Volume 5, Issue 3, March 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performing Automatic Clustering Using Active Clusters Approach with Particle Swarm Optimization

Sameema, Zahid Ansari
Dept. of Computer Science & Engineering
P.A. College of Engineering, Karnataka, India

Abstract ---- Data clustering is an important analysis in data mining. Due to its important role, many clustering methods have been proposed. Unfortunately, most of them require predefined number of clusters. Thus in this paper, we tried to overcome this problem by performing automatic clustering using active clusters approach with particle swarm optimization heuristic method (ACACAPSO). We used the concept of active clusters for the clustering of data. Also, we used K-means method to update the cluster centroids.

Keywords ---- Automatic clustering, Particle swarm optimization, Particle, Active clusters, K-means

I. INTRODUCTION

Unsupervised data clustering is an important analysis in data mining. Due to its important role, many clustering methods have been proposed. These methods can be roughly divided into two categories, namely partition and hierarchical clustering algorithms[23]. Partition algorithm divides a dataset into several clusters such that each instance is assigned to a particular cluster. The hierarchical algorithm constructs cluster step-by-step based on certain proximity measurement. There are two basic approaches applied in hierarchical algorithms: agglomerative and divisive approaches. Agglomerative starts with defining each instance as a cluster. It then iteratively merges the closest two clusters until optimal number of clusters are obtained. On the other hand, divisive hierarchical algorithm works by splitting a cluster into two smaller clusters until optimal number of clusters are obtained.

Many such clustering algorithms have been proposed, yet most of them require predefined number of clusters[23]. But, unavailable information regarding number of clusters is commonly happened in real-world problems. An appropriate number of clusters is an important parameter in clustering since it may affect the final clustering results. Thus, we tried to overcome this problem by proposing automatic clustering based on particle swarm optimization heuristic method which performs clustering automatically using the concept of active clusters.

Particle Swarm Optimization (PSO) is a population-based search algorithm which simulates bird behavior in finding food([4], [23]). In PSO, birds represented as particles are employed to perform solution exploration. In this paper, PSO algorithm is used to overcome automatic clustering problem. Generally, there are two main issues in performing automatic clustering, namely determining number of clusters and finding cluster centroids. In order to solve these two problems, we perform automatic clustering using active clusters approach with particle swarm optimization (ACACAPSO) which does not require to specify the exact number of clusters by the user in advance.

In PSO, the swarm consists of set of particles. In ACACAPSO, each particle provides its own clustering solution. In each iteration, the particles' positions and velocities are updated to improve the clustering. As the particles are explored, they provide different clustering of data. Each time, the particle that provides the best solution based on its fitness value is selected as the victim. This is continued until optimal solution has been found.

Automatic clustering generally provides better clustering solution for the given data. Unlike many of the traditional clustering methods, it does not require to specify the number of clusters manually and hence avoids manual interpretation in performing clustering which in turn improves the final result.

II. LITERATURE REVIEW

A. Particle Swarm Optimization (PSO)

PSO is a population based stochastic optimization technique first developed by Dr. Eberhart and Dr. Kennedy in 1995[6]. It is mainly developed from swarm intelligence and is based on the research of social behavior of bird flocking and fish schooling. Thereafter, it has received a lot of attentions from the researchers around the globe. The idea was developed based on the artificial life concepts described by many authors previously.

Craig W. Reynolds presented his distributed model on flocks, herds and schools[1] in 1987. Heppner, Frank, and Ulf Grenandar presented simulations of bird flocking[2]. Both of these models relied on the fact that synchrony of birds' flocking behavior heavily depends on a function of birds' efforts to maintain ideal distance between themselves and their neighbors. A Socio-biologist, E. O. Wilson[5] based on his work on fish schooling explains that individual members can be benefited for their discovery from the previous knowledge of all the members in the school.

B. Why word 'Particle'?

Kennedy and Eberhart selected the word 'particles' instead of 'points' even though the swarm members are mass-less and volume-less[7]. This is because the particles in the swarm have position and velocities and in general, velocities are applied to particles.

