Indian J.Sci.Res. 4 (3): 335-353, 2014
ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

# ANALYSIS OF THE USAGE OF CHAOTIC THEORY IN DATA CLUSTERING USING PARTICLE SWARM OPTIMIZATION

**SAMAN POURSIAH NAVI[a1], EHSAN TOREINI[b], MARYAM MEHRNEJAD[c] AND S. KAZEM SHEKOFTEH[d]**

[a]Department of Computer Engineering, Quchan Branch, Islamic Azad University, Quchan, Iran
[bc]Centre of Software Reliability, School of Computing Science, Newcastle University, Newcastle Upon Tyne, UK
[d]Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

## ABSTRACT

**Clustering has been a popular topic of research for many years. In fact, the first clustering methods were statistical methods which were used prior to computer age. Clustering plays an important role in lots of sciences such as medicine, astronomy, economy, etc. In recent years, application of artificial intelligence methods in clustering gets more attention. Particle Swarm Optimization is one of the optimization algorithms which proposed in 1995 and in spite of being relatively new, has gained popularity. Using PSO algorithms in clustering started from 2003 and from that time, different methods has been proposed. A clustering method should partition the data set so that the most related data items are placed in the same group. In this paper, we tried to make a novel attempt to apply PSO and chaotic theory to clustering problem. In our proposed solution, the fitness function is changed and then, using chaotic map functions, the initialization method of PSO algorithm has been altered. Then, we tried to improve the results using a specified mutation technique called chaotic mutation. Finally, we applied inertia weights to improve our results to a new level. The empirical results shows using the suggested fitness function alongside chaotic and inertia weights improve the results hugely.**

**KEYWORDS:** Clustering, Particle Swarm Optimization, Fuzzy Clustering, Chaos Theory, Chaotic PSO, CPSO, Data Mining

Popularity of internet and computers, along with the increasing usage of computer software interacting with different sources of data has provided us with vast amount of data stored in physical devices. This data could be valuable source of information if we know how to use, considering this issues, the importance of more advanced and more efficient Data Mining techniques seems to be a good trend of research.

Data Clustering, which will be defined later, as one of the methods of Data Mining, is not a new trend of analyzing data, but by nature, has always been interested by the researchers. More efficient algorithms in clustering could affect our everyday cyber life such as web searching, recommender systems and even our social network interactions.

In this paper, we are going to discuss the impact of using chaotic theory in Clustering of Data using Particle Swarm Optimization Algorithm.

Particle Swarm Optimization, as PSO, is an evolutionary based optimization algorithm that is influenced by the patterns of movement in animal herds toward a pre-defined destination or in search of a goal. This algorithm is relatively new comparing to other types of evolutionary algorithms and was proposed in 1995 by an Electrical Engineer and a Sociologist.

The usage of PSO in the Data clustering problem is first mentioned in 2003 paper of Merwe and Englbrecht[1] which puts a simple but efficient solution that overcomes K-Means algorithm in experiments. This paper was a starting point of applying PSO to Data Clustering which is still open area of research.

Different papers were published in diverse conferences and journals which a glance of the most important innovations in this field will be discussed in the further sections.

In this paper, we suggest different ways of dealing with the particles of PSO algorithm using chaotic theory. We study the influence of chaos equations to different parts of the PSO Clustering and do different types of experiments to check the efficiency of each proposed ideas, individually.

The Structure of the paper is as follows:

The section 1 is the introduction to data mining and data clustering, to be more specific. We discuss different types of clustering and mention some important algorithms in each category. Then, we go deeper into

[1]Corresponding author

particle swarm optimization algorithm and its different types and finally, we end the section with chaos theory and the chaotic equations. In section 2, we provide the state of the art of PSO data clustering and review the researched done in the hybridization of the PSO algorithm with Chaos Theory. Section 3 is about our proposed method of clustering. We demonstrate our work and our proposed algorithms and ideas are discussed more precisely in this section. In section 4, our proposed ideas in the previous section are put into trial and illustrate our experiment results and find which of our proposed algorithms produces the best experimental results. Finally, in section 5 we sum up the paper and conclusion and future suggestions and trends will be discussed.

**Data Mining**

Data Mining is the art of extraction of knowledge from the raw data. This concept has been around ever since the humans live in this world. All humans always need to gather the knowledge around them from different sources in order to manage to live, but the researching this field of study dates back to the mathematics. The first methods of Data Mining are statistical ones. The computer and implementation of algorithms made new looks to the Data Mining and ever since, different algorithms have been discovered and practiced. The research for Data Mining is still open and different algorithms are proposed in order to improve the concept.

In this section, we go deep into the meaning of Data Mining and in particular, we talk about different methods of Data Clustering, as a way of mining the data. Data mining, as is defined in [2], is analytical phase of knowledge discovery in large databases. The Data mining is the intersection of different sciences like statistics, artificial intelligence, machine learning and database systems. As told before, the goal of data mining is to extract knowledge from the raw data and transform it into human understandable form so that we could utilize the knowledge for our purposes.

**Knowledge Discovery in Databases (KDD) Process**

As is mentioned in [2], the process of knowledge extraction from Data is within these steps:

1) Definition: In this step, we define the domain of the problem and understand the goal of our analysis from the customer point of view. This step is one of the most important phases because if we select the wrong criteria for our Data, the results could be far away from the desired ones.
2) Selection: In this step, we select the required data for our analysis process. In this phase, we select the subset of Data for our analysis or gather various data to form a new dataset.
3) Pre-Processing: The Data that have to be analyzed must have the proper attributes from which we want to extract knowledge. This step is the process of choosing the right data for our analysis step, cleaning this data from missing cells and noises and finally preparing the results for the next step toward mining.
4) Transformation: In this step, the useful features of the selected dataset that could influence the analysis process are chosen. These features depend on the data mining task you want to process in the next step.
5) Data Mining: This step is the main one in the KDD process. In this phase, we apply one of the data mining tasks to our clean and properly chosen dataset and produce knowledge from the raw data.
6) Evaluation: In this final phase, we evaluate the knowledge produced in the previous step to check whether the knowledge is adapted to the user's goals or not.

