies that the velocity of the *Gbest* particle tends to zero by iteration. This condition may lead to local convergence point. In this paper, the proposed algorithm is specifically developed to address a drawback of the original PSO, where the *Gbest* particle is not able to search locally as well as other particles do. The basic idea behind this algorithm is that the selection of the *Gbest* particle for each individual is according to the ACO best path selection methodology. On the other hand, in this paper a new method is proposed to incorporate intelligent decision-making structure of ACO algorithm into the original PSO where the global best position is unique for every particle. However, the proposed algorithm uses randomly selection procedure of ACO algorithm to assign different global best positions to every distinct agent. For clustering problem, the *k*-means algorithm tends to converge faster than the PSO and ACO algorithms as it requires fewer function evolutions, but it usually results in less accurate clustering. The proposed algorithm uses the advantages of this algorithm to improve the final results of simulation. In other word, the results of the PSO–ACO algorithm are used as the initial condition of the *k*-means algorithm. The pseudo code and the flowchart of the hybrid algorithm, called PSO–ACO–K, are shown in Figs. 2 and 3, respectively.

```
Begin
    Generate an initial population randomly
    Generate initial trail intensity between each individual of the initial population
    Calculate the objective function for the initial population
    Sort the initial population based on their objective function values
     do{
        do {
            Select the best global position
            Select the $i^{th}$ individual
            Select the best local position ($Pbest_i$) for the $i^{th}$ individual
            Determine the neighbors ($S_i$) for the $i^{th}$ individual
             if there are not any neighbors
                 Consider the best global position as Gbest
                 Calculate the velocity for the $i^{th}$ individual based on the $Pbest_i$ and Gbest values and the FAPSO parameters
                 Update the position of the $i^{th}$ individual
                Update the trail intensity between the $i^{th}$ and the best global position
             else
                Calculate the transition probability between the $i^{th}$ individual and each individual in $S_i$
                Calculate the cumulative probability
                Select the best global position by using roulette wheel
                Consider the best global position as Gbest
                Calculate the velocity for the $i^{th}$ individual based on the $Pbest_i$ and Gbest values and the FAPSO parameters
                Update the position of the $i^{th}$ individual
               Update the trail intensity between the $i^{th}$ and $j^{th}$ individuals
            endif
            }
       while (all individuals selected)
        Calculate the objective function value for the new population
        Sort the initial population based on their objective function values
        }
      while ( the termination criteria satisfied)
           Run the k-means algorithm while Gbest is considered as its initial point
           If the result of k-means algorithm is better than Gbest
                Consider the result of k-means as the final results
           else
                Consider the Gbest as the final results
           Endif
     End
```

Fig. 2. Pseudocode for the hybrid algorithm.

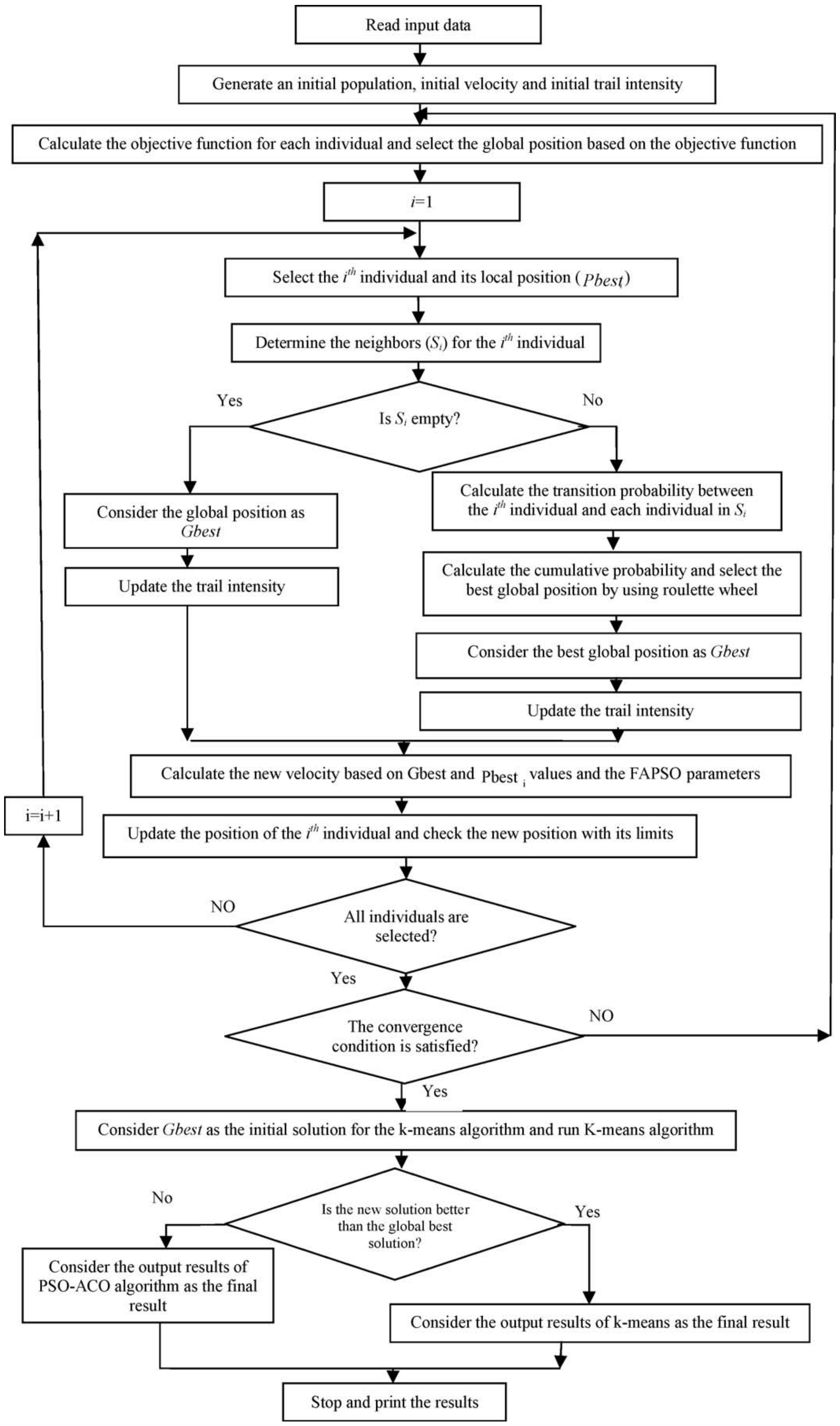
Read input data
Generate an initial population, initial velocity and initial trail intensity
Calculate the objective function for each individual and select the global position based on the objective function
i=1
Select the i^th individual and its local position (Pbest_i)
Determine the neighbors (S_i) for the i^th individual
Is S_i empty?
Yes
No
Consider the global position as Gbest
Update the trail intensity
Calculate the transition probability between the i^th individual and each individual in S_i
Calculate the cumulative probability and select the best global position by using roulette wheel
Consider the best global position as Gbest
Update the trail intensity
Calculate the new velocity based on Gbest and Pbest_i values and the FAPSO parameters
i=i+1
Update the position of the i^th individual and check the new position with its limits
All individuals are selected?
NO
Yes
The convergence condition is satisfied?
NO
Yes
Consider Gbest as the initial solution for the k-means algorithm and run K-means algorithm
Is the new solution better than the global best solution?
No
Yes
Consider the output results of PSO-ACO algorithm as the final result
Consider the output results of k-means as the final result
Stop and print the results

Fig. 3. Flowchart of PSO–ACO.