C. PSO Topology

There are two main topologies of PSO algorithm, global version of PSO and local version of PSO([6], [8], [20]). In the global version of PSO, the particle's velocity is adjusted according to its personal best performance achieved so far and the best performance achieved so far by all the particles in the swarm. In this case, every particle should have information about its best performance and best performance of every other particle in the group. In the local version of PSO, each particle's velocity is adjusted according to its best performance achieved so far and the best performance achieved so far within its neighborhood. Hence in this case, each particle has information about best performance of itself and best performance of its neighbors[6]. This local version of gBest value is called as lBest.

D. Automatic Clustering using Active Clusters with PSO

The main drawback of many clustering algorithms is the need to specify the number of clusters by the user in advance. Since, different number of clusters might result in different types of clustering for the same data, we should be able to decide optimal number of clusters. In this paper, we performed automatic clustering using PSO[23] by using the idea of active clusters and we consider only the current active clusters for the calculation of fitness value of the particle and hence pBest and gBest values of particles each time.

III. METHODOLOGY

We performed automatic clustering using particle swarm optimization. It uses the concept of particles where each particle provides different clustering solution for the given data. Also, we performed clustering based on current active clusters within each particle. We select the particle providing best fitness value as best solution in each iteration until optimal number of iterations have been performed.

Terminologies used:

Let *SWARM_SIZE* be the total number of particles in the swarm
TOTAL_ITERATION be the total number of iterations executed
t be the current iteration
Nmax be the maximum number of clusters
M be the number of attributes of the data points

A. Initializing Swarm of Particles

Each particle is created by setting the position and velocity. The positions and velocities are set along different dimensions. We considered two dimensions X and Y in this paper. We also create the particle structure and decide some maximum number of clusters '*Nmax'* for the particles. Then the centroids for each of these clusters are found. In this way, we create required number of particles, initialize them and add them to the swarm to form a swarm of particles.

1) Setting Initial Position and Velocity for the Particle : In order to limit the particles' exploration within search space, we set some constant low and high values along all dimensions and also for velocities. Then the initial positions and velocities among these dimensions are set randomly based on above constant values. In this way, all the particles in the swarm are initialized.

2) Particle Creation: Each particle or solution has same structure. It consists of "***Nmax + M * Nmax***" bits, where

Nmax is the maximum number of clusters for the given particles
M is the number of attributes in the data

The first *'Nmax'* bits specify the clusters of the particles and the next '*M * Nmax'* bits specify the cluster centroids. All the bits are continuous numbers in the range [0,1]. This is shown in following figure.

1	2	….	Nmax	Nmax +1	Nmax + 2	….	Nmax + M * Nmax

Fig.1 Particle or Solution Representation

The first *Nmax* bits of each particle decide the active and inactive clusters for that particle. i.e., among the first *Nmax* bits, if the bit is "*equal to or greater than 0.5"*, then it is an active cluster otherwise, inactive. Only the active clusters are considered for the current clustering. In this way, the number of clusters are decided without manual specification based on their first *Nmax* bits. These *Nmax* bits are generated randomly as double numbers.

Then from the data set, *Nmax* data points are selected randomly and assigned to the next *M * Nmax* bits as the centroids of corresponding *Nmax* clusters of particle. These will be the initial centroids for the particles' clusters. The cenriods information may be kept separately for faster retrieval. Finally all the particles are added to the swarm.

For every particle from 1 to SWARM_SIZE
Randomly generate X position
Randomly generate Y position

Randomly generate velocity along X dimension
Randomly generate velocity along Y dimension
Set these as initial position and velocity for the particle
Create particle structure
Generate first *Nmax* bits of the particle as random double numbers to indicate clusters
Select Nmax data points of the data set randomly and assign them to next *M* * *Nmax* bits of the particle as cluster centroids
Add the particle to the swarm

Fig. 2 Algorithm for Initializing Swarm

Once the swarm of particles is created, then for each iteration, the following tasks will be performed.

B. Clustering

Within each particle, once their cluster centroids are found, we calculate the distance of each data point to all the cluster centroids of particle by using Euclid's distance. The data point is added to the cluster to which the Euclid's distance of the data point is the minimum.

If (a,b) is the data point and (x,y) is the cluster centroid, then their Euclid's distance is calculated using the Euclid's distance formula as follows :

$$\text{sqrt}((a-x)^2 + (b-y)^2) \quad (1)$$

This process is repeated inside each particle so that the data points are added to the desired clusters within each particle .