The data mining is our focus in this paper. Diverse methods are suggested for the data mining, depending on the user needs and the nature of the knowledge to be acquired. We can categorize different algorithms for Data mining in the following tasks:

1- Association, 2-Classification, 3- Clustering, 4- Prediction, 5- Sequential Patterns

**Data Clustering**

Nowadays data have an important role in all aspects of human life and so, analyzing theses data for discovering proper knowledge, will causes huge changes in our life. Data mining tasks are methods that their purpose is to conclude from raw data. Data clustering is an unsupervised data mining task that its goal is to categorize data into some groups that their characteristics are unknown. These groups called clusters and the task of grouping is called clustering. The exact meaning of data clustering is not globally agreed so in different sciences, different definitions for data clustering are proposed. According to [3]"formal definition (of cluster) is not only

difficult but may even be misplaced." But there have been some attempts to define the clustering term.

The authors in [4] tried to summarize these definitions and some of these definitions are as bellow:

- Cluster is a set of entities which are alike, and entities from different clusters are not alike.
- A cluster is an aggregate of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.
- Clusters may be described as continuous regions of this space (d-dimensional feature space) containing a relatively high density of points, separated from other such regions by regions containing a relatively low density of points.

clustering may be called with different names in different applications such as, unsupervised learning (in pattern recognition), numerical taxonomy (in biology, ecology), typology (in social sciences) and partition (in graph theory) [5] .

The algorithms used to cluster data are varying based on application and usage but they all can be divided into different categories. In [6] the clustering algorithm categories are as:

- **Partitional Clustering:** partitions the data into some groups and then tries to optimize a certain criteria such as mean of the clusters. K-means is one the most famous algorithms of this type. The most famous algorithms of this type are k-means, K-Mode, PAM, CLARA, FCM, and CLARANS.
- **Hierarchical Clustering:** Merges some clusters in order to make a bigger cluster or divides a cluster into some clusters to make smaller clusters. The most famous algorithms of this type are: BIRCH, CURE, and ROCK.
- **Density Based Clustering:** in these algorithms, we group the neighbors' based on their density in the area. The most famous algorithms of this type are: DBSCAN and DENCLUE.
- **Grid Based clustering:** the algorithms quantize the space into finite number of areas and then they perform their operations in each area separately. The most famous algorithms of this type are: STING and Wave-Cluster.

Our PSO based clustering algorithm is one kind of partitional algorithm. In the rest of this paper, when we use "clustering", we mean partitional clustering.

**Particle Swarm Optimization**

PSO is an efficient, simple, and effective global optimization algorithm that can solve discontinuous, multi-modal, and non-convex problems [7].This algorithm was first introduced in 1995 by a social-psychologist named Eberhart and electrical engineer Kennedy [8]. The PSO algorithm is a kind of social intelligence algorithms like ACO and Stochastic Search. These methods are inspired from behaviors of a group of animals or insects in order to reach a presumed goal like a shelter or food.

In this algorithm, a swarm containing N different particle that are the potential resolutions of our problem, considered to move in the solution space. Each particle has two movement characteristic, velocity and location. In each generation, these two characteristics are updated by some equations that are explained later. Alongside the movement characteristics, a particle has two behavioral characteristics named Global Best and Personal Best which are the best state of all of the particles in swarms in all duration of the algorithm and the best state of the particle in comparison to its own previous states, respectively. These two characteristics are also used in the movement characteristic in order to find better solutions for our problem. The philosophy of Global Best and Personal Best is that in nature, all of the members of a swarm have tendency to imitate the leader or literally, the strongest member of the swarm and also want to reach to their previous best conditions in the swarm in order to maintain their dignity in the swarm.N particles all move in a D-Dimensional solution space. Each particle considered to be a D-Dimension p=(p1,p2,…,pd) in the location of $X=(x_1,x_2,\ldots,x_d)$ that is moving in the solution space with velocity $V=(v_1,v_2,\ldots,v_d)$. The velocity is bounded by $V_{max}$= ($V_{max1}$, $V_{max2}$ … $V_{maxd}$) and if a particles has a velocity more than $V_{max}$, its velocity is replaced with $V_{max}$.

In figure 1, we see how particle moves in our problem space using velocity and location.

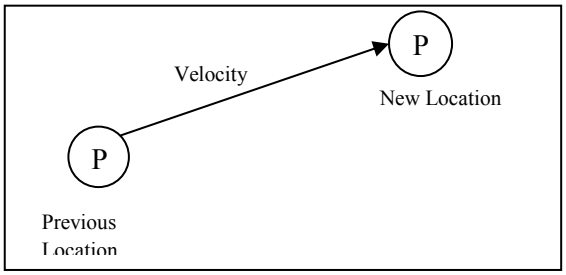


**Figure 1: Particle Update in Solution Space**

Equation 1 and Equation 2 are the velocity and location update functions. In each generation, all particles are updated using these equations until the maximum generation count reaches or a particle gets to our termination criteria. In all cases, the global best reached so far, is the solution of our problem.

$$v(i,j)_{t+1} = w * v(i,j)_i + c_1 * rand1(j)_t * \left(P(i,j)_{Best_t} - X(i,j)_t\right) + c_2 * rand2(j)_t * \left(G(j)_{Best_t} - X(i,j)_t\right) \quad \textbf{(1)}$$

$$X(i,j)_{t+1} = X(i,j)_t + v(i,j)_t \quad \textbf{(2)}$$

In Equation 1, w is inertia weight. A large w is good for global exploration and a small w is good for a local exploration in order to make our found solution more accurate. According to [9] it is better our w to be a number between in (0.9, 1.2) so that our exploration keep balance of global and local exploration. $V(i,j)_t$ is current velocity of particle I in dimension j in generation t. $X(i,j)_t$ is current location of particle I in dimension j in generation t. $P(i,j)_{Best}$ is Personal Best of Particle I in dimension j so far and $G(j)_{Best}$ is Global Best of all Particles in dimension j so far. $c_1$ and $c_2$ are called cognitive and social parameter respectively and are used to determine the impact of Global and Personal Best of current generation in the final solution. In … it is shown that these parameters are better to be $c_1=c_2=2$. But in recent it is suggested that it is better to have larger cognitive parameter ($C_1$) but we have to consider $c_1+c_2=4$. $Rand1(j)_t$ and $rand2(j)_t$ are a random numbers in U(0,1) for $j_{th}$ dimension of the particle. These random numbers are just for maintain the stochastic nature of the PSO algorithm.

In figure 2 we can see the exact movement of a particle in solution space according to Equation 1 and Equation2.

