Expert Systems with Applications 39 (2012) 1582–1588

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Review

A particle swarm optimization approach to clustering

Tunchan Cura *

Istanbul University, Faculty of Business Administration, Quantitative Methods, Turkey

ARTICLE INFO

Keywords:
Particle swarm optimization
Clustering
Heuristics

ABSTRACT

The clustering problem has been studied by many researchers using various approaches, including tabu searching, genetic algorithms, simulated annealing, ant colonies, a hybridized approach, and artificial bee colonies. However, almost none of these approaches have employed the pure particle swarm optimization (PSO) technique. This study presents a new PSO approach to the clustering problem that is effective, robust, comparatively efficient, easy-to-tune and applicable when the number of clusters is either known or unknown. The algorithm was tested using two artificial and five real data sets. The results show that the algorithm can successfully solve both clustering problems with both known and unknown numbers of clusters.

1. Introduction

Clustering groups objects in a given dataset with respect to similarities between these objects. In fact, this clustering problem must often be solved as part of more complicated tasks in pattern recognition, image analysis and other fields of science and engineering (Zhang, Ouyang, & Ning, 2010). The similarity criteria used in clustering vary from author to author. However, most of the similarity criterion functions are non-convex and nonlinear such that the resulting clustering problem may have local minimum solutions. Moreover, they show exponential complexity in terms of the number of clusters, and thus, the clustering problem is NP-hard when number of clusters exceeds three (Welch, 1982).

A survey of relevant literature shows many studies on clustering. However, heuristic approaches have become more appropriate to solving large-size problems when the clustering problem becomes NP-hard. Thus, the majority of prior studies cover heuristics. Some related studies include tabu searching (Al-Sultan, 1995), genetic algorithms (Jiang, Wang, Chu, & Yu, 1997; Krishna & Murty, 1999; Murthy & Chowdhury, 1996), simulated annealing (Maulik & Mukhopadhyay, 2010; Selim & Al-Sultan, 1991; Sun, Xie, Song, Wang, & Yu, 1994), ant colonies (Shelokar, Jayaraman, & Kulkarni, 2004), a hybridized approach (Kao, Zahara, & Kao, 2008), and artificial bee colonies (Zhang et al., 2010). However, very few of those approaches have employed particle swarm optimization (PSO). Although Kao et al. (2008) used PSO in their study, it was not a pure PSO approach. They developed a hybrid technique combining PSO, Nelder–Mead simplex searching and *K*-means algorithm. Here we present a pure PSO approach to clustering that is efficient, effective, robust and easy to tune as compared to other methods.

The *K*-means algorithm (Kaufman & Rousseeuw, 1990) is a well-known approach to clustering. Its popularity depends on its simplicity and computational efficiency. However, that approach tends to fixate on local optima near the initial cluster centers, which are assigned randomly. Thus, many researchers have presented heuristic clustering algorithms to overcome this problem (Zhang et al., 2010).

Maulik and Bandyopadhyay (2002) proposed a genetic algorithm approach to clustering. They tested the algorithm on synthetic and real-life datasets to evaluate its performance. Another genetic algorithm approach called the genetic *K*-means algorithm was presented by Krishna and Murty (1999); they defined a basic mutation operator specific to clustering.

Selim and Al-Sultan (1991) presented a simulated annealing approach to the clustering problem and theoretically proved that a clustering problem's global optimum solution can be reached. Maulik and Mukhopadhyay (2010) also presented a simulated annealing approach to clustering. They combined their heuristic with artificial neural networks to improve solution quality. Different from widely-known methods, they preferred to use a similarity criterion function called the XB cluster validity index. Sung and Jin (2000) proposed a tabu search-based heuristic to clustering. They combined two procedures, namely, packing and releasing, using the tabu search algorithm.

Shelokar et al. (2004) presented an ant colony optimization (ACO) method for clustering. Their algorithm employs distributed agents that mimic the way real-life ants find the shortest path from their nest to a food source and back. Researchers tested the algorithm on several simulated and real datasets and showed that the algorithm performed quite well.

Kao et al. (2008) proposed a hybridized approach that combines the *K*-means algorithm, Nelder–Mead simplex search and PSO technique. The authors used the *K*-means algorithm alone to

* Tel.: +90 212 560 82 60; fax: +90 212 590 40 00.
E-mail address: tunchan@istanbul.edu.tr

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.07.123

generate one particle in the initial population. They set the population size to $3m + 1$, where m is the number of attributes that objects have. At each iteration, they implemented a Nelder–Mead search only on the best $m + 1$ particles, and then the rest of the population was moved toward the best particle of the whole population and toward the best neighbor.