For every particle from 1 to SWARM_SIZE
For every data point in the data set
For every cluster from 1 to Nmax in particle
Find the Euclid's distance from the data point to the cluster centroid
Select the cluster to which the distance of the data point is the minimum
Assign the data point to this cluster having minimum distance

Fig. 3 Algorithm for clustering the data

C. Evaluating Active Clusters

One of the important aspects we use in ACACAPSO is evaluating active clusters since only the active clusters are considered for clustering at any point of time. The clusters which have the value "*equal to greater than 0.5*" in their corresponding bits in the particle representation are considered as active clusters, otherwise inactive.

Also we consider the infeasibility case here. i.e., if the number of active clusters within any particle is less than two, then we select any of the two clusters randomly such that at least one of them is currently inactive and make them active by adding 0.5 to their corresponding bit values. This will make sure that always there will be at least two active clusters within each particle.

For every particle from 1 to SWARM_SIZE
Calculate number of active clusters
If number of active clusters is less than 2
Select any two clusters randomly with at least one of them is inactive
Make them active by adding 0.5 to their corresponding bit values in the particle

Fig. 4 Algorithm for Evaluating Number of Active Clusters

The above algorithm results in final number of active clusters for each particle before considering the second infeasibility case (explained below).

D. Evaluating Cluster Size

Here for every particle, we calculate the number of active clusters and the number of data points within each of the particle's *Nmax* clusters. A second infeasibility case is considered for the number of data points within each active cluster. i.e., if the number of data points in any of the active clusters is less than two, we have to move the data points among the clusters of the particle to make sure that every active cluster will have at least two data points. In such case, we may encounter the following two possibilities.

Case 1 : The number of data points in the active cluster is less than two and the total number of active clusters in the particle is more than two

For every active cluster with this case, we first deactivate or make that cluster inactive by deducting 0.5 from its corresponding bit value. It may contain either 0 or 1 data point within it. If it contains one data point, then we need to move that data point to any other active cluster such that the Euclid's distance between the centroids of these two clusters is the minimum. Then we reduce the number of active clusters for that particle by one.

Case 2 : The number of data points in the active cluster is less than two and the total number of active clusters in the particle is less than or equal to two

For every active cluster with this case, we select one of the inactive clusters with at least two data points and move the last two data points of selected inactive cluster to this active cluster. We then update the number of data points for these two clusters by adding two to the active cluster in consideration and reducing two from the inactive cluster from which the data points are moved.

```
For every particle from 1 to SWARM_SIZE
    For every active cluster
        If number of data points is less than 2 and the total number of active clusters is more than 2
            Deactivate the cluster
            If number of data points is 1
                Select any other active cluster such that the
                Euclid's distance between their centroids is minimum
                Move the data point to that cluster
            Reduce number of active clusters by 1
        If number of data points is less than 2 and the total number of active clusters is less than or equal to 2
            Select any of the inactive cluster with at least 2 data points
            Move the last 2 data points from that cluster to this active cluster
            Increase the number of data points in this active cluster by 2
            Decrease the number of data points in the inactive cluster
    from which the data points are moved by 2
```

Fig. 5 Algorithm for Evaluating Active Cluster Size

This will give the final number of active clusters within each particle for further processing in the current iteration.

E. Calculation of Fitness Value

The next step is to calculate the fitness value for each particle. For each particle, we calculate the fitness value as follows: For every active cluster in the particle, we find Euclid's distance of the data points to the cluster centroid to which they belong. Within each active cluster, the distances of the data points are summed up. Finally we find the total sum of the sums within each acive cluster of the particle. The reciprocal of this total sum is returned as the fitness value for that particle.

```
For every particle from 1 to SWARM_SIZE
    For every active cluster
        Find the Euclid's distance of each data point to the cluster centroid
        Find the sum of these distances
    Find the total sum of the sums in each active cluster
    Return the reciprocal of this total sum as the fitness value of the particle
```

Fig. 6 Algorithm for Calculation of Fitness Value

F. Calculation of Particle Best (pBest) and Global Best (gBest) Values

The central point of PSO method is to find the *particle best* or *local best (pBest)* values for each particle and selecting the best pBest value as the *global best* or *social best (gBest)* value of the swarm.