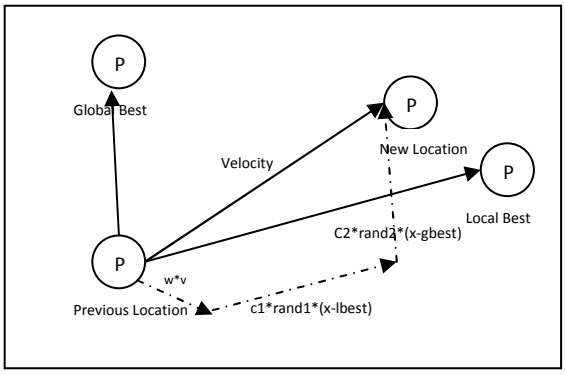


**Figure 2: Detailed Movement of Particle Update in Solution Space**

## Chaotic Theory

Chaos theory is a field of study mostly in mathematics, philosophy and physics. This theory studies the behavior of dynamic systems that are highly sensitive to their initial environments [10]. Butterfly effect is a public name of this theory. The chaos systems are systems that are completely dependent to their initial conditions and a tiny difference in their initial condition could cause a huge difference in outcome of these systems. Even though these systems were deterministic, there are not fully predictable. This behavior is called chaos.

We can all see chaotic systems all around ourselves in nature. The weather is the most common example of a chaotic system. Other examples of chaotic systems are economy and stock market.

The most important characteristics of a chaotic system are its randomness while being predictive.

**Chaotic Maps**

In mathematics, there are some equations that produce a number series with chaotic characteristics. The simplest one of these equations is one dimensional noninvertible maps. In this paper, we introduce three most famous chaotic maps.

1) **Logistic Map:** This Equation is the simplest chaotic map that was introduced by Sir Robert May in 1967 in[11]. Equation 3 shows Logistic map:

$$X_{n+1} = aX_n(1 - X_n) \qquad (3)$$

This map is ergodic in (0, 1) only if a is 4 but under the condition that $X_0 \in (0,1)$ and $X_0 \notin \{0.0,0.25,0.5,0.75,1.0\}$

2) **Tent Map:** This equation is very similar to Logistic Map. The definition of this Map is described in Equation 4.

a. $$X_{n+1} = \begin{cases} \frac{X_n}{0.5} & , \quad X_n < 0.5 \\ \frac{(1-X_n)}{0.5} & , otherwise \end{cases} \qquad (4)$$

3) **Improved Tent Map**: According to [12] if our tent Map is as in Equation 5, it will be better distributed in our solution area. So, the Improved Tent Map is as in Equation 5:

$$X_{n+1} = \begin{cases} \frac{X_n}{0.8} & , \quad X_n < 0.8 \\ \frac{(1-X_n)}{0.2} & , otherwise \end{cases} \qquad (5)$$

**PSO Based Clustering**

Among all the efforts that were tried for clustering of data with PSO algorithm, [1] has the best and simplest idea. In this approach in clustering, the cluster centroids of all clusters are in a particle and they all get updated and move toward the best cluster centroid that could be found until termination criteria was satisfied.

**Background**

The researches done with the PSO algorithm could be discussed in two categories:

- The Fuzzy clustering with Particle Swarm Optimization
- The Non-Fuzzy(Crisp) clustering with Particle Swarm Optimization

Furthermore, we will go to the depth of each category separately.

**The Fuzzy clustering with Particle Swarm Optimization**

In comparison with crisp data clustering, less research has been made on this category and this trend seems to have a potential for a more accurate observation. Fuzzy clustering is a kind of clustering that a point is not a member of just a cluster, each cluster owns this point but with different membership degrees.

The idea of using Particle Swarm Optimization Algorithm in fuzzy clustering was first proposed in [13]. In that paper, a new fitness function was suggested which results was better that the Fuzzy C-Means algorithm. In the next section of their paper, they corrected their own fitness function and got even better results.

Afterwards, Runkler and Ketz in [14] had done some researches on fuzzy clustering. They tried to use the C-Means fitness function in order to suggest a new fitness. Their model had better performance in comparison with Ant Colony and Stochastic Search Methods, in spite of being duller.

Wang [15] in 2007 managed to improve Fuzzy C-Means algorithm with application of Particle Swarm Optimization algorithm in association with Quantum theory. He used quantum error and clustering error which were used as fitness functions in next researches.

Jang in [16] used a specific type of Particle Swarm Optimization named Predator Prey. This algorithm uses the update equation of velocity and location twice. This model still used fuzzy c-mean as a base model and optimizes this algorithm.

Mei in [17] combines fuzzy C-Means Algorithm with Particle Swarm Optimization and proposed a new optimization algorithm. The results of this model showed that their suggested algorithm acted well in optimization of the regular and complex optimization functions. This

paper was not implemented in clustering and only used ordinary clustering techniques for optimizing algorithms.

Alizadeh in [18] tried to suggest a whole new idea using PSO algorithm. The algorithm used in his model was compared with global and local methods and got better results.

Izakian in [19] used fuzzy clustering with fuzzy PSO algorithm and used it as a new clustering method.

In Hsiang [20] uses Mahakanobis distance instead of Euclidean distance. This technique was more efficient in recognition of non-geometric shapes.

**Crisp Data Clustering using Particle Swarm Optimization**

As told before, Merwe and Englbrecht[1] showed the potential of clustering using Particle Swarm Optimization technique for the first time. In their model, they used a clustering error equation as fitness function which got better results and then, they used K-Means algorithm for the initialization in which they managed to optimize the K-Means results. Merwe's model was used as a foundation of many researches in the next years.

Ching in [21] in year 2004 also tried to use PSO algorithm in clustering using different ideas than Merwe but the results was not satisfying as the Merwe technique. Jiann [22] managed to use support vector learning idea in Particle Swarm optimization and applied his algorithm to dynamic clustering. Cohen [23] in 2006 used local PSO (LPSO) instead of ordinary Global PSO. The core idea was what Merwe used. The results showed progress in the end.

Sharma [24] considered the application of PSO algorithm in determining cluster borders. In his algorithm, he used the information taken from SOM technique, one of the most famous reduction techniques, and succeeded to find cluster borders resulting to more efficient clustering. In the same year, Kao [25] used PSO algorithm with a new fitness function.

Premalatha in [26] uses local search to improve the results. In this algorithm, after applying PSO algorithm, the local search is used for 20% of the final particles. The results of this technique were much better than previous algorithms. In [27] a hybrid technique was proposed. They combined the hierarchical clustering algorithms with merge techniques. Resembling clusters was merged together in each step and shaped bigger clusters. Merwe's fitness function is used in this model.