Lastly, Zhang et al. (2010) presented the artificial bee colony (ABC) as a state-of-the-art approach to clustering. To tackle infeasible solutions, they adopted Deb's constrained handling method (Goldberg & Deb, 1991) instead of the greedy selection process usually used in the ABC algorithm. When they tested their algorithm, they found very encouraging results in terms of effectiveness and efficiency.

The rest of the paper is organized as follows. Section 2 describes the clustering problem. Section 3 briefly defines the PSO technique. Section 4 presents the proposed PSO approach. Section 5 provides the experimental results, and Section 6 concludes the paper.

2. The clustering problem

As briefly mentioned above, clustering involves gathering similar objects in the same cluster. Thus, a similarity metric between two objects must first be defined. Most researchers have used Euclidean distance for that purpose, which is derived from the Minkowski metric and is defined as (Zhang et al., 2010):

$$D(o_i, o_l) = \left(\sum_{j=1}^{m} (o_{ij} - o_{lj})^r\right)^{1/r} \Rightarrow D(o_i, o_l) = \sqrt{\sum_{j=1}^{m} (o_{ij} - o_{lj})^2} \quad (1)$$

where $D(o_i, o_l)$ is a function that yields a dissimilarity measure between object i and object l, and o_{ij} ($i = 1,\ldots,n$ and $j = 1,\ldots,m$) denotes the value of jth attribute of object i. Thus, the number of objects in the dataset and the number of attributes an object has are denoted by n and m, respectively. The goal of a clustering algorithm is to determine a partition $G = \{C_1, C_2, \ldots, C_K | \forall k : C_k \neq \emptyset \text{ and } \forall h \neq k : C_k \cap C_h = \emptyset\}$ such that objects that belong to the same cluster are as similar to each other as possible, while objects that belong to different clusters are as dissimilar as possible. To achieve this clustering goal, a measure of adequacy with respect to the partition must be defined (Zhang et al., 2010). Using the above notation, consider a given dataset of n objects in m dimensional space to be partitioned into K clusters. Thus, the mathematical formulation of the clustering problem can be described as follows (Shelokar et al., 2004):

$$Min \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} D(o_i, z_k) \quad (2)$$

$$\sum_{k=1}^{K} w_{ik} = 1, \quad i = 1, \ldots, n \quad (3)$$

$$\sum_{i=1}^{n} w_{ik} \geqslant 1, \quad k = 1, \ldots, K \quad (4)$$

where z_k denotes the center of cluster k; z_{kj} is an average of the jth attribute values of all objects in cluster k; $w_{ik} \in \{0,1\}$ denotes that object i belongs to cluster k if $w_{ik} = 1$ (otherwise $w_{ik} = 0$). Assuming that t indicates the iteration number, w_{ik}^t can be calculated as follows:

$$w_{ik}^t = \begin{cases} 1 & \text{if } k = \text{argmin}_{\delta=1,\ldots,K} D\left(o_i, z_\delta^{t-1}\right) \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \ldots, n, \; k = 1, \ldots, K \quad (5)$$

According to Eq. (5), we assign each object to the nearest cluster center out of the all cluster centers that were updated at iteration $t - 1$ using Eq. (6). This equation also guarantees that the constraint given in Eq. (3) is satisfied. Thus, the center of each cluster at iteration t can be obtained as follows:

$$z_{kj}^t = \frac{\sum_{i=1}^{n} w_{ik}^t o_{ij}}{\sum_{i=1}^{n} w_{ik}^t}, \quad k = 1, \ldots, K, \; j = 1, \ldots, m \quad (6)$$

3. The PSO technique

PSO is a heuristic technique recently introduced by Kennedy and Eberhart (1995). It is one of the latest evolutionary and population-based optimization algorithms, which can simulate bird flocking or fish schooling behavior. PSO searches for the optimum solution in the search space. However, this search process is not carried out entirely randomly. A problem-specific fitness function is employed to determine the next search step.

In the PSO algorithm, each individual in the population is called a particle and is subject to move in the search space. In addition, each particle is a candidate solution. Particles have memory, and thus, they retain part of their previous state. There is no restriction that particles share the same point in the search space, but their individuality is preserved. Each particle's movement is the composition of a velocity and two randomly-weighted influences. The two randomly-weight influences are individuality, or the tendency to return to its best previous position, and sociality, or the tendency to move towards its neighborhood's best previous position.