6. Application of FAPSO–ACO–K on clustering

In this section, the application of FAPSO–ACO–K on the clustering problem is presented. To apply the FAPSO–ACO–K algorithm to solve the clustering problem, the following steps should be taken and repeated.

Step 1: Generate the initial population and initial velocity

The initial population and initial velocity for each particle are randomly generated as follows:

$$\text{Population} = \begin{bmatrix} C_1 \\ C_2 \\ \cdots \\ C_{N_{\text{Swarm}}} \end{bmatrix}$$
$$C_i = [Center_1, Center_2, \ldots, Center_K], \quad i = 1,2,3,\ldots,N_{\text{Swarm}}$$
$$Center_j = [c_1, c_2, \ldots, c_d]$$
$$c_i^{\min} < c_i < c_i^{\max} \tag{13}$$

$$\text{Velocity} = \begin{bmatrix} V_1 \\ V_2 \\ \cdots \\ V_{N_{\text{Swarm}}} \end{bmatrix}$$
$$V_i = [Center\_V_1, Center\_V_2, \ldots Center\_V_K], \quad i = 1,2,3,\ldots,N_{\text{Swarm}}$$
$$Center\_V_j = [v_1, v_2, \ldots, v_d]$$
$$v_i^{\min} < v_i < v_i^{\max} \tag{14}$$

where $Center_j$ is the jth cluster center for the ith individual. $Center\_V_j$ is the velocity of the jth cluster center for the ith individual. V_i and C_i are the velocity and position of the ith individual, respectively. d is the dimension of each cluster center. $v_i^{\max}$ and $v_i^{\min}$ are the maximum and minimum value of the velocity of each point belonging to the jth cluster center, respectively. $c_i^{\max}$ and $c_i^{\min}$ (each feature of center) are the maximum and minimum value of each point belonging to the jth cluster center, respectively.

Step 2: Generate the initial trail intensity

At initialization phase, it is assumed that the trail intensity between each pair of swarms is the same and is generated as follows:

$$Trail\_Intensity = [\tau_{ij}]_{N_{\text{Swarm}} \times N_{\text{Swarm}}} \qquad \tau_{ij} = \tau_0 \tag{15}$$

where τ_{ij} and τ_0 are trial intensity between the ith and jth swarms and initial trial intensity, respectively.

Step 3: Calculate objective function value

The objective function is evaluated for each individual.

Step 4: Sort the initial population based on the objective function values

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

Step 5: Select the best global position

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

Step 6: Select the best local position

The best local position ($Pbest_i$) is selected for each individual.

Step 7: Select the ith individual

The ith individual is selected and neighbors of this particle should be defined dynamically as below:

$$S_i = \left\{ C_j \| \|C_i - C_j\| \le 2D_0 \left(\frac{1}{1 - \exp(-at/t_{\max})} \right), \quad i \ne j \right\} \tag{16}$$

where D_0 is the initial neighborhood radius, a is a parameter used to tune the neighborhood radius over the iteration, t, and $\|\cdot\cdots\cdot\|$ is the Euclidean distance operator.

Step 8: Calculate the next position for the ith individual

There are two cases to calculate the next position as follows:

- *Case (A)* if $S_i \ne \{\}$, where {} stands for null set.

In this case, at first, the transition probabilities between the C_i and each individual in S_i are calculated as indicated in (17):

$$[\text{Probability}]_i = [P_{i1}, P_{i2}, \ldots, P_{i,M}]_{1\times M}$$
$$P_{ij} = \frac{(\tau_{ij})^{\gamma_2}(1/L_{ij})^{\gamma_1}}{\sum_{j=1}^{M} (\tau_{ij})^{\gamma_2}(1/L_{ij})^{\gamma_1}} \tag{17}$$
$$L_{ij} = \frac{1}{|J(C_i) - J(C_j)|}$$

where P_{ij} is the state transition probability between C_i and the jth individual in S_i. M is the number of members in S_i.

Then the cumulative probabilities are calculated as below:

$$[\text{Cumulative probability}]_i = [Cp_1, Cp_2, \ldots, Cp_M]_{1\times M}$$
where
$$Cp_1 = P_{i1}$$
$$Cp_2 = Cp_1 + P_{i2}$$
$$\cdots$$
$$Cp_j = Cp_{j-1} + P_{ij}$$
$$\cdots$$
$$Cp_M = Cp_{M-1} + P_{iM} \tag{18}$$

In above equations, Cp_j is the cumulative probability for the jth individual in S_i. The roulette wheel is used for stochastic selection of the best global position as follows.

A number between 0 and 1 is randomly generated and compared with the calculated cumulative probabilities. The first term of the cumulative probabilities (Cp_j), which is greater than the generated number, is selected and the associated position is considered as the best global position.

The ith particle is then moved according to the following rules, if X_j is selected as the best:

$$V_i^{(t+1)} = \omega \cdot V_i^{(t)} + c_1 \cdot rand_1(\cdot) \cdot (Pbest_i - C_i^{(t)}) + c_2 \cdot rand_2(\cdot) \cdot (C_j - C_i^{(t)})$$
$$C_i^{(t+1)} = C_i^{(t)} + V_i^{(t+1)} \tag{19}$$

The presumed pheromone level between C_i and C_j is updated at the next stage:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + P_{ij} \tag{20}$$

- *Case* (*B*) if S_i = {}, which means there is not any individual in particle's neighborhood.

In this case, the ith particle is moved according to the following rules:

$$V_i^{(t+1)} = \omega \cdot V_i^{(t)} + c_1 \cdot rand_1(\cdot) \cdot (Pbest_i - C_i^{(t)}) + c_2 \cdot rand_2(\cdot) \cdot (Gbest - C_i^{(t)})$$
$$C_i^{(t+1)} = C_i^{(t)} + V_i^{(t+1)} \quad (21)$$

Then, the trail intensity is updated as following, where index j represents the best particle index in the group.

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + r; \quad 0.1 \le r \le 0.5 \quad (22)$$

The modified position for the ith individual is checked with its limit.

In Eqs. (19) and (21), the fuzzy rules are used to evaluate the values of ω, c_1 and c_2 parameters.

Step 9: If all of the individuals are selected, go to the next step, otherwise $i = i + 1$ and go back to step 7.

Step 10: Check the termination criteria

If the current iteration number reaches the predetermined maximum iteration number, go to the next step, otherwise the initial population is replaced with the new population of swarms and then go back to step 3.

Step 11: Consider the last Gbest value as the initial solution for the k-means algorithm

In this step to use the k-means clustering algorithm advantageous, the *Gbest* is considered as an initial solution of the k-means clustering problem. If the results of k-means algorithm are better the *Gbest* value, the k-means results are considered as the final results, otherwise the last *Gbest* is considered as the final results.