For the first iteration, the particle *fitness values* are themselves considered as their *pBest* values. For every next iteration, the current *fitness values* are compared with the previous *pBest* values and minimum among them will be considered as *pBes*t values for that iteration. Finally, the minimum among all the particles' *pBest* values is selected as the *gBest* value and the particle whose *pBest* value has been selected as the *gBest* value will be considered as optimal solution for that iteration. i.e., this will provide the best clustering of data for that iteration.

```
For every particle from 1 to SWARM_SIZE
    Calculate fitness value
    If it is first iteration
        Assign fitness value as particle's pBest value
    Else
        If fitness value is less than previous pBest value
            Assign this fitness value as particle's pBest value
        Else
            Retain previous pBest value as particle's pBest value for this iteration too
Select the minimum among all pBest value as gBset value which gives the solution for the current iteration
```

Fig. 7 Algorithm for Calculation of pBest and gBest Values

G. Updating Particle Position and Velocity

Once we finish calculating the best solution for the current iteration, we need to update position and velocity of all the particles. We use the following equations for updating position and velocity.

$$v_i^{t+1}(x) = w * v_i^t(x) + c_1 * r_1 * (pBest^t - x_i^t) + c_1 * r_2 * (gBest^t - x_i^t) \quad (2a)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}(x) \quad (2b)$$

$$v_i^{t+1}(y) = w * v_i^t(y) + c_1 * r_1 * (pBest^t - y_i^t) + c_1 * r_2 * (gBest^t - y_i^t) \quad (2c)$$

$$y_i^{t+1} = y_i^t + v_i^{t+1}(y) \quad (2d)$$

Where, $v_i^{t+1}(x)$ is the new velocity of particle i along X dimension for iteration (t+1)
$v_i^t(x)$ is the current velocity of particle i along X dimension at iteration t
x_i^{t+1} is the new position of particle i along X dimension X for iteration (t+1)
x_i^t is the current position of particle i along X dimension at iteration t
$v_i^{t+1}(y)$ is the new velocity of particle i along Y dimension for iteration (t+1)
$y_i^t(y)$ is the current velocity of particle i along Y dimension at iteration t
y_i^{t+1} is the new position of particle i along Y dimension for iteration (t+1)
y_i^t is the current position of particle i along Y dimension at iteration t
w is the inertia value which controls exploration and exploitation of the particles
c_1 & c_1 are the acceleration coefficients
r1 & r2 are the random numbers in the range [0,1]
$pBest^t$ is the particle best value of particle i at iteration t
$gBest^t$ is the global best value at iteration t

1) Inertia Value (w): ACACAPSO applies time varying inertia weight. Herein, the inertia weight linearly decreases based on the following equation (3).

$$w = W\_MAX - ((t/TOTAL\_ITERATION) * (W\_MAX - W\_MIN)) \quad (3)$$

Where,
w is the inertia weight for the current iteration
W_MAX and W_MIN are the maximum and minimum inertia weight respectively
t is the current iteration
TOTAL_ITERATION is the total number of iterations to be executed

When the equations [2a – 2d] were first suggested, the researchers did not include inertia weight. Inertia weight was added later to better control the exploration and exploitation of the particles. It was first reported in the literature in 1998 (Shi and Eberhart 1998a, 1998b)[3]. Inertia weight improves performance. Suitable selection of inertia weight provides a balance between global and local exploration and exploitation and also results in fewer iterations on average to find a sufficiently optimal solution.

Linearly decreasing inertia weight was suggested by original authors with values decreasing linearly from 0.9 to 0.4 during a run. Zheng and Yong-Ling asserted that PSO with increasing inertia weight performs better([4], [17]). Another approach to use inertia weight is to adapt it using a fuzzy system. This was first reported in the paper published by Eberhart, and Shi in 2000 [16]. It takes two inputs, global fitness value for the current iteration and current inertia weight and gives change in the inertia as output.