When compared to many of the other population-based approaches such as other well-known genetic algorithms, the convergence rate of the population is much slower for PSO (Kao et al., 2008). Thus, there is no need for any extra mechanism, such as the mutation operator in genetic algorithms, to diversify different areas of the search space.

4. Proposed PSO approach

As mentioned above, particles continuously move to search for better solutions, and movement depends on various topologies. This study follows the so-called "gbest neighborhood topology" described by Kennedy, Eberhart, and Shi (2001), according to which each particle remembers its best previous position and the best previous position visited by any particle in the whole swarm. In other words, a particle moves towards its best previous position and towards the best particle.

Let there be $K \times m$ dimensions, each representing an attribute value of a given center of a given cluster for each particle. This method will organize the swarm in this study, if two modifications are employed. First, each particle includes a variable that represents the number of clusters denoted by K_p ($p = 1,\ldots,P$), where P is the number of particles in the swarm. Second, each particle includes variables, each of which represents an attribute value of a given center of a given cluster denoted by z_{kj}^p. Thus, the number of dimensions that particle p has will be $K_p \times m + 1$.

4.1. Fitness function

Kennedy and Eberhart (1995) suggested a fitness value associated with each particle. Thus, a particle moves in the solution space with respect to its previous position when it has met the best fitness value and the neighbor's previous position when the neighbor has met the best fitness value. In this study, the fitness function is defined as:

$$f_p^t = \sigma_p^t = \sum_{k=1}^{K_p} \sum_{i=1}^{n} w_{ik}^{pt} D\left(o_i, z_k^{pt}\right) \quad (7)$$

where f_p^t is the fitness value of particle p at iteration t. Note that the vast majority of previous studies have assumed that the number of

clusters are given in the problem. In that case, the number of the clusters, for all of the particles, would be the same, that is, equal to the given number of clusters. Thus, $K_1 = K_2 = \cdots = K_p$. However, this study presents an algorithm that can also be employed for a clustering problem in which the number of clusters is unknown. In that case, we suggest using the following fitness function:

$$f_p^t = \sigma_p^t - \min_{k \neq l} D(z_k^{pt}, z_l^{pt}) \tag{8}$$

Note that when partitioning is compact and satisfactory, the value of σ_p^t should be low, while $\min_{k \neq l} D(z_k^{pt}, z_l^{pt})$ should be high, thereby yielding a lower values from the fitness function.

At each one of the iterations, a particle's personal best position and the best neighbor in the swarm are updated if an improvement in any of the best fitness values is observed.

4.2. Moving a particle

According to gbest neighborhood topology, a particle moves towards its best position and towards the best neighbor in the swarm. Indeed, this movement depends on its current velocity, which is defined as:

$$vK_p^{t+1} = vK_p^t + \omega_1\left(GK_b - K_p^t\right) + \omega_2\left(GK_p - K_p^t\right) \tag{9}$$

$$vz_{kj}^{p,t+1} = vz_{kj}^{pt} + \omega_1\left(Gz_{\partial j}^b - z_{kj}^{pt}\right) + \omega_2\left(Gz_{\partial j}^p - z_{kj}^{pt}\right) \tag{10}$$

where both ω_1 and ω_2 are uniform random numbers between zero and two, and b is the best particle in the swarm. vK_p^t and vz_{kj}^{pt} denote velocities of particle p on dimension K and on dimension z_{kj}, respectively. GK_p and Gz_{kj}^p denote the memorized best positions of particle p on dimension K and on dimension z_{kj}, respectively. Because both of the best solutions, that is, a particle's memorized best solution and the best solution of the swarm, can possibly have different numbers of clusters, we suggest using the nearest clusters of the best solutions for velocity calculations. Thus, ∂ denotes the index number of a given best solution's cluster, that is, the nearest cluster to the center z_k^{pt}, which is computed as follows:

$$\partial = \underset{l=1,\ldots,GK_b}{\operatorname{argmin}} D\left(Gz_l^b, z_k^{pt}\right) \quad or \quad \partial = \underset{l=1,\ldots,GK_p}{\operatorname{argmin}} D\left(Gz_l^p, z_k^{pt}\right) \tag{11}$$