7. Experimental results

The experimental results comparing the FAPSO–ACO–K clustering algorithm with several typical stochastic algorithms including the PSO–ACO, PSO, ACO, SA, GA, TS, HBMO, PSO–SA, ACO–SA and k-means algorithms are provided for four artificial data sets and six real-life data sets (*Iris*, *Wine*, *Vowel*, *Contraceptive Method Choice* (*CMC*), *Wisconsin breast cancer and Ripley's glass*), which are described as follows:

Artificial data set one ($n = 600$, $d = 2$, $k = 4$). This is a two-featured problem with four unique classes. A total of 600 patterns were drawn from four independent bivariate normal distributions, where classes were distributed according to

$$N_2\left(\mu = \begin{pmatrix} m_i \\ 0 \end{pmatrix}, \sum = \left[\begin{bmatrix} 0.5 & 0.05 \\ 0.05 & 0.5 \end{bmatrix}\right]\right), \quad (23)$$
$$i = 1, 2, 3, 4 \quad m_1 = -3, \quad m_2 = 0, \quad m_3 = 3, \quad m_4 = 6,$$

μ and Σ being mean vector and covariance matrix, respectively [1]. The data set is illustrated in Fig. 4.

Artificial data set two ($n = 250$, $d = 3$, $k = 5$). This is a three-featured problem with five classes, where every feature of the classes was distributed according to Class 1—Uniform(85, 100), Class 2—Uniform(70, 85), Class 3—Uniform(55, 70) Class 4—Uniform(40, 55), Class 5—Uniform(25, 40) [1]. The data set is illustrated in Fig. 5.

ArtSet*3*: This is an overlapping two-dimensional triangular distribution of data points having nine classes where all the classes are assumed to have equal a priori probabilities (=1/19). It has 900 data points. The X–Y ranges for the nine classes are as follows:

Class 1: $[-3.3, -0.7] \times [0.7, 3.3]$,
Class 2: $[-1.3, 1.3] \times [0.7, 3.3]$,
Class 3: $[0.7, 3.3] \times [0.7, 3.3]$,
Class 4: $[-3.3, -0.7] \times [-1.3, 1.3]$,
Class 5: $[-1.3, 1.3] \times [-1.3, 1.3]$,
Class 6: $[0.7, 3.3] \times [-1.3, 1.3]$,
Class 7: $[-3.3, -0.7] \times [-3.3, -0.7]$,
Class 8: $[-1.3, 1.3] \times [-3.3, -0.7]$,
Class 9: $[0.7, 3.3] \times [-3.3, -0.7]$.

Thus the domain for the triangular distribution for each class and for each axis is 2.6. Consequently, the height will be 1/1.3 (since 12*2.6*height"1). The value of K is chosen to be 9 for this data set.

ArtSet*4*: This is an overlapping ten-dimensional data set generated using a triangular distribution of the form shown in

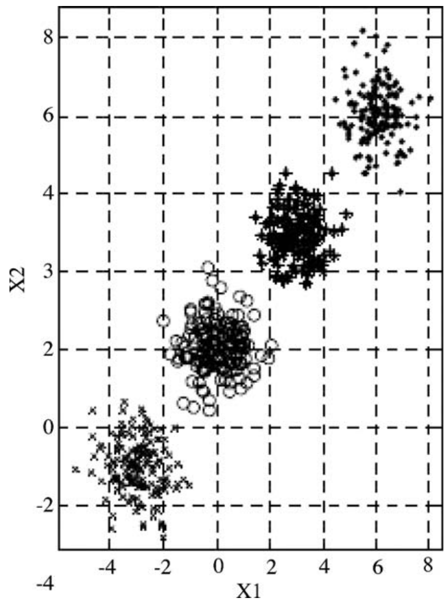

Fig. 4. ArtSet1.

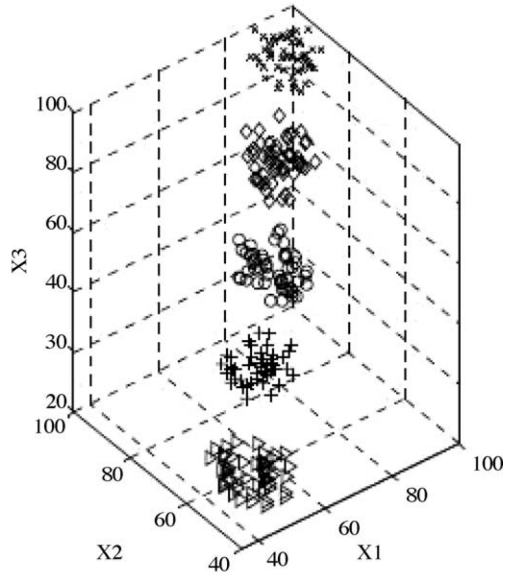

Fig. 5. ArtSet2.

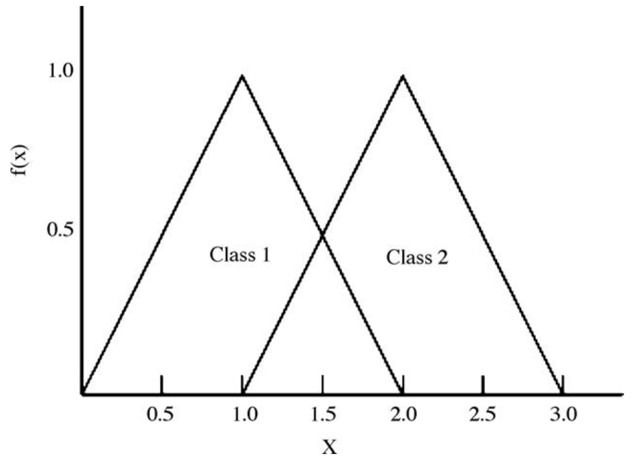

Fig. 6. Triangular distribution along the *X*-axis.

Fig. 6 for two classes, 1 and 2. It has 1000 data points. The value of K is chosen to be 2 for this data set. The range for class 1 is $[0, 2] \times [0, 2] \times [0, 2]\ldots$10 times, and that for class 2 is $[1, 3] \times [0, 2] \times [0, 2]\ldots$9 times, with the corresponding peaks at (1, 1) and (2, 1). The distribution along the first axis (X) for class 1 may be formally quantified as

$$f_1(x) = \begin{cases} 0 & \text{for}\, x \le 0, \\ x & \text{for}\, 0 < x \le 1, \\ 2 - x & \text{for}\, 1 < x \le 2, \\ 0 & \text{for}\, x > 2. \end{cases} \tag{24}$$

for class 1. Similarly for class 2

$$f_1(x) = \begin{matrix} 0 & \text{for}\, x \le 1, \\ x - 1 & \text{for}\, 1 < x \le 2, \\ 3 - x & \text{for}\, 2 < x \le 3, \\ 0 & \text{for}\, x > 3. \end{matrix} \tag{25}$$

The distribution along the other nine axes (Y_i, $i = 1, 2, \ldots, 9$) for both the classes is

$$f_1(x) = \begin{cases} 0 & \text{for}\, y_i \le 0, \\ y_i & \text{for}\, 0 < y_i \le 1, \\ 2 - y_i & \text{for}\, 1 < y_i \le 2, \\ 0 & \text{for}\, y_i > 2. \end{cases} \tag{26}$$

Iris data ($N = 150$, $d = 4$, $K = 3$). This is the iris data set. These data set with 150 random samples of flowers from the iris species setosa, versicolor, and virginica collected by Anderson (1935). From each species there are 50 observations for sepal length, sepal width, petal length, and petal width in cm. This dataset was used by Fisher (1936) in his initiation of the linear-discriminant-function technique [11–13].