Esmin in [28] improved Merwe fitness function. He tried to analyze the fitness function and found the disadvantages, and then proposed a solution for each of these problems. Karthi in [29] used almost all of Eberhart and Shi's proposed algorithms in clustering and compared all the results. He used these algorithms in real world data and got the final results. Zhenkui in [30] combined K-Means and PSO algorithms and proposed a new algorithm based on both of them. He optimized the K-Means algorithm using his model.

Panchal in [31] combined the PSO algorithm with some of the standard clustering algorithms and compared all of them together. He used these algorithms in unsupervised clustering. Finally, he managed to use his algorithm in image clustering. Ghali in [32] used a mutation PSO for clustering. He used Merwe fitness function for his algorithm. Kiranyaz[33] applied PSO algorithm to dynamic data clustering. He used a multi dimension PSO algorithm for better clustering.

In [34] used adaptive PSO algorithm in clustering which resulted more efficient than the previous models. Johnson [35] also used improved converging PSO algorithms which improved the clustering results. Khan in [36] provided a survey of the clustering techniques used PSO algorithm. In [37] changed the inertia weight factor and used the reducing linear iteration inertia. He used a mutation based PSO in his algorithm. Naik et al [38], mixed PSO and K-means to introduce a novel method for extracting cluster centroids which lead to better performance. Furthermore Yu et al [22] exploited previous works in this field to introduce a specified clustering method for image segmentation.

**PSO Particles**

In [1] The design of the particles was as:

$P_i=(m_{11},m_{12},\ldots m_{dNc})$

Which in particle $P_i$, there is $N_c$ centroids with d Dimensions that $N_c$ is the number of clusters should be found in our dataset.

In this paper we didn't change this structure and our particles are the same as the original method.

**PSO Fitness Function**

In [1] ,the fitness function was as in Equation 6:

$$J_c = \frac{\sum_{j=1}^{N_c}\left[\sum_{\forall Z_p \in C_{ij}} \frac{d(z_p, m_j)}{|C_{ij}|}\right]}{N_c} \quad (6)$$

Which $Z_p$ shows the $p_{th}$ data vector, $|C_i|$ is the number of data vectors belonging to cluster i, and d is Euclidean distance between $Z_p$ and $m_j$.

Equation 6 fitness function shows good and approving results but the results was not good enough and also it was time consuming.

According to [28] the fitness function of Equation 6 has the following problems:

1. Sometimes a particle that does not represent a good solution is mistaken as it is a good cluster
2. This equation is not representing a reward for homogeneous solutions.

In order to solve the first problem, in [28] the Equation 6 fitness function was changed to Equation 7 in order to make better results.

$$\boldsymbol{F_1} = \left\{\sum_{\boldsymbol{j=1}}^{\boldsymbol{N_c}}\left[\left(\sum_{\forall \boldsymbol{Z_p} \in \boldsymbol{C_{ij}}} \frac{\boldsymbol{d(z_p, m_j)}}{\boldsymbol{|C_{ij}|}}\right) x \left(\frac{C_{ij}}{N_o}\right)\right]\right\} \quad (7)$$

And also Equation 5 is introduced based on Equation 8 in [28] to solve the second problem.

$$F_2 = F_1 x(|C_{ik}| - |C_{il}| + 1)$$

$$|C_{ik}| = \max_{\forall j=1,\ldots,N_c}\{|C_{ij}|\}\text{،}|C_{il}| = \min_{\forall j=1,\ldots,N_c}\{|C_{ij}|\} \quad (8)$$

The results using the fitness function of Equation 7 and 8 shows tremendous improvements. But still the time consuming problem was an issue. In this paper, we use a new fitness function instead of these functions.

**Fuzzy Fitness Function**

In this paper, we suggest using FCM fitness function instead of the fitness function proposed in [1] and updated in [28] . The FCM fitness function is as in Equation 9:

$$J_c = \sum_{j=1}^{N_o}\sum_{i=1}^{N_c} u_{ij}^q d^2(x_j, m_i) \quad (9)$$

Note that $N_o$ is the number of data vectors in our Data set, $N_c$ is the number of clusters, $u_{ij}$ is membership factor of data vector j to cluster i. $u_{ij}$ can be calculated by using equation 10 and $d^2(x_j, m_i)$ is Euclidean distance of data vector j and centroid of cluster i. q is a constant and must be more than 1, in this paper, we use q=2.

$$u_{ij} = \frac{1}{\sum_{k=1}^{N_c}\left(\frac{d(x_i, m_j)}{d(x_i, m_k)}\right)^{\frac{2}{q-1}}} \quad (10)$$

**PSO Algorithm Initialization**

The initialization of Particles in PSO Clustering was based on completely random factors. All the particles and all of their dimensions are initialized with completely random numbers and then they will move on solution space iteratively using Equation 7 and 8 to improve the solutions and find the correct answer.

**The PSO Clustering Algorithm**

The pseudo code for data clustering technique proposed in [1] is as follows in Figure 4:

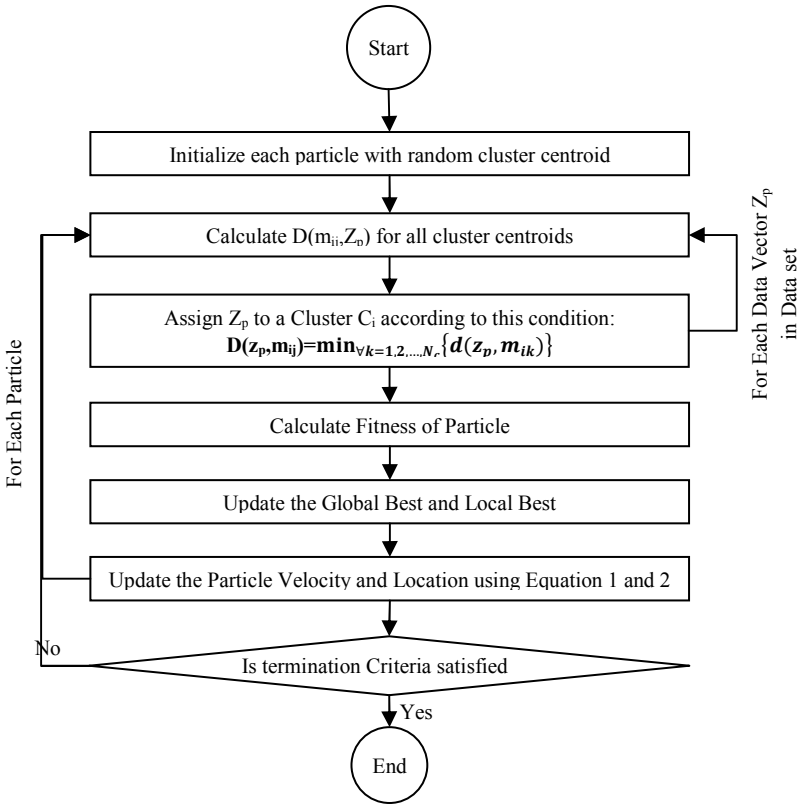


**Figure 3: Pseudo Code of PSO Clustering Algorithm**

## Our Proposed Method

Our proposed approaches in this paper are rather development of the algorithm first suggested in [1] which we call it the initial algorithm in the following. This development is done in various aspects as mentioned below:

1. Improvement of the Fitness Function of the initial algorithm
2. Improvement of the initial population production using the chaotic equations
3. Preventing the initial algorithm to be trapped in the local maxima and minima of the solution space using our suggested mutation.
4. the improvement of the solution and the convergence using variant inertia weights

In the following sections, we are going to explain each suggested method in more details.

## Optimization of the Fitness Function

In these papers, in order to evaluate the fitness of each cluster, they used the fitness function of the C-Means algorithm. The purpose of these papers was to optimize these algorithms. The C-Means fitness function is defined as follows:

Previously, the equation 10 has never been used in the crisp clustering algorithms. We have suggested to use this equation instead of the fitness function used in the initial algorithm in [39]. The reason is as follows:

- the faster computation of the Fitness using the Equation 10
- Equation 10 is more efficient because the C-Means Algorithm is proved to be an efficient method in clustering.
- According to the evaluations in [39] and the next section, this fitness is much faster and more efficient than the initial in action.

## Optimization of the Initial Population using the Chaotic Equations

As discussed in the previous section, chaotic theory is a physical theory which tries to model the chaotic movements without any pre-defined order. As in chaotic theory, the movement which seems to be simple

in the first glance grows extensively during time. This growth is predictable, although looks random. In other words, we can assume chaos as a series of semi random numbers that spread in the space in a predefined order.

Considering the above paragraph, we can use the chaotic equations as a good Successor of the random numbers.

As told in the introduction section, the PSO algorithms need to be initiated with some random numbers for the particles and their velocities. We can use chaotic numbers instead of these random initializations as well. We used the equations used and discussed in section 1.5 and mentioned in equations 3, 4 and 5.

In [7] a new approach for using chaos in PSO is proposed in their algorithm shows more efficiency than the regular PSO algorithms in action. One of the key suggestions in this paper was to use chaotic equations as a replacement for random initializations of the PSO and this caused the first generation to be more spread in the solution space, because of the nature of chaotic numbers. In this paper, we propose applying this approach to the clustering problem. The results of our new approach are discussed in the following sections.

Normally, the generation of the first numbers is done using the equation 11.

$$P_i = (U_i - L_i) \text{ x Rand} \qquad (11)$$

In equation11, $U_i$ and $L_i$ are the upper and lower bound for the $i^{th}$ dimension of the particle in the solution space, respectively. Instead of using the regular random generation approach, we produce a series of chaotic numbers using equation 3, 4 and 5. Considering this change, the equation 11 changes to equation 12 as bellow:

$$P_i = (U_i - L_i) \text{ x } X_{n+1} \qquad (12)$$

In the equation 12, Xn+1 is the next produced number calculated by the chaotic functions with the previous number. Note that Xn+1 is a number between 0 and 1 like the regular random function.

We have already published our results in [40] in which we discussed our method in more details. Our method caused the PSO algorithm to be much better fitness in the first generation of the algorithm. We will discuss the results in the next section. In figure 5, you can find the pseudo code of chaotic initialization.

**Avoiding Premature Convergence Suggested Mutation**

As discussed before, one of the most important shortcomings of the PSO algorithm comparing to the other algorithms of the same type (like Genetic Algorithm) is trapping in the local minima and maxima, especially in the functions in which there are plenty of local minima and maxima. From the beginning days of the PSO, there have been several attempts to overcome this disadvantage and different approaches have been proposed. One of the solutions for this problem is the usage of the mutation like Genetic Algorithm. First we consider a probability value for mutation, in case of fulfillment of all conditions; a random number is produced and replaced with the current number. In this case, we can throw one of our solutions to the new space; we can reduce the likelihood of being trapped in local minima and maxima, though it still exists to some extent.

In our proposed method in this paper, we focused on using the chaotic series instead of using regular random algorithms for mutation. So, our mutation is called "Chaotic Mutation" as well. In [41] proposed a chaotic based PSO which used the chaotic mutation and the outcome shows much better results comparing to the regular mutation ones.

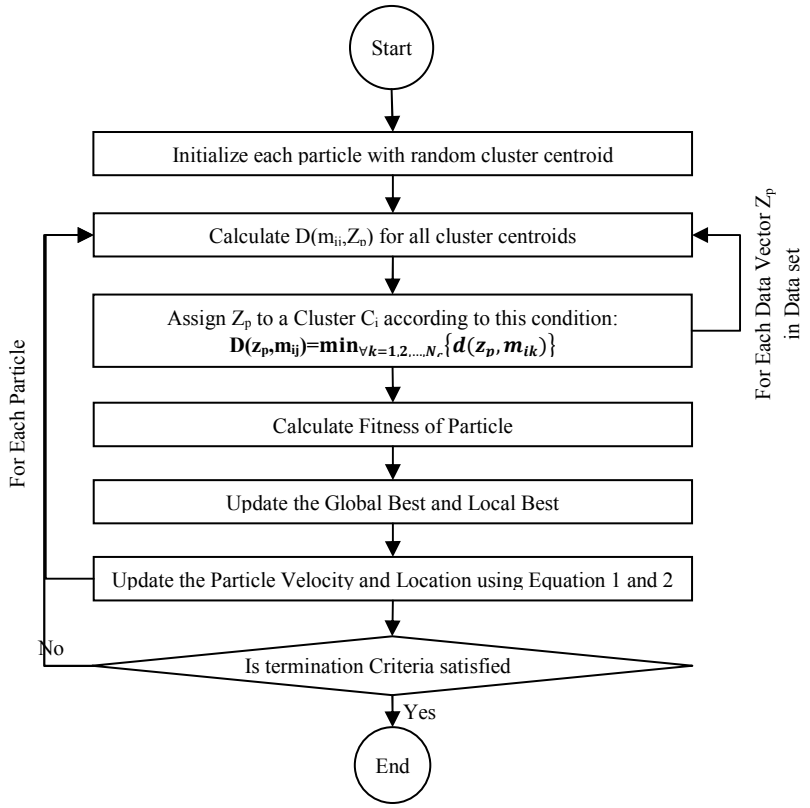


**Figure 4: Pseudo code of Chaotic Initialization**

We can also assume that using the chaotic mutations caused the reduction in number of generations required to converging of our algorithms to the solution. We will see this assumption in action the in the following section.