As seen in Eq. (9), vK_p^{t+1} will always be zero if the number of clusters is known (that is, $GK_b = GK_p = K_p^t$) because initial velocities are set to zero in this study. Thus, particle p moves at iteration $t+1$ as follows:

$$K_p^{t+1} = \max\left(round\left(vK_p^{t+1} + K_p^t\right), 1\right) \tag{12}$$

$$z_{kj}^{p,t+1} = vz_{kj}^{p,t+1} + z_{kj}^{pt} \tag{13}$$

After moving particle p on dimension K_p, we suggest moving only one randomly-selected cluster center of current solution. In other words, in Eq. (13), k is a uniform random integer between one and K_p. However, before moving cluster centers, new cluster centers must be enlarged or some of the existing centers must be removed from the current solution for particle p if $K_p^{t+1} \neq K_p^t$, similar to the method proposed by Maulik and Mukhopadhyay (2010).

In the case in which $K_p^{t+1} < K_p^t$, we suggest identifying the smallest cluster and deleting its center from configuration. The cluster-removing operation shown in Fig. 1 is repeated until $K_p^{t+1} = K_p^t$. To identify the smallest cluster, the size S_k^{pt} of each cluster k is computed as follows:

$$S_k^{pt} = \sum_{i=1}^{n} w_{ik}^{pt} \quad k = 1, \ldots, K_p^t \tag{14}$$

In the case in which $K_p^{t+1} > K_p^t$, we suggest splitting the biggest cluster according to Eq. (14) into two new clusters and repeating that splitting operation until $K_p^{t+1} = K_p^t$. To split the biggest cluster, first its range is identified by finding the maximum and minimum values for each attribute, and then new cluster centers are determined randomly within that range. Fig. 2 shows the algorithm for the splitting operation.

4.3. Satisfying constraints

As we discussed previously, each particle would have been repositioned in $K_p \times m + 1$ dimensional search space at the end of

x = Find the smallest cluster of particle p using Eq. (14)
delete z_x^p
$K_p = K_p - 1$
Assign each object to the nearest cluster // see Eq. (5)

Fig. 1. Removing procedure.

x = Find the biggest cluster of particle p using Eq. (14)
identify the range of cluster x
for each $j = 1$ **to** m
    $z_{xj}^p = random\ number\ within\ the\ range$
End for
add new cluster q
for each $j = 1$ **to** m
    $z_{qj}^p = random\ number\ within\ the\ range$
End for
$K_p = K_p + 1$
Assign each object to the nearest cluster // see Eq. (5)

Fig. 2. The splitting procedure.

any iteration. We know that particles represent candidate solutions, and each particle must be feasible and satisfy Eqs. (3) and (4). However, as mentioned in Section 2, Eq. (5) guarantees that Eq. (3) is satisfied. Thus, we only need to focus on satisfying Eq. (4). In other words, we must make sure that at least one object belongs to each cluster of a given solution. Therefore, solutions that include empty clusters are not feasible. To address infeasible solutions, we adopted Deb's constraint handling method as per Zhang et al. (2010). According to that method, we can determine the choosing process between solutions a and b as follows:

(1) If $infeasibility_a < infeasibility_b$ then a is better than b
(2) If $infeasibility_a = infeasibility_b$ and $fitness_a < fitness_b$ then a is better than b

where $infeasibility_a$ is the number of empty clusters of solution a, and $fitness_a$ is the objective function value for solution a, which is given in Eq. (2).

In summary, the PSO heuristic used for clustering in this study is shown in Fig. 3.

4.4. Parameter tuning

The proposed PSO heuristic is very easy to tune because there are only two parameters used, namely, population size P and maximum iteration count *itermax*. To determine appropriate parameter settings, we generated two artificial problems that were used to test the ant colony approach of Shelokar et al. (2004) using a random number generator that produced a Gaussian distributed set of objects. In the first problem, the dataset (K) is composed of three clusters with 50 objects in each cluster. The data was generated using means $\mu_1 = [3,0]$, $\mu_2 = [0,3]$ and $\mu_3 = [1.5,2.5]$ and variances $\lambda_1 = [0.3,1]$, $\lambda_2 = [1,0.5]$, $\lambda_3 = [2,1]$. In the second problem, the dataset (K) is composed of six clusters with 25 objects in each cluster. The data was generated using means $\mu_1 = [3,0]$, $\mu_2 = [0,3]$, $\mu_3 = [1.5,2.5]$, $\mu_4 = [0.2,0.1]$, $\mu_5 = [1.2,0.8]$, $\mu_6 = [0.1,1.1]$ and variances $\lambda_1 = [0.3,1]$, $\lambda_2 = [1,0.5]$, $\lambda_3 = [2,1]$, $\lambda_4 = [0.03,1]$, $\lambda_5 = [2,0.5]$, $\lambda_6 = [0.2,0.4]$.