Wine data ($N = 178$, $d = 13$, $K = 3$). This is the wine data set, which is also taken from MCI laboratory. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. There are 178 instances with 13 numeric attributes in wine data set. All attributes are continuous. There is no missing attribute value [11–13].

Contraceptive method choice ($N = 1473$, $d = 10$, $K = 3$). This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics [11–13].

Vowel data set ($N = 871$, $d = 3$, $K = 6$). This data set consists of 871 patterns. There are six overlapping vowel classes and three input features [11–13].

Wisconsin breast cancer ($N = 683$, $d = 9$, $K = 2$), which consists of 683 objects characterized by nine features: clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. There are two categories in the data: malignant (444 objects) and benign (239 objects) [11–13].

Ripley's glass ($N = 214$, $d = 9$, $K = 6$), for which data were sampled from six 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 with nine features, which are refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron [11–13].

The original PSO, original ACO, PSO–ACO and FAPSO–ACO algorithms needs to determine the associated parameters such as γ_1, γ_2, ρ, a, r, D_0, c_1, c_2, $\omega_{\min}$ and $\omega_{\max}$. In this paper, the best values for the aforementioned parameters are $\gamma_1 = \gamma_2 = 1.0$, $\rho = .99$, $a = 15$, $r = 0.5$, $D_0 = 10$, $c_1 = c_2 = 2$, $N_{\text{Swarm}} = 10–15$, $\omega_{\min} = 0.4$ and $\omega_{\max} = 0.9$ determined by 10 runs of the algorithm. For example, the mentioned parameters are determined as shown in Tables 4–7 for iris dataset.

The algorithms are implemented by using Matlab 7.1 on a Pentium IV, 2.8 GHz, 512 GB RAM computer.

Tables 8–17 present a comparison among the results of PSO–ACO, ACO [11,12], PSO [1] and [13], SA [6] and [13], PSO–SA [13], ACO–SA [11,12], GA [6], TS [6], HBMO [6] and k-means [11–13] for 100 random tails on the mentioned data sets.

The comparison of results for each dataset based on the bet solution found in 100 distinct runs of each algorithm and the convergence processing time taken to attain the best solution. The quality of the respective clustering will also be compared, where the quality is measured by the following two criteria:

Table 4
Simulation results of PSO algorithm parameters for iris data set.

Case	c_1, c_2	$\omega_{\min}$, $\omega_{\max}$	Best solution	Worst solution	Average solution
1	1, 1	0.2, 1	98.7531	99.8739	99.173
2	1, 1	0.4, 1	98.7531	101.36	99.987
3	1, 1	0.4, 0.9	98.7531	100.823	99.873
4	1.5, 1	0.4, 0.9	97.146	99.783	99.001
5	1.5, 2	0.4, 0.9	97.237	99.237	98.402
6	2, 1.5	0.4, 0.9	97.862	99.739	98.875
7	2, 2	0.2, 1	96.8942	97.8973	97.2328
8	2, 2	0.3, 1	96.8942	98.7563	978.012
9	2, 2	0.4, 1	96.8942	98.7823	97.2583
10	2, 2	0.4, 0.9	96.8942	97.8973	97.2328

Table 5
Simulation results of ACO algorithm parameters for iris data set.

Case	γ_1	γ_2	ρ	Best solution	Worst solution	Average solution
1	0.9	0.9	0.98	97.238	99.9127	98.731
2	0.92	0.9	0.99	97.453	100.871	98.697
3	0.9	0.92	0.98	97.751	99.387	98.182
4	0.94	0.92	0.99	97.521	101.236	99.018
5	0.92	0.94	0.98	97.197	99.997	98.634
6	0.92	0.96	0.99	97.273	100.890	99.320
7	0.96	0.92	0.98	97.236	100.347	99.105
8	0.98	0.94	0.99	97.934	99.719	98.506
9	0.92	0.98	0.98	97.236	98.723	98.003
10	1	1	0.99	97.100777	97.808466	97.171546

Table 6
Simulation results of PSO–ACO algorithm parameters for iris data set.

Case	c_1, c_2	$\omega_{\min}, \omega_{\max}$	γ_1	γ_2	D_0	a	r	ρ	Best solution	Worst solution	Average solution
1	1, 1	0.2, 1	1	1	10	16	0.5	0.99	96.688	96.6986	96.68975
2	1, 1	0.4, 1	1	1	11	15	0.51	0.99	96.67973	96.69817	96.68898
3	1, 1	0.4, 0.9	1	0.9	10	14	0.52	0.99	96.67573	96.67819	96.677236
4	1.5, 1	0.4, 0.9	0.9	1	10	15	0.49	0.99	96.67673	96.69793	96.68773
5	1.5, 2	0.4, 0.9	0.9	0.9	10	15	0.48	0.99	96.675	96.6895	96.67835
6	2, 1.5	0.4, 0.9	1	1	9	15	0.5	0.98	96.665	96.665	96.665
7	2, 2	0.2, 1	1	1	9	15	0.5	0.98	96.662	96.662	96.662
8	2, 2	0.3, 1	1	1	8	15	0.5	0.99	96.660	96.660	96.660
9	2, 2	0.4, 1	1	1	10	15	0.5	0.99	96.661	96.661	96.661
10	2, 2	0.4, 0.9	1	1	10	15	0.5	0.99	96.6500	96.6500	96.6500

Table 7
Simulation results of FAPSO–ACO algorithm parameters for iris data set.

Case	γ_1	γ_2	D_0	a	r	ρ	Best solution	Worst solution	Average solution
1	0.9	0.9	10	16	0.5	0.99	96.6500	96.664	96.6523
2	0.92	0.9	11	15	0.51	0.99	96.6500	96.667	96.6547
3	0.9	0.92	10	14	0.52	0.99	96.6500	96.668	96.6563
4	0.94	0.92	10	15	0.49	0.99	96.6500	96.672	96.6548
5	0.92	0.94	10	15	0.48	0.99	96.6500	96.665	96.6541
6	0.92	0.96	9	15	0.5	0.98	96.6500	96.665	96.6538
7	0.96	0.92	9	15	0.5	0.98	96.6500	96.662	96.65101
8	0.98	0.94	8	15	0.5	0.99	96.6500	96.660	96.6552
9	0.92	0.98	10	15	0.5	0.99	96.6500	96.661	96.6523
10	1	1	10	15	0.5	0.99	96.6500	96.6500	96.6500

Table 8
Results obtained by the algorithms for 100 different runs on Artset1.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	515.878	515.878	515.878	0	~1.5	1923	1.000 (0.000)
PSO–ACO	515.879	515.88	515.890	1E−5	~1.5	1996	1.000 (0.000)
PSO	515.93	627.74	705.598	180.24	~3	3240	1.000 (0.000)
SA	518.9584	684.682	709.985	195.15	~3	3608	1.000 (0.000)
TS	518.9985	659.801	706.845	191.084	~3	2846	1.000 (0.000)
GA	518.0982	638.094	705.86	189.862	~3	2946	1.000 (0.000)
ACO	517.879	519.88	521.890	2.01	~3	1999	1.000 (0.000)
HBMO	515.879	515.88	515.890	2E−5	~3	2053	1.000 (0.000)
PSO–SA	515.879	515.88	515.890	1.1E−5	~3	1999	1.000 (0.000)
ACO–SA	515.879	515.88	515.890	1.15E−5	~3	1998	1.000 (0.000)
k-Means	516.04	721.57	936.450	295.84	~0.2	80	1.000 (0.000)