2) Acceleration coefficients(c1 & c2): The terms c1 and c2 are called as acceleration coefficients. These terms control how far a particle will move in a single iteration[10]. Generally, these are set to 2.7([8], [11], [13]) since with this value, search can cover all surrounding regions centered at the pBest and gBest values. A value of 1.49445 was used by Clerc[14]. Also, it is shown that assigning different values to c1 and c2 improves performance[12]. Low values allow particles to traverse far from target regions before being draw back. High values result in blunt movement toward, or past, target regions[8].

For every particle from 1 to SWARM_SIZE
Calculate inertia weight using equation (3)
Calculate new velocity for the particle along X dimension using equation (2a)
Calculate new X position for the particle using equation (2b)
Calculate new velocity for the particle along Y dimension using equation (2c)
Calculate new Y position for the particle using equation (2d)
Set the new X and Y positions to the particle

Fig. 8 Algorithm for Updating Particle Position and Velocity

H. Updating Active Cluster Centroids using K-means

After finding the solution for the current iteration, we need to update all active cluster centroids of each particle for the next iteration. We use K-means method to update the centroids as described below.

Let (x_i, y_i) be the 'n' data points belonging to the active cluster C where i = 1, 2, … , n. We calculate mean of all 'x_i' values and mean of all 'y_i' values in C. Let this be (x_{avg}, y_{avg}). Now we update the cluster centroid by assigning this

average (x_{avg}, y_{avg}) to it which becomes the centroid for this cluster for next iteration. This is repeated for each active cluster of all the particles to form the new centroids for the next iteration.
Mathemeatically,

$$Xsum = \Sigma x_i \qquad Ysum = \Sigma y_i \qquad (4a)$$

$$x_{avg} = Xsum/n \qquad y_{avg} = Ysum/n \qquad (4b)$$

For every particle from 1 to SWARM_SIZE
    For every active cluster C
        Find the mean of all x values of data points in C
        Find the mean of all y values of data points in C
        Assign this average as centroid for this cluster for the next iteration
    For every inactive cluster
        Retain same centroid

Fig. 9 Algorithm for Updating Active Cluster Centroids using K-means

I. Reclustering Data

We recluster the data according to the new centroids using same clustering method decribed in III-B section.

J. Regenerating Particle Structure

Now we regenerate all the particles so that their structure changes for the next iteration. The first *Nmax* bits are again generated randomly. This will change the previous number of active and inactive clusters of the particle. Then, the newly found centroids (obtained in III-H) are assigned to the next *M * Nmax* bits.

For every particle from 1 to SWARM_SIZE
    Regenerate first *Nmax* bits as random double numbers to indicate clusters
    Assign newly calculated cluster centroids to next *M * Nmax* bits

Fig. 10 Algorithm for Regenerating particle Structure

The following algorithm is the overall ACACAPSO algorithm.
Initialize swarm of SWARM_SIZE particles
Cluster the data within each particle
For every iteration from 1 to TOTAL_ITERATION
    Evaluate active clusters of the particles
    Evaluate size of all the active clusters of the particles
    Calculate current fitness value of each particle
    Calculate pBest value of each particle
    Calculate gBest value of the swarm
    Update position and velocity of each partile for the next iteration
    Update cluster centroids of all particles using K-means for the next iteration
    Recluster the data in each particle according to new centroids for the next iteration
    Regenerate structures of all the particles for the next iteration

Fig. 11 Overall ACACAPSO Algorithm

IV. CONCLUSION

In this paper, we primarily concentrated on performing automatic clustering. We used the particle swarm optimization heuristic method and the concept of active clusters to improve the clustering result. We also used K-means to update centroids of the clusters of each particle as particles are explored in search area. Automatic clustering generally provides better clustering solution for the given data. Unlike many of the traditional clustering methods, it does not require to specify the number of clusters manually in advance and hence avoids manual interpretation in performing clustering which in turn improves the final result.

REFERENCES

[1] Reynolds, Craig W., "Flocks, herds and schools: A distributed behavioral model." *ACM Siggraph Computer Graphics* 21.4 (1987): 25-34.

[2] Heppner, Frank, and Ulf Grenander. "A stochastic nonlinear model for coordinated bird flocks." *AMERICAN ASSOCIATION FOR THE ADVANCEMENT OF SCIENCE, WASHINGTON, DC(USA). 1990.* (1990).