Four levels for each parameter were selected. For each instance and each level, we ran the algorithm five times. We used the relative increase given in Eq. (15) to select the proper values from the parameter levels. Table 1 shows the parameter levels and selected values.

Table 1
Parameter levels and selected values for PSO heuristic.

Parameter	Considered levels	Selected value
P	50, 100, 250, 500	250
itermax	$m \times 5$, $m \times 15$, $m \times 20$, $m \times 25$	$m \times 15$

```
t = 0
for p = 1 to P
    randomly generate initial particle p
    assign each object to the nearest cluster of particle p   // see Eq. (5)
    update cluster centers of particle p   // see Eq. (6)
    calculate f_p using Eq. (7) or Eq. (8)
    particle p memorizes the initial position as the best
end for
while t < itermax
    find the best particle of the swarm with respect to Deb' s rule
    t = t + 1
    for p = 1 to P
        move particle p on dimension K   // see Eq. (12)
        while K_p^t < K_p^{t-1}
            call removing procedure   // see Fig. 1
        end while
        while K_p^t > K_p^{t-1}
            call splitting procedure   // see Fig. 2
        end while
        k = random integer between 1 and K_p^t
        for j = 1 to m
            move particle p on dimension z_kj   // see Eq. (13)
        end for
        assign each object to the nearest cluster of particle p   // see Eq. (5)
        update cluster centers of particle p   // see Eq. (6)
        calculate f_p using Eq. (7) or Eq. (8)
        if the current position is better than the memorized position with respect to Deb' s rule then
            particle p memorizes current position as the best
    end for
end while
```

Fig. 3. PSO heuristic for clustering.

$$R = \frac{mean - Best}{Best} \times meantime \quad (15)$$

where *Best* is the minimum objective function value among all runs, and *mean* is the average objective function value of five runs. Finally, *meantime* is the average computation time for five runs.

5. Experimental results

In this section, we present the results obtained when searching for the solution of the problem formulated in Eqs. (1)–(6). The PSO approach presented in this study was compared to well-known algorithms, namely, the ACO proposed by Shelokar et al. (2004), an algorithm that is a combination of the *K*-means algorithm, Nelder–Mead simplex search and PSO technique (K-NM-PSO) as proposed by Kao et al. (2008) and the ABC algorithm proposed by Zhang et al. (2010).

Because we have not been able to contact any of the authors to obtain original source codes, all algorithms were programmed in Java, and Java codes were compiled under JDK 6. Algorithms are executed on an AMD Athlon X2 250 3.00 GHz computer running Microsoft Windows XP. The parameter settings for ACO, K-NM-PSO and ABC are set the same as in the original corresponding paper.

To evaluate these heuristics, we used seven datasets. Two of them are artificial datasets taken from Kao et al. (2008). The other five datasets, namely, iris, thyroid, wine, contraceptive method choice (CMC) and glass, which have been employed by many researchers to test the performance of their algorithms, are taken from Machine Learning Laboratory (Blake & Merz, 1998). The datasets used in this study can be described as follows:

Data set 1: Artificial dataset one (Art1). This dataset contains 600 objects with two attributes and four clusters. Samples were drawn from four independent bivariate normal distributions, where classes were distributed according to $\mu = \begin{pmatrix} \varphi_i \\ \varphi_i \end{pmatrix}$, $\Sigma = \begin{bmatrix} 0.5 & 0.05 \\ 0.05 & 0.5 \end{bmatrix}$, $i = 1, \ldots, 4$, $\varphi_1 = -3$, $\varphi_2 = 0$, $\varphi_3 = 3$, $\varphi_4 = 6$. μ is the mean vector, and Σ is the covariance matrix.

Data set 2: Artificial dataset two (Art2). This dataset contains 250 objects with three attributes and five clusters. Samples were drawn from five independent uniform distributions with ranges of [85,100], [70,85], [55,70], [40,55] and [25,40].

Data set 3: The iris dataset. This dataset contains three categories of 50 objects each, where each category refers to a type of iris plant. In the iris dataset, there are 150 instances with four attributes, which are sepal length in cm, sepal width in cm, petal length in cm and petal width in cm.