1. Total mean-square quantization error of a data point to all the centers, as defined in Eq. (3). Clearly, the smaller the sum is, the higher the quality of clustering is.
2. The F-Measure uses the ideas of precision and recall from information retrieval [26,27]. Each class i (as given by the class labels of the used benchmark data set) is regarded as the set of n_i items desired for a query; each cluster j (generated by the algorithm) is regarded as the set of n_j items retrieved for a query; n_{ij} gives the number of elements of class i within cluster j. For each class i and cluster j precision and recall are then defined as $p(i, j) = (n_{ij}/n_j)$ and $r(i, j) = (n_{ij}/n_j)$ and the corresponding value under the F-Measure is $F(i, j) = ((b^2 + 1){\cdot}p(i, j){\cdot}r(i, j)/$

Table 9
Results obtained by the algorithms for 100 different runs on Artset2.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	1743.20	1745.90	1746.89	2.6	~3.5	6890	0.958 (0.023)
PSO–ACO	1743.20	1746.99	1748.943	2.75	~4	7015	0.928 (0.036)
PSO	1743.20	2517.20	2934.084	415.02	~5	11,325	0.8 (0.223)
SA	1743.20	2686.84	2988.098	429.025	~5	11,881	0.754 (0.298)
TS	1743.20	2681.59	3015.481	420.84	~5	10598	0.783 (0.301)
GA	1743.20	2667.30	2985.846	437.05	~5	11243	0.810 (0.348)
ACO	1743.20	1948.97	2075.729	134.068	~5	9985	0.896 (0.197)
HBMO	1743.20	1756.47	1761.864	6.57	~5	11684	0.895 (0.207)
PSO–SA	1743.20	1748.73	1751.946	3.84	~5	7129	0.941 (0.045)
ACO–SA	1743.20	1749.01	1753.927	3.46	~5	7201	0.955 (0.066)
k-Means	1746.9	2762.00	3347.068	720.66	~0.3	150	0.912 (0.102)

Table 10
Results obtained by the algorithms for 100 different runs on Artset3.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	964.083265	964.083265	964.083265	0	~16	3,684	1.000 (0.000)
PSO–ACO	964.326528	965.001684	966.106284	1.08	~17	3,762	1.000 (0.000)
PSO	966.562856	967.684592	969.965824	1.98	~30	9,845	1.000 (0.000)
SA	966.418263	968.614089	970.397392	2.01	~32	9,942	1.000 (0.000)
TS	972.629478	975.209275	979.528463	3.46	~33	9,843	1.000 (0.000)
GA	966.649837	969.772302	972.853946	2.94	~39	11,086	1.000 (0.000)
ACO	964.739472	965.048327	966.283745	1.26	~27	9,346	1.000 (0.000)
HBMO	964.536298	965.029763	966.014309	1.11	~32	9,328	1.000 (0.000)
PSO–SA	964.418263	965.614089	966.797392	1.06	~28	9,427	1.000 (0.000)
ACO–SA	964.268542	965.010634	966.201953	1.01	~28	9,648	1.000 (0.000)
k-Means	968.695841	977.594862	981.0896425	3.64	0.5	185	1.000 (0.000)

Table 11
Results obtained by the algorithms for 100 different runs on Artset4.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	1248.026845	1248.026845	1248.026845	0	~16	3,584	0.979 (0.011)
PSO–ACO	1248.562846	1249.010628	1249.268501	0.17	~17	3,652	0.899 (0.021)
PSO	1248.769582	1249.062985	1249.695824	0.57	~123	16,354	0.878 (0.054)
SA	1249.736287	1249.968105	1250.895375	0.98	~124	17,492	0.871 (0.096)
TS	1282.538294	1285.988483	1299.789237	13.84	~128	19,294	0.810 (0.079)
GA	1258.673362	1263.777767	1271.635528	4.628	~135	20,549	0.890 (0.073)
ACO	1248.958685	1249.034036	1249.335442	0.29	~123	16,495	0.894 (0.043)
HBMO	1248.662849	1249.010628	1249.295784	0.18	~136	20,762	0.888 (0.054)
PSO–SA	1248.663290	1249.124098	1249.809627	0.65	~28	3,845	0.894 (0.064)
ACO–SA	1248.806283	1249.010582	1249.310597	0.19	~40	4,246	0.896 (0.072)
k-Means	1254.9452	1297.6945	1392.9843	85.5	0.5	191	0.872 (0.088)

Table 12
Results obtained by the algorithms for 100 different runs on iris data.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	96.6500	96.6500	96.6500	0	~16	2,480	0.788 (0.004)
PSO–ACO	96.6542	96.6548	96.67412	0.009764	~17	2,523	0.787 (0.006)
PSO	96.8942	97.2328	97.8973	0.347168	~30	4,953	0.782 (0.011)
SA	97.4573	99.957	102.01	2.018	~32	5,314	0.776 (0.025)
TS	97.365977	97.868008	98.569485	0.53	~135	20,201	0.777 (0.023)
GA	113.986503	125.197025	139.778272	14.563	~140	38,128	0.778 (0.008)
ACO	97.100777	97.171546	97.808466	0.367	~75	10,998	0.779 (0.009)
HBMO	96.752047	96.95316	97.757625	0.531	~82	11,214	0.781 (0.022)
PSO–SA	96.66	96.67	96.678	0.008	~17	2,566	0.785 (0.006)
ACO–SA	96.6602	96.73192	96.86381	0.12196	~25	3,629	0.786 (0.005)
k-Means	97.333	106.05	120.45	14.6311	~0.4	120	0.782 (0.000)

Table 13
Results obtained by the algorithms for 100 different runs on wine data.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	16,295.31	16,295.31	16,295.31	0	~30	6,315	0.521 (0.000)
PSO–ACO	16,295.34	16,295.92	16,297.93	0.869661	~33	6432	0.519 (0.002)
PSO	16,345.9670	16,417.4725	16,562.3180	85.4974	~123	16,532	0.518 (0.055)
SA	16,473.4825	17,521.094	18,083.251	753.084	~129	17,264	0.515 (0.039)
TS	16,666.22699	16,785.45928	16,837.53567	52.073	~140	22,716	0.516 (0.075)
GA	16,530.53381	16,530.53381	16,530.53381	0	~170	33,551	0.515 (0.049)
ACO	16,530.53381	16,530.53381	16,530.53381	0	~121	15,473	0.519 (0.054)
HBMO	16,357.28438	16,357.28438	16,357.28438	0	~40	7,238	0.518 (0.029)
PSO–SA	16,295.86	16,296.001	16,296.1034	0.89612	~38	6,987	0.520 (0.000)
ACO–SA	16,298.628	16,310.283	16,322.438	10.62197	~84	11,628	0.520 (0.000)
k-Means	16,555.68	18,061	18,563.12	793.213	0.7	390	0.521 (0.002)