In [41]chaotic mutation using the Logistic Equation of equation 3 to find our solution. The equation used for chaotic mutation is as equation 13:

$$P_d(t)' = P_d(t) + \sigma_d(t)\gamma_d(t) \quad (13)$$

In the above equation 13, $\gamma_d(t)$ is the chaotic number produced by the Logistic equation 3 and $\sigma_d(t)$ is acquired with the equation 14 mentioned below:

$$\sigma_d(t) = \sigma_0 \exp(-at) \quad (14)$$

In equation 14, a is the constant value and $\sigma_0$ is a random number between 0 and 1.

The pseudo Code and Flowchart of this chaotic mutation is mentioned in figure 6 and 7.

```
1.      Select Number of Particles/5 of the top best Particles and Mark them as
Elite
2.      For the rest of Particles
2.1.    if rand<Mutation Factor
2.1.1.  p=chaosNumberProducer(p,'Logistic      ');
2.1.2.  q=q*exp(-1*t*4);
2.1.3.  Current_Particle =Current_Particle+p*q;
3.      Calculate fitness for all of the particles Again
```

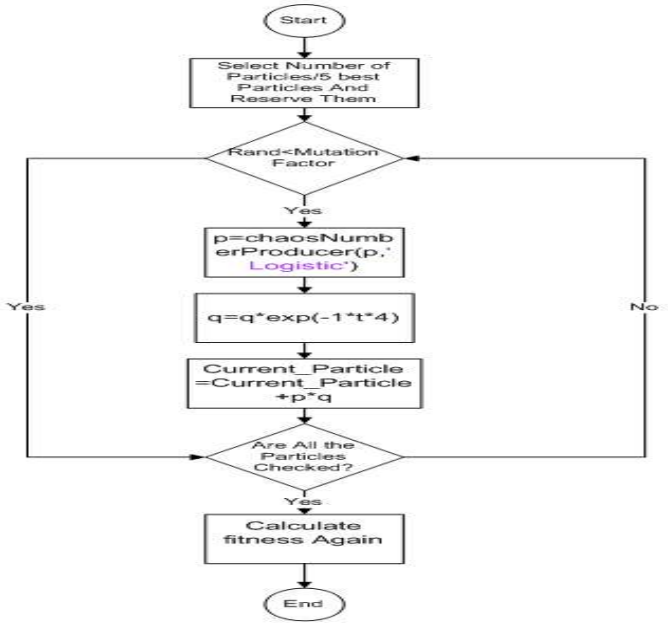


**Figure 5: Pseudo code of chaotic Mutation**

The results of this approach in the final solution in clustering problem are discussed in the next section in more details.

**Using Different types of inertia Weights**

In the optimization problems, like many other problems in the computer science, the researchers have to choose between velocity and power. For instance, a strong search in the solution space will find the best optimized answer but it is time consuming.

As we discussed before, one of the biggest problems of PSO algorithm is the immature convergence which caused the particles to be trapped in local minima and maxima. There have been a lot of efforts to overcome this problem and we mentioned two of them in the previous sections. One of the main solutions for this problem is using a coefficient called inertia weight for the movement of the particles. This solution s first proposed in [9] and gained attention soon after. This coefficient is something like acceleration in the Newtonian physical movement. The inertia weight changes the main equation for calculating the velocity as shown in equation 15.

$$V_{i+1} = w * V_i + C_1 * \text{rand} * (P_i - LB_i) + C_2 * \text{rand} * (P_i - GB_i) \quad (15)$$

In the equation 15, inertia weight is shown with w. The inertia weight should be designed in such way that if the particle is close to the final solution, the impact of the velocity reduces and the particle movement is more affected by global and personal bests of the particle (w should be small) and otherwise, if the particle is far from the solution, w increases to boost the impact of the velocity in the movement. We have to define a bound to fulfill our aim. In [42] the optimal values for w is defined as 0.4 and 0.9. In this paper, we used this values as the lower and upper bounds for our inertia weight, respectively.

As discussed before, in order to overcome the immature convergence of the PSO algorithm an increase the speed to reach the global solution, inertia weights are used an these weights must be a small value in case we are close to the global solution an otherwise, it has to be a large number. Different methods are proposed to fulfill this method. In the next sections, we describe some of these methods and will test them in action in the experimental results.

**Constant Inertia Weights**

In the first papers about this shortcoming which was done by Shi and Eberhart in [9] , they assume a constant value for inertia weight. With their experiments, they concluded that the inertia factor must be a value between 0.9 and 1.2 and the PSO algorithm which uses these values has better average efficiency than the other values.

**Fitness-Based Inertia Weights**

This method was first proposed in [43]. In this type of inertia weight, we consider fitness value of a particle as the core of our calculations. This method is sometimes called as "Adaptive Inertia Weight Factor (AIWF)". In this inertia, the factor is first initiate with the highest value and then, depending on their fitness value, the reduction in the lower bound of inertia linearly. Not that the upper and lower bounds for the inertia is defined by the user. Considering this method, each particle has its own inertia weight which will be calculated using the equation 16.

$$w = \begin{cases} w_{min} + \frac{(w_{max}-w_{min})(f-f_{min})}{f_{avg}-f_{min}} & f \le f_{avg} \\ w_{max} & f > f_{avg} \end{cases} \quad (16)$$

In the equation 16, $w_{max}$ and $w_{min}$ represent the minimum and maximum inertia values which are defined by the user. f is the current fitness of the particle and $f_{min}$ and $f_{avg}$ are the minimum and average of all the particles in the current generation.

This method is rather time consuming because of the following reasons:

1. Since it has to compute the average of the fitness
2. Each particle has to be calculated individually using the equation 16

In the next section, the experimental results for these methods are demonstrated ad discussed in details.

**Implementation and Evaluations**

In the previous section, we proposed a model for the clustering problem and described each part of the model in details with the special attention to the chaos theory in physics. In this section, we are going to put these methods in trial and find out there are efficient enough to be considered as a solution for the clustering problem or not.