Data set 4: The thyroid gland dataset. This dataset contains three categories of human thyroid diseases, namely, euthyroidism, hypothyroidism and hyperthyroidism. In the thyroid gland dataset, there are 215 samples with five attributes that were evaluated with various laboratory tests, including the T3-resin uptake test, total Serum thyroxin as measured by the isotopic displacement method, total serum triiodothyronine as measured by radioimmuno assay, basal thyroid-stimulating hormone as measured by radioimmuno of 200 mg of thyrotropin releasing-hormone and the basal value (Zhang et al., 2010).

Data set 5: The wine dataset. This dataset contains chemical analyses of 178 wines derived from three different cultivars, with 13 attributes, namely, alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavonoids, nonflavonoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines and praline.

Data set 6: The CMC dataset. This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The objects are married women who either were not pregnant or did not know if they were at the time of interview. The problem involves predicting the choice of the current contraceptive method of a woman based on her demographic and socio-economic characteristics (Kao et al., 2008). This dataset contains 1473 objects with nine attributes and three clusters.

Data set 7: The glass identification dataset. This dataset contains 214 objects with nine attributes, namely, refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium and iron. The data were sampled from six different types of glass, that is, float-processed building windows, non-float processed building windows, float-processed vehicle windows, containers, tableware and headlamps.

To compare the performance of our algorithm with those of other approaches, algorithms were each run 10 times for each of the datasets. Table 2 provides the objective function values expressed in Eq. (2) obtained from the four clustering algorithms for the datasets described above. Because each of the algorithms was run 10 times, we reported the average of 10 objective function values and the best (i.e., minimum) and the worst (i.e., maximum) objective function values of 10 experiments, which indicates the range of values that the algorithms span. Note that for the comparisons given in Tables 2–4, we derived the PSO algorithm such that it employed the fitness function given in Eq. (7) and its initial particles were set equal to the given number of clusters on dimension K_p because all the other approaches assume that the number of clusters are known.

From the results given in Table 2, we can see that none of the previously-proposed algorithms outperforms the PSO approach in terms of the average and best objective function values for the seven datasets used here. Although PSO is outperformed by ACO and ABC in terms of worst objective function value for glass dataset,

Table 2
Comparison of objective function values for the four algorithms.

Data set	Criteria	K-NM-PSO	ACO	ABC	PSO
Art1	Average	161.08	718.47	158.51	158.51
	Best	158.51	622.57	158.51	158.51
	Worst	184.21	870.18	158.51	158.51
Art2	Average	2102.66	1940.25	1794.86	1788.70
	Best	1788.70	1836.72	1788.70	1788.70
	Worst	2671.54	2026.87	1850.31	1788.70
iris	Average	97.23	97.34	97.22	97.22
	Best	97.22	97.22	97.22	97.22
	Worst	97.33	97.83	97.22	97.22
thyroid	Average	1986.38	1987.19	1963.51	1960.71
	Best	1966.85	1965.81	1960.59	1960.59
	Worst	2012.93	2008.23	1973.04	1961.75
wine	Average	16534.52	16531.10	16530.54	16530.54
	Best	16530.54	16530.54	16530.54	16530.54
	Worst	16550.45	16536.19	16530.54	16530.54
cmc	Average	5542.05	8163.75	5542.77	5541.64
	Best	5541.64	7863.54	5541.65	5541.64
	Worst	5544.25	8415.07	5544.02	5541.64
glass	Average	225.95	219.90	214.84	213.41
	Best	208.93	216.30	208.91	202.92
	Worst	250.27	223.12	222.16	226.51

Table 3
Comparison of error rates for the four algorithms.

Data set	Criteria	K-NM-PSO	ACO	ABC	PSO
Art1	Average	0.10	16.18	0.00	0.00
	Best	0.00	14.74	0.00	0.00
	Worst	0.99	17.54	0.00	0.00
Art2	Average	5.18	3.02	0.12	0.00
	Best	0.00	0.94	0.00	0.00
	Worst	13.62	4.75	1.25	0.00
iris	Average	11.59	10.95	11.41	12.03
	Best	11.41	9.45	11.41	12.03
	Worst	12.03	11.41	11.41	12.03
thyroid	Average	34.31	36.23	36.71	35.10
	Best	21.33	32.53	36.14	35.04
	Worst	39.28	39.57	37.91	35.64
wine	Average	28.06	28.12	28.09	27.96
	Best	27.96	28.09	28.09	27.96
	Worst	28.09	28.44	28.09	27.96
cmc	Average	44.13	44.69	44.17	44.18
	Best	44.08	44.49	44.14	44.18
	Worst	44.21	44.85	44.20	44.18
glass	Average	33.36	26.93	32.18	30.63
	Best	31.60	26.42	26.62	27.45
	Worst	38.45	27.44	34.92	32.98

Table 4
Comparison of computation times for the four algorithms.