Table 14
Results obtained by the algorithms for 100 different runs on CMC data.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	5,694.2816	5,694.2816	5,694.2816	0	~31	6,850	0.334 (0.002)
PSO–ACO	5,694.5179	5,694.9214	5,697.4254	0.868771	~35	6,923	0.333 (0.002)
PSO	5,700.9853	5,820.9647	5,923.2490	46.959690	~131	21,456	0.331 (0.028)
SA	5,849.0380	5,893.4823	5,966.9470	50.867200	~150	26,829	0.325 (0.067)
TS	5,885.0621	5,993.5942	5,999.8053	40.84568	~155	28,945	0.327 (0.084)
GA	5,705,6301	5,756.5984	5,812.6480	50.3694	~160	29,483	0.324 (0.036)
ACO	5,701.9230	5,819.1347	5,912.4300	45.634700	~127	20,436	0.328 (0.046)
HBMO	5,699.2670	5,713.9800	5,725.3500	12.690000	~122	19,496	0.330 (0.068)
PSO–SA	5,696.059	5,698.6942	5,701.8140	1.8691	~73	10,528	0.333 (0.002)
ACO–SA	5,696.6075	5,698.2618	5,700.2681	1.98238	~89	12,628	0.333 (0.002)
k-Means	5,842.20	5,893.60	5,934.43	47.16	0.5	270	0.334 (0.000)

Table 15
Results obtained by the algorithms for 100 different runs on vowel data.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	148,976.0005	148,976.0010	148,976.0100	0.0001	~16	3,450	0.652 (0.001)
PSO–ACO	148,976.0100	148,995.2032	149,101.6800	24.5420931	~17	3,523	0.651 (0.002)
PSO	148,976.0152	148,999.8251	149,121.1834	28.8134692	~30	9,635	0.648 (0.056)
SA	149,370.4700	161,566.2810	165,986.4200	2,847.08594	~32	9,423	0.645 (0.084)
TS	149,468.268	162,108.5381	165,996.4280	2,846.23516	~33	9,528	0.645 (0.044)
GA	149,513.735	159,153.498	165,991.6520	3,105.5445	~39	10,548	0.647 (0.059)
ACO	149,395.602	159,458.1438	165,939.8260	3,485.3816	~27	8,046	0.649 (0.074)
HBMO	149,201.632	161,431.0431	165,804.671	2,746.0416	~32	8,436	0.650 (0.066)
PSO–SA	149,001.85	150,343.45	154,751.26	351.45	~33	9,328	0.652 (0.001)
ACO–SA	149,005	149,141.4	149,364.3	120.38	~35	9,991	0.652 (0.001)
k-Means	149,422.26	159,242.89	161,236.81	916	0.45	180	0.652 (0.000)

Table 16
Results obtained by the algorithms for 100different runs on Wisconsin breast cancer.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	2,964.25	2,964.25	2,964.25	0	~16	3,492	0.830 (0.002)
PSO–ACO	2,964.38	2,964.39	2,964.50	0.037947	~17	3,545	0.830 (0.008)
PSO	2,973.50	3,050.04	3,318.88	110.8013	~123	16,290	0.819 (0.033)
SA	2,993.45	3,239.17	3,421.95	230.192	~126	17,387	0.818 (0.042)
TS	2,982.84	3,251.37	3,434.16	232.217	~130	18,981	0.818 (0.086)
GA	2,999.32	3,249.46	3,427.43	229.734	~135	20,221	0.819 (0.079)
ACO	2,970.49	3,046.06	3,242.01	90.50028	~123	15,983	0.821 (0.020)
HBMO	2,989.94	3,112.42	3,210.78	103.471	~136	19,982	0.825 (0.011)
PSO–SA	2,965.17	2,966.32	2,967.41	1.7201	~28	3,781	0.829 (0.008)
ACO–SA	2,967.83	2,966.63	2,968.29	1.7732	~40	4,799	0.829 (0.009)
k-Means	2,999.19	3,251.21	3,521.59	251.14	0.5	180	0.829 (0.000)

Table 17
Results obtained by the algorithms for 100 different runs on Ripley's glass.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	199.53	199.53	199.53	0	~31	6,459	0.435 (0.003)
PSO–ACO	199.57	199.61	200.01	0.13914	~35	6,517	0.434 (0.014)
PSO	270.57	275.71	283.52	4.557134	~400	198,765	0.359 (0.067)
SA	275.16	282.19	287.18	4.238458	~410	199,438	0.347 (0.026)
TS	279.87	283.79	286.47	4.192734	~410	199,574	0.351 (0.077)
GA	278.37	282.32	286.77	4.138712	~410	199,892	0.333 (0.049)
ACO	269.72	273.46	280.08	3.584829	~395	196,581	0.364 (0.064)
HBMO	245.73	247.71	249.54	2.438120	~390	195,439	0.401 (0.079)
PSO–SA	200.14	201.45	202.45	0.892430	~38	6,782	0.430 (0.010)
ACO–SA	200.71	201.89	202.76	0.887234	~49	7,894	0.431 (0.017)
k-Means	215.74	235.5	255.38	12.47107	~1	630	0.431 (0.011)

$(b^2 \cdot p(i,j) + r(i,j))$, where we chose $b = 1$ to obtain equal weighting for $p(i,j)$ and $r(i,j)$. The overall F-Measure for the data set of size n is given by $F = \sum_i \frac{n_i}{n} MAX_j\{F(i,j)\}$. Obviously, the bigger F-Measure is, the higher the quality of clustering is.

The simulation results given in Tables 8–17 show that FAPSO–ACO–K is very precise. In other word, it provides the optimum value and small standard deviation in compare to those of obtained by the other methods. For instance, the results obtained on the iris dataset show that FAPSO–ACO–K converges to the global optimum of 96.6500 in all of runs and PSO–ACO reaches to 96.6542 at almost times while the best solutions of PSO, ACO, SA, TS, GA, HBMO, PSO–SA, ACO–SA and K-means are 96.8942, 96.853, 97.4573, 97.365977, 113.986503, 96.752047, 96.66, 96.6602 and 97.333, respectively. The standard deviation of the fitness function for this algorithm, PSO–ACO and PSO–SA are 0, 0.009764 and 0.008, respectively, which they significantly are smaller than other methods. Table 9 shows the results of algorithms on the wine dataset. The optimum value is 16295.31, which is obtained in all the runs of FAPSO–ACO–K algorithm. Noticeably other algorithms fail to attain this value even once within 100 runs. Table 10 provides the results of algorithms on the CMC dataset. As seen from the results, the FAPSO–ACO–K algorithm are far superior those of obtained by the others. For the vowel data set, the best global solution, the worst global solution, the average and the standard deviation of the FAPSO–ACO–K are 148976.0005, 148976.0010, 148976.0100 and 0.001 respectively. For the PSO–ACO algorithm they are 148976.01, 149101.68, 148995.2032 and 24.5420931, respectively. The results of the FAPSO–ACO–K and the PSO–ACO algorithms are much better than those of other algorithms. On Wisconsin breast cancer dataset results given in Table 12, show that the FAPSO–ACO–K provides the optimum value of 2964.25 while the PSO–ACO, PSO, ACO and *k*-means algorithms obtain 2,964.38, 2,973.50, 2,970.49 and 2,987.19, respectively. The FAPSO–ACO–K was able to find the optimum in all of runs. Finally, Table 17 shows the best, average, worst and standard deviation values obtained by algorithms for Ripley's glass dataset. It is found that the FAPSO–ACO–K clustering algorithm is able to provide the same partition of the data points in all the runs.