We have selected 3 different datasets for the experiments. These datasets are standard ones which are obtained from [44]. The information for our datasets is demonstrated in table 1.

**Table 1: Datasets used for experiments**

| Name | Records | Properties | Clusters |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Wine | 178 | 13 | 3 |
| Glasses | 214 | 10 | 7 |

The specifications of the computer system used for our experiments are shown in table 2:

**Table 2: Hardware Specifications of the System we used for Experiments**

| CPU | Memory | OS | HDD |
|---|---|---|---|
| Intel Core 2 Due T 8300 2.54 GHz | 4.0 GB Dual-Channel DDR2 332MHz | MS Windows 7 Ultimate 64-bit | 488 GB Western Digital WDC |

We have implemented the methods using these programming languages in table 3:

**Table 3: Programming Languages for Experiments**

| Algorithm Implementation | GUI Implementation |
|---|---|
| Math works Matlab 7.8.0.347(R2009a) | Microsoft Visual Studio Team System 2008 Architecture Edition |

Each algorithm is repeated 30 times to find more accurate results. We use mean of these repeated experiments in our tables as the final results.

We divided our experiments in five phases respectively according to our proposed method plan in the previous section. To be me more precise, our phases of improvement is as follows:

- Optimization of the fitness Function
- Optimization of the Initialization Population with chaotic equations
- Simple Elitism
- the impact of the inertia weights

We have described each phase in the previous section. In this section, we define the implementation and compare each phase to the previous phases or other proposed PSO clustering algorithms in other papers.

We have evaluated our results with 11 different validity indices in three categories as follows:

- The Average Per Cluster Category
- The Overall Category
- The Implementation Category

In each category, we are going to explain some indices which we used to evaluate our algorithms.

**The Average Per Cluster Category**

This category calculates the indices about each cluster and then the average of these calculations is given as a final result of the evaluation.

We calculate these validity indices in this category:

1. **Average Compactness**: This indices is proposed in [45]and calculates the average compactness of the clusters. The best clusters demonstrated with the least value in average compactness.
2. **Intercluster Diameter**: The maximum distance of the two points in clusters
3. **IntraCluster Separation**: The maximum distance of cluster centroids
4. **Quantization Error**: The Average distance of the points. This function is the fitness first used by Merwe in [1]

**The Overall Category**

In this category, we evaluate our clusters in a general view. We used 4 indices to evaluate our clusters in general. These evaluation indices are as follows:

- **RMSSTD Index**: This factor is the Root-Square Standard Deviation (RMSSTD) in a dataset. In other words, these indices evaluate the homogeneity of the clusters produced by an algorithm. This factors needs to be the minimum in order to have an optimized answer.
- **RS Index**: This index shows the difference between the clusters. Less values means better clusters.
- **SDbw Index**: This index is about the compactness, Separation and interdistance and intradistance of the clusters. The minimum values of this index mean the optimum results.
- **SD Index**: This index is defined according to the average scattering in the clusters and their separation. The same as the other indices, the less this index is, the better the clusters are.

**The Implementation Category**

In this category we compare the phases from the implementation point of view. The factors we mention in this category are as follows:

- **Time Elapsed**: This index is the average time to elapse for the algorithms to finish with the result.
- **First Generation Fitness**: The Fitness value in the first iteration of the algorithm.
- **Last Generation Fitness**: The fitness value in the last iteration of the algorithm.

Each of categories is demonstrated in separate table for each dataset in each phase.

Afterwards, we are going to discuss our phases of implementation and experiments in more details.

**PSO Parameters**

In general, The PSO parameters for our experiment are as follows. Inertia Weigh is set to 0.9, Cognitive and Social Parameters are 2, Number of Particles and Generations are both 100.

We use these parameters as default values; they may change according to the nature of the experiment. Furthermore, we will change the inertia weights later but we will mention the change in the experiment description. You can find the parameters for our experiments in table 4.

**Table 4: PSO Specifications**

| Inertia Weight | Cognitive & Social Rate | Particles | Generation |
|---|---|---|---|
| 0.9 | 2 | 100 | 100 |

**The Implementation of Previously Proposed Algorithms**

The Comparison of the original fitness proposed in [1] and the improvements suggested in [28] shows a huge improvement in the results, based on our implementations of the methods. Please Note that because of the limitations, we just demonstrate the graph for the best Global in the 100 iterations testing the Iris dataset in figures 8 and 9.

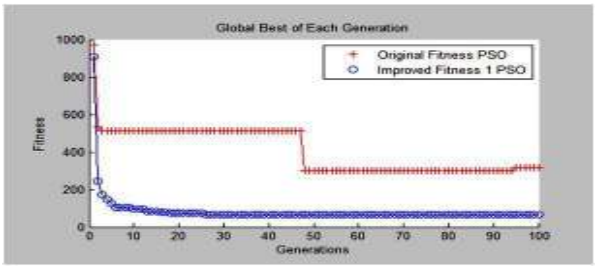


**Figure 5: Mean Fitness of 100 Generations for Iris Dataset**

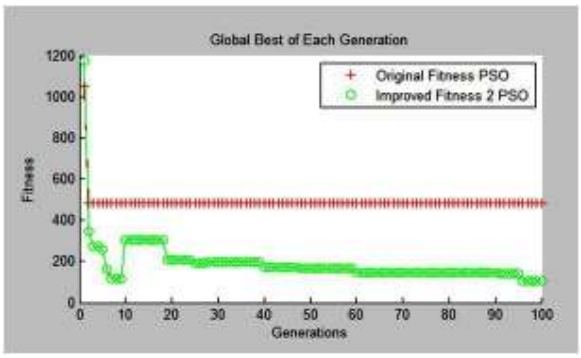


**Figure 6: Mean Fitness of 100 Generations for Iris Dataset**

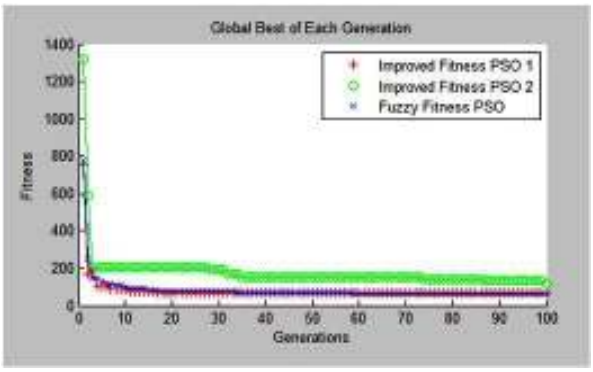


**Figure 7: Mean Fitness of 100 Generations for Iris Dataset**

The comparison of the performance of Improved Algorithm I and the original algorithm in three datasets of Iris, Wine and Glasses are as in Tables 5, 6 and 7.