Data set	Criteria	K-NM-PSO	ACO	ABC	PSO
Art1	Average	0.023	5.959	3.116	0.630
	Best	0.015	5.921	3.109	0.625
	Worst	0.032	6.000	3.125	0.641
Art2	Average	0.024	2.680	1.952	0.588
	Best	0.015	2.656	1.938	0.578
	Worst	0.032	2.734	1.969	0.610
iris	Average	0.014	1.592	0.939	0.359
	Best	0.000	1.578	0.937	0.343
	Worst	0.016	1.610	0.953	0.375
thyroid	Average	0.036	2.402	1.541	0.697
	Best	0.031	2.390	1.531	0.687
	Worst	0.047	2.421	1.547	0.718
wine	Average	0.234	2.692	2.589	2.944
	Best	0.218	2.672	2.578	2.922
	Worst	0.250	2.703	2.625	3.000
cmc	Average	0.717	18.817	15.620	11.862
	Best	0.688	18.734	15.578	11.843
	Worst	0.750	18.906	15.656	11.906
glass	Average	0.205	3.014	4.480	3.342
	Best	0.187	2.984	4.437	3.328
	Worst	0.219	3.047	4.515	3.375

none of the previously-proposed heuristics outperforms the PSO approach in terms of the worst objective function value for the remaining datasets. The results also show that PSO is robust because the difference between the objective function values of the best and worst solutions are always very low and even zero for the most of the datasets.

Inspired by the study of Kao et al. (2008), we employed another criterion to measure solution quality called error rate (ER). This measurement equals the number of misplaced pair of objects divided by the total number of all pairs, expressed as percentage. It is calculated as follows:

$$ER = \left[\left(\sum_{i=1}^{n-1}\sum_{l=i+1}^{n}|A_{il}-B_{il}|\right)\Big/\left(\frac{n(n-1)}{2}\right)\right]\times 100 \qquad (16)$$

where n denotes the total number of objects. A_{il} is equal to one if objects i and l are members of the same cluster before clustering; otherwise, A_{il} is equal to zero. B_{il} is equal to one if objects i and l are members of the same cluster after clustering; otherwise, B_{il} is equal to zero.

When considering error rates, Table 3 shows that none of the four algorithms clearly outperforms the others in terms of the average, best and worst error rates for all datasets. Additionally, consolidating Tables 2 and 3, we note that PSO heuristic is obviously not less effective than any other previously-proposed approach, but it may be more effective in some cases.

Table 4 gives the computation time in seconds for the four algorithms applied to the seven datasets.

From these results, we note that except for K-NM-PSO, none of the previously-proposed algorithms clearly outperforms the PSO approach in terms of the average, best and worst computation times for all datasets. However, the PSO heuristic clearly outperforms ACO and ABC for the Art1, Art2, iris, thyroid and cmc datasets. Note that K-NM-PSO is very efficient because population size and number of maximum iterations were set very low by the researchers, that is, $3 \times m + 1$ and $m \times 10$, respectively. Also note that to reach the optimum solution with a lower population size and a fewer number of iterations, the K-NM-PSO heuristic employs the K-means algorithm and Nelder–Mead simplex search, which can also be adapted to any other heuristic. However, although K-NM-PSO is always far ahead of other heuristics in terms of efficiency, we note that the PSO heuristic is fairly efficient according to the results shown in Table 4.

Because our PSO algorithm is able to solve clustering problems in which the number of clusters is not known, Table 5 reports the results obtained by PSO algorithm when searching solutions of these types of problems. We again ran the algorithm 10 times for each dataset. Thus, the third, fourth and fifth columns of Table 5 show the fitness value described in Eq. (8), error rate (expressed as percentage) shown in Eq. (16) and computation time in seconds, respectively. The sixth column shows the real number of clusters for the respective datasets, while the seventh column provides the average number of clusters predicted by the PSO algorithm. The initial particles were randomly assigned to each dimension.

Table 5
The results of the PSO heuristic when the number of clusters is unknown.