In terms of the number of function evaluations, *k*-means needs the least number of function evaluations, but the results are less than satisfactory. For the iris dataset, the number of function evaluations of PSO–ACO–K, PSO–ACO, PSO, ACO, SA, TS, GA, HBMO, PSO–SA, ACO–SA and *k*-means are 2468, 2523, 4953, 4931, 5314, 20201, 38128, 11214, 2566, 3629 and 120, respectively. The number of function evaluations of FAPSO–ACO–K for Wine, CMC, Vowel, Wisconsin breast cancer and Ripley's glass are 6315, 6868, 3487, 3492 and 6503, respectively. These results show that the number of function evaluations of FAPSO–ACO–K and PSO–ACO are less than those of other evolutionary algorithms. Based on the obtained simulation results, we can conclude that the changes of the number of fitness function evaluations of the proposed algorithm are less than other evolutionary algorithms for all cases. In the other words, the number of swarms in the FAPSO–ACO–K algorithm does not depend on the number of variables greatly. In the proposed algorithm, N_{Swarm} for iris, wine, CMC, vowel, Wisconsin breast cancer and Ripley's glass is 10, 12, 12, 10, 10 and 14, respectively. Algorithmic parameters for all algorithms are illustrated in Table 18.

The simulation results of the tables illustrate that the average and the standard deviation of F-Measure of proposed algorithm is better than those of obtained by others. This is an indication that the clusters are spatially well separated.

The simulation results in the tables demonstrate that the proposed hybrid evolutionary algorithm converges to global optimum with a smaller standard deviation and less function evaluations and leads naturally to the conclusion that the FAPSO–ACO–K algorithm is a viable and robust technique for data clustering.

8. Market segmentation: a case of an internet bookstore

FAPSO–ACO–K is the best method for clustering analysis as shown in Section 7. For further demonstration of the proposed method, an advanced comparison of five methods made, using real-world data of an internet bookstore in Iran for market

Table 18
Values of parameters of each of five algorithms.

HBMO		ACO		GA		TS	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
# queens	1	# ants	50	Population	50	Tabu list size	25
# drones	150	Probability threshold for maximum trail	0.98	Crossover	0.8	Number of trial solutions	40
Capacity of spermatheca	50	Local search probability	0.01	Mutation rate	0.001	Probability threshold	0.98
Maximum speed	Randomly ∈ [0.5 1]	Evaporation rate	0.01	# iterations	1000	# iterations	1000
Minimum speed	Randomly ∈ [0 1]	# iterations	1000				
Speed reduction	0.98						
Crossover	1.5						
# iterations	1000						

SA		PSO–SA		PSO		ACO–SA	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Probability threshold	0.98	# Swarm	10–15	# Swarm	10 × K × d	# ants	50
Initial temperature	5	Probability threshold	0.98	$c_1 = c_2$	2	Probability threshold for maximum trail	0.98
Temperature multiplier	0.98	Initial temperature	5	$\omega_{\min}$	0.5	Local search probability	0.01
Final temperature	0.01	Temperature multiplier	0.98	$\omega_{\max}$	1	Evaporation rate	0.01
Number of iterations detect steady stat	100	Final temperature	0.01	# iterations	500	Probability threshold	0.98
# iterations	30,000	Number of iterations detect steady stat	100			Initial temperature	5
		$c_1 = c_2$	2			Temperature multiplier	0.98
		$\omega_{\min}$	0.4			Final temperature	0.01
		$\omega_{\max}$	0.9			Number of iterations detect steady stat	50
		# iterations	500			# iterations	500

Table 19
Result obtained by the algorithms for 10 different runs on real-world problem.

Method	Function value			Standard deviation	CPU time (S)	Number of function evaluations	F-Measure
	Best	Average	Worst				
PSO–ACO–K	18177.294528	18177.294528	18177.294528	0	~45	8980	1.000 (0.000)
PSO–ACO	18177.294528	18179.221845	18180.684595	1.001	~37	12546	1.000 (0.000)
PSO	18205.548625	18230.485263	18255.625482	22.84	~150	39556	0.985 (0.023)
SA	18201.289966	18205.281477	18208.681844	2.84	~200	86025	0.978 (0.065)
TS	18449.512506	18637.767038	18789.804808	21.845	~135	37948	0.945 (0.044)
GA	18209.330750	18231.881661	18267.101294	30.622	~175	58503	0.921 (0.058)
ACO	18201.289966	18201.828425	18206.674560	4.99	~190	76878	0.948 (0.036)
HBMO	18178.736324	18180.234000	18193.764244	12.64	~180	69804	0.956 (0.054)
PSO–SA	18177.294528	18179.845682	18180.985672	0.98	~42	13548	1.000 (0.000)
ACO–SA	18177.294528	18179.958463	18180.995843	0.88	~43	13846	1.000 (0.000)
k-Means	18204.658249	18270.658256	18301.694582	39.55	~0.6	190	1.000 (0.000)

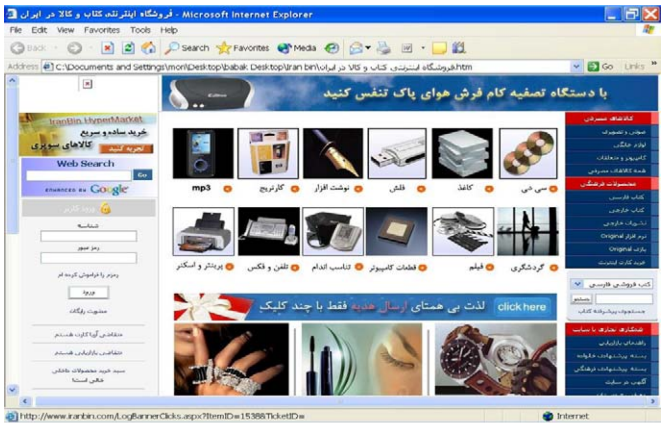
Microsoft Internet Explorer
File Edit View Favorites Tools Help
Web Search
Google
click here

Fig. 7. Iranbin website.

segmentation based on customer loyalty. Iranbin is the biggest internet bookstore in Iran with more than 170,000 Persian and 20,000,000 English books and journals. Iranbin to tailor its products, services and marketing messages to its customers needs to segment them. Customer segments have traditionally been based on market research and demographics. There might be a "young and single" segment or a "loyal entrenched segment". Fig. 7 shows the Iranbin website.

8.1. RFM model

Direct marketing professionals have been trying to gain such insight ever since the end of the nineteenth century, when the first catalogue of products that could be ordered by mail appeared in the USA [28]. However, it was only at the beginning of the 1960s that a simple and effective quantitative method to separate customers who are likely to make purchases from those who are not was devised: the recency, frequency and monetary value (RFV or RFM) analysis [29]. Generally, shortened to RFV, it is sometimes known as "RFM" analysis. In this approach to market segmentation, customers are clustered together into an arbitrary number of segments according to their most recent day of purchase, the number of purchases they have made and the monetary value of their purchases. A random sample taken from the segmented customer database is then, subjected to a direct marketing campaign. As a result, some customer segments may reveal themselves to be profitable, while others may do the reverse. Subsequently, the remaining customers in the database who belong to profitable segments are targeted by the same campaign [30].