[3] Shi, Yuhui, and Russell Eberhart. "A modified particle swarm optimizer." *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998.

[4] Hu, Xiaohui, Yuhui Shi, and Russell C. Eberhart. "Recent advances in particle swarm." *IEEE congress on evolutionary computation*. Vol. 1. 2004.

[5] Engelbrecht, Andries. "Particle swarm optimization: Velocity initialization."*Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012.

[6] Eberhart, Russ C., and James Kennedy. "A new optimizer using particle swarm theory." *Proceedings of the sixth international symposium on micro machine and human science*. Vol. 1. 1995.

[7] James Kennedy, and Russell Eberhart, "Particle swarm optimization", *Encyclopedia of Machine Learning*. Springer US, 2010. 760-766.

[8] Eberhart, Russell C., and Yuhui Shi. "Particle swarm optimization: developments, applications and resources." *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. Vol. 1. IEEE, 2001.

[9] Fan, H., and Y. Shi. "Study on Vmax of particle swarm optimization." *Proc. Workshop on Particle Swarm Optimization, Purdue School of Engineering and Technology*. Indianapolis, IN, Apr, 2001.

[10] Van den Bergh, Frans, and Andries Petrus Engelbrecht. "A cooperative approach to particle swarm optimization." *Evolutionary Computation, IEEE Transactions on* 8.3 (2004): 225-239.

[11] R. C. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*: Academic, 1996, ch. 6, pp. 212–226.

[12] Suganthan, Ponnuthurai N. "Particle swarm optimiser with neighbourhood operator." *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 3. IEEE, 1999.

[13] Hu, Xiaohui, Yuhui Shi, and Russell C. Eberhart. "Recent advances in particle swarm." *IEEE congress on evolutionary computation*. Vol. 1. 2004.

[14] Clerc, Maurice. "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization." *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 3. IEEE, 1999.

[15] Shi, Yuhui, and Russell C. Eberhart. "Empirical study of particle swarm optimization." *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 3. IEEE, 1999.

[16] Eberhart, Russ C., and Yuhui Shi. "Comparing inertia weights and constriction factors in particle swarm optimization." *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. Vol. 1. IEEE, 2000.

[17] Zheng, Yong-Ling, et al. "On the convergence analysis and parameter selection in particle swarm optimization." *Machine Learning and Cybernetics, 2003 International Conference on*. Vol. 3. IEEE, 2003.

[18] Clerc, Maurice, and James Kennedy. "The particle swarm-explosion, stability, and convergence in a multidimensional complex space." *Evolutionary Computation, IEEE Transactions on* 6.1 (2002): 58-73.

[19] Innocente, Mauro Sebastián, and Johann Sienz. "Particle swarm optimization with inertia weight and constriction factor." *Proceedings of the International conference on swarm intelligence (ICSI'11)*. 2011.

[20] Kenndy, J., and R. Mendes. "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. Systems, Man, and Cybernetics, Part C: Applications-and-Reviews." *IEEE-Transactions* 36.4 (2006): 515-519.

[21] Doval, Diego, Spiros Mancoridis, and Brian S. Mitchell. "Automatic clustering of software systems using a genetic algorithm." *Software Technology and Engineering Practice, 1999. STEP'99. Proceedings*. IEEE, 1999.

[22] Das, Swagatam, Ajith Abraham, and Amit Konar. "Automatic clustering using an improved differential evolution algorithm." *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 38.1 (2008): 218-237.

[23] R. J. Kuo, and Ferani E. Zulvia, "Automatic clustering using an improved particle swarm optimization", *Journal of Industrial and Intelligent Information* Vol. 1, No. 1, March 2013.

[24] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." *Applied statistics* (1979): 100-108.

[25] Omran, Mahamed GH, Andries P. Engelbrecht, and Ayed Salman. "An overview of clustering methods." *Intelligent Data Analysis* 11.6 (2007): 583-605.

[26] Jain, Anil K., M. Narasimha Murty, and Patrick J. Flynn. "Data Clustering : a review." ACM computing surveys *(CSUR)* 31.3 (1999):264-323

[27] Neshat, Mehdi , et al. "A new cooperative algorithm based on PSO and k-means for data clustering." *Journal of Computer Science* 8.2 (2012):188