Data set	Criteria	Fitness value	ER (%)	Time (s)	# of clusters real	Average # of clusters predicted
Art1	Average	151.216	1.554	5.598	4.00	4.50
	Best	140.403	0.000	4.078		
	Worst	154.283	6.255	9.765		
Art2	Average	1829.205	4.176	8.211	5.00	6.10
	Best	1584.399	0.896	4.204		
	Worst	2410.243	8.623	12.343		
iris	Average	78.449	15.819	5.125	3.00	4.80
	Best	75.434	12.412	3.750		
	Worst	83.352	21.969	5.875		
thyroid	Average	1560.735	35.454	12.306	3.00	4.80
	Best	1454.146	22.791	9.515		
	Worst	1788.311	38.535	13.938		
wine	Average	10443.604	28.623	519.986	3.00	5.00
	Best	10371.906	28.350	490.109		
	Worst	10563.697	31.080	562.531		
cmc	Average	4604.153	40.706	2298.795	3.00	5.00
	Best	4551.796	40.385	2133.438		
	Worst	4701.457	41.494	2466.562		
glass	Average	217.020	34.226	46.925	6.00	6.90
	Best	197.173	31.885	29.875		
	Worst	256.049	40.200	66.672		

According to the average error rates, the results show that the PSO heuristic finds quite similar solutions for problems with both a known number of clusters and an unknown number of clusters. However, in the case in which the number of clusters is unknown, robustness is reduced because the difference between the best and worst fitness values is relatively high for all datasets. Additionally, although efficiency considerably decreases, it is within tolerable limits for five the Art1, Art2, iris, thyroid and glass datasets. Consequently, we can say that PSO algorithm is still effective, and its computation times are mostly within acceptable limits when dealing with problems with an unknown number of clusters.

6. Conclusion

There have been many studies that develop algorithms to solve the clustering problem. In this paper, a new particle swarm optimization (PSO) algorithm, which using characterizes bird flocking or fish schooling behavior, is developed to solve the clustering problem. Differing from many of the previously-proposed approaches, the PSO algorithm can be applied both when the number of clusters is known as well as when this number is unknown. Computational experiments show that the proposed algorithm of this study is effective, robust, easy to tune and tolerably efficient as compared with other approaches.

References

Al-Sultan, K. S. (1995). A tabu search approach to clustering problem. *Pattern Recognition, 28*, 1443–1451.

Blake C. L. & Merz C. J. (1998). UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Goldberg D. E. & Deb, K. (1991). A comparison of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 69–93).

Jiang, J. H., Wang, J. H., Chu, X., & Yu, R. Q. (1997). Clustering data using a modified integer genetic algorithm (IGA). *Analytica Chimica Acta, 354*, 263–274.

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

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley & Sons.

Kennedy J. & Eberhart R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (vol. IV, pp. 1942–1948).

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence*. San Mateo, CA, SA: Morgan Kaufmann.

Krishna, K., & Murty, M. N. (1999). Genetic *K*-means algorithm. *IEEE Transactions on Systems Man and Cybernetics B Cybernetics, 29*, 433–439.

Maulik, U., & Bandyopadhyay, S. (2002). Genetic algorithm based clustering technique. *Pattern Recognition, 33*, 1455–1465.

Maulik, U., & Mukhopadhyay, A. (2010). Simulated annealing based automatic fuzzy clustering combined with ANN classification for analyzing microarray data. *Computers and Operations Research, 37*, 1369–1380.

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

Selim, S. Z., & Al-Sultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition, 24*, 1003–1008.

Shelokar, P. S., Jayaraman, V. K., & Kulkarni, B. D. (2004). An ant colony approach for clustering. *Analytica Chimica Acta, 509*, 187–195.

Sung, C. S., & Jin, H. W. (2000). A tabu search-based heuristic for clustering. *Pattern Recognition, 33*, 849–858.

Sun, L. X., Xie, Y. L., Song, X. H., Wang, J. H., & Yu, R. Q. (1994). Cluster-analysis by simulated annealing. *Computers and Chemistry, 18*, 103–108.

Welch, J. W. (1982). Algorithmic complexity – 3 NP-Hard problems in computational statistics. *Journal of Statistical Computation and Simulation, 15*, 17–25.

Zhang, C., Ouyang, D., & Ning, J. (2010). An artificial bee colony approach for clustering. *Expert Systems with Applications, 37*, 4761–4767.