8.2. Evaluation of clustering methods

Several methods used to cluster 700 customers of http://www.IranBin.com, an internet bookstore in Iran based on customer loyalty. Based on RFM model, loyalty of each customer determined with three parameters, R (recency of purchase), F (Frequency) and M (Monetary), in this research customers segmented based on these variables. The performance function of F (Eq. (4)), is also calculated since it is used to evaluate the methods. Customers clustered into five clusters. The results have been illustrated in Table 19.

Table 16 shows that FAPSO–ACO–K has the best performance which is identical to the previously experimental result. The FAPSO–ACO–K required the least number of function evaluation (8980) which it is less than other evolutionary algorithms. The FAPSO–ACO–K finds the optimum solution of 18177.294528 in 100% of all its 10 runs.

Cluster 1 is customers that have long relationship with the Iranbin, buy more recently but the money they paid is low. Iranbin should increase the value of sale to these customers by applying appropriate marketing strategies.

Cluster 2 is customers that buy recently but they do not have long relationship with Iranbin. In other word, they are new customers that Iranbin should study more about them and attempt to attract them and sales more to them.

Cluster 3 is customers that have long relationship with Iranbin but recently sales to them decreased. Iranbin should apply appropriate marketing strategy for those customers to retention them.

Cluster 4 is the worth segment. Customers in this segment purchase more, frequency of purchase in these customers is high and they purchased recently. Customers in this segment have long relationship with Iranbin and they are loyal customers.

Cluster 5 is customers purchased recently but value of purchase is not high. They are partly loyal customers. Iranbin should apply appropriate marketing strategies to increase value of sales to customers of this cluster.

9. Conclusion

The clustering problem is a very important problem and has attracted much attention of many researchers. The *k*-means algorithm is a simple and efficient clustering method that has been applied to many engineering problems; nevertheless it suffers from several drawbacks due to its choice of initializations. This paper has developed a new hybrid algorithm for solving the clustering problem which is based on the combination of PSO, ACO and *k*-means algorithms. The algorithm has been implemented and tested on several well known real datasets and preliminary computational experience is very encouraging. In other word it has been proved that the FAPSO–ACO–K algorithm will definitely converge to optimal solution in almost runs. The FAPSO–ACO–K clustering algorithm developed in this paper can be applied when the number of clusters is known a prior.

References

[1] Y.T. Kao, E. Zahara, I.W. Kao, A hybridized approach to data clustering, Expert Systems with Applications 34 (3) (2008) 1754–1762.
[2] D.N. Cao, J.C. Krzysztof, GAKREM: a novel hybrid clustering algorithm, Information Sciences 178 (2008) 4205–4227.
[3] K.R. Zalik, An efficient *k*-means clustering algorithm, Pattern Recognition Letters 29 (2008) 1385–1391.
[4] K. Krishna, Murty, Genetic *k*-means algorithm, IEEE Transactions of System Man Cybernetics Part B-Cybernetics 29 (1999) 433–439.
[5] U. Mualik, S. Bandyopadhyay, Genetic algorithm-based clustering technique, Pattern Recognition 33 (2000) 1455–1465.
[6] M. Fathian, B. Amiri, A honey-bee mating approach on clustering, The International Journal of Advanced Manufacturing Technology (2007), doi:10.1007/s00170-007-1132-7..
[7] M. Laszlo, S. Mukherjee, A genetic algorithm that exchanges neighboring centers for *k*-means clustering, Pattern Recognition Letters 28 (16) (2007) 2359–2366.
[8] P.S. Shelokar, V.K. Jayaraman, B.D. Kulkarni, An ant colony approach for clustering, Analytica Chimica Acta 509 (2) (2004) 187–195.
[9] M.K. Ng, J.C. Wong, Clustering categorical data sets using tabu search techniques, Pattern Recognition 35 (12) (2002) 2783–2790.
[10] C.S. Sung, H.W. Jin, A tabu-search-based heuristic for clustering, Pattern Recognition 33 (5) (2000) 849–858.
[11] T. Niknam, J. Olamaie, B. Amiri, A hybrid evolutionary algorithm based on ACO and SA for cluster analysis, Journal of Applied Science 8 (15) (2008) 2695–2702.
[12] T. Niknam, B. Bahmani Firouzi, M. Nayeripour, An efficient hybrid evolutionary algorithm for cluster analysis, World Applied Sciences Journal 4 (2) (2008) 300–307.
[13] T. Niknam, B. Amiri, J. Olamaie, A. Arefi, An efficient hybrid evolutionary optimization algorithm based on PSO and SA for clustering. Journal of Zhejiang University Science A, 2008, doi:10.1631/jzus.A0820196.
[14] J. Kennedy, R. Eberhart, Particle swarm optimisation, vol. 4, in: Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, (1999), pp. 1942–1948.
[15] T. Niknam, A new fuzzy adaptive hybrid particle swarm optimization algorithm for non-linear, non-smooth and non-convex economic dispatch problem, Applied Energy (2009), doi:10.1016/j.apenergy.2009.05.016..
[16] Z.L. Gaing, A particle swarm optimization approach for optimum design of PID controller in AVR system, IEEE Transaction on Power Systems 19 (2) (2004) 384–391.
[17] T. Niknam, An approach based on particle swarm optimization for optimal operation of distribution network considering distributed generators, in: Proceedings of the 32nd Annual Conference on IEEE Industrial Electronics, IECON 2006, (2006), pp. 633–637.
[18] R.J. Mullen, D. Monekosso, S. Barman, P. Remagnino, A review of ant algorithms, Expert Systems 36 (2009) 9608–9617.
[19] N. Holden, A.A. Freitas, Web page classification with an ant colony algorithm, Parallel Problem Solving from Nature 3242 (2004) 1092–1102.
[20] J. Ji, N. Zhang, C. Liu, N. Zhong, An ant colony optimization algorithm for learning classification rules, in: WI'06: Proceedings of the 2006 IEEE/WIC, 2006, pp. 1034–1037.
[21] R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Transactions on Evolutionary Computation 6 (2002) 321–332.
[22] M. Dorigo, L.M. Gambardella, A study of some properties of Ant-Q, in: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, 1996, pp. 656–665.
[23] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperating learning approach to the travelling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[24] N.D. Monekosso, P. Remagnino, Phe-Q: a pheromone based Q-learning, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, 2001, pp. 1611–3349.
[25] N.D. Monekosso, P. Remagnino, The analysis and performance evaluation of the pheromone-Q-learning algorithm, Expert Systems 21 (2) (2004) 80–91.
[26] A. Dalli, Adaptation of the F-measure to cluster-based Lexicon quality evaluation, EACL, Budapest, 2003.
[27] J. Handl, J. Knowles, M. Dorigo, On the performance of ant-based clustering, Design and Application of Hybrid Intelligent Systems. Frontiers in Artificial Intelligence and Applications 104 (2003) 204–213.
[28] M. Raphael, Where did you come from? Direct Marketing 62 (2002) 36–38.
[29] G.J. Cullinan, Picking Them by Their Batting Averages: Recency-Frequency-Monetary Method of Controlling Circulation, Direct Mail/Marketing Association, New York, NY, 1977.
[30] L.F. Armando, E.A. Schmitz, P. Lima, F.S.P. Manso, Optimized RFV analysis, Marketing Intelligence & Planning 24 (2006) 106–118.