Analyzing the results shows that the efficiency of the original fitness function is much worse in comparison with the improved ones. And also, the improved fitness 1 shows better efficiency in our experiments.

**Performance of our Proposed Fitness**

In the next step, we implemented our suggested fuzzy C-Means fitness function and compared it to the two improved fitness functions of the previous section. You can see that our proposed fitness function has better efficiency both in speed and performance.

The performance results for Iris, Wine and Glasses datasets are in tables 5, 6 and 7. You can find the average fitness value of the 100 generations of Iris dataset in figure 10.

After analyzing the results, it can be deducted that using Fuzzy C-means fitness function is a better choice to improve the clustering results. In the next step, we analyze the chaotic initialization of the PSO algorithm.

**Improving the Initialization using Chaotic Functions**

In this step, instead of randomly initialization of the particles in the first step, we used chaotic functions. You can find the output of this step in figures 11. The evaluation results are in tables 5, 6 and 7.

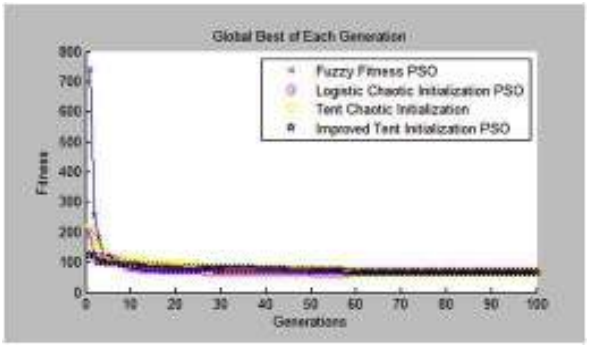


**Figure 8: Mean Fitness of 100 Generations for Iris Dataset**

Analyzing the performance results, we can find a noticeable change in the mean fitness value of the first generation but worse results in the last generation. We relate this problem to the pre-mature convergence of PSO which is one the main drawbacks of this algorithm. We can suppose that our algorithm is trapped in one of the local minimum and maxima as finally, this maximum is introduced as the outcome. We introduce two solutions to overcome this problem:

1) Like genetic algorithm, we can use mutation to randomly change one of the particles to a new spot in the solution space.
2) Using different methods of inertia weight in PSO

In the following sections we'll discuss the impact of each of these solutions.

**Using Mutation**

As discussed in section 3, one of the potential solutions to overcome the premature convergence is mutation. We described our way of mutating the Particles and the mean fitness of 100 generations in Iris dataset are shown in figure 12. You can find the evaluation results of this method in tables 5, 6 and 7.

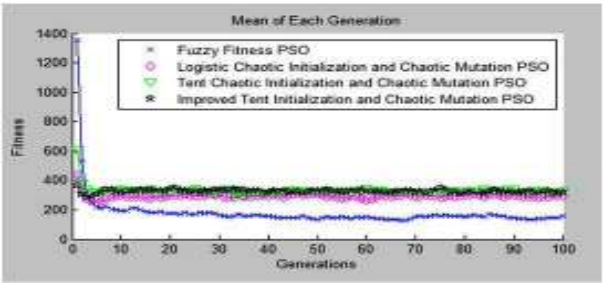


**Figure 9: Mean Fitness of 100 Generations for Iris Datase**

The improvements in final results are noticeable but we are trying to make that better using inertia weights in the next section.

### Using Different Fitness Bases Inertia Weights

So far, all the inertia weights we used were the constant value, 0.9 but in this section we measure the impact of different methods of inertia weight calculation on the output of our algorithm.

As described in section 3, the fitness based inertia weight is calculated for each particle individually considering the particle's fitness value. In our experiments, $w_{max}$ is 0.9 and $w_{min}$ is 0.4. In figure 13 you can find the mean fitness value of 100 generations of Iris dataset. You can find the final performance of this method in tables 5, 6 and 7.


**Figure 10: Mean Fitness of 100 Generations for Iris Datas**

**Table 1: Experimental Results for Iris Dataset**

| | Average Compactness | RMSSTD Index | RS Index | InterCluster Diameter | Intracluster Separation | S_Dbw Index | SD Index | Quantization Error | Time Elapsed | First Generation Fitness | Last Generation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Original Fitness** | 0.487801115 | 1.009305 | 0.13481 | 2.37565349 | 1.435702884 | 0.340707 | 1.749081 | 0.657400362 | 18.07589377 | 1007.226087 | 347.5279123 |
| **Improved Fitness 1** | 0.098155593 | 0.136969 | 0.882589 | 2.541173935 | 0.886169452 | 0.384138 | 0.986564 | 0.672323163 | 16.25228895 | 848.5668818 | 65.40915555 |
| **Improved Fitness 2** | 0.102598904 | 0.143578 | 0.876923 | 2.681839862 | 0.990520205 | 0.668878 | 1.251142 | 1.060757004 | 17.98046851 | 1305.738991 | 133.8676261 |
| **Fuzzy Fitness** | 0.097449 | 0.693966 | 0.096786 | 8.178978 | 0.135159 | 0.88414 | 0.375197 | 0.895028 | 5.310083 | 780.682 | 63.16279 |
| **Logistic Chaotic Initialization** | 0.097763 | 0.72085 | 0.088455 | 8.154 | 0.135945 | 0.883466 | 0.353736 | 0.889894 | 5.28463 | 190.733 | 64.57099 |
| **Tent Chaotic Initialization** | 0.097821 | 0.751889 | 0.09469 | 8.105214 | 0.135771 | 0.883615 | 0.355979 | 0.95817 | 5.31877 | 205.5774 | 65.5556 |
| **Improved Tent Chaotic Initialization** | 0.097821 | 0.726708 | 0.090086 | 8.224449 | 0.135577 | 0.883782 | 0.371217 | 0.950556 | 5.864117 | 171.6303 | 65.3231 |
| **Logistic Chaotic Initialization and Chaotic Mutation** | 0.103615 | 1.355397 | 0.066363 | 7.961879 | 0.145741 | 0.875069 | 0.573699 | 1.421839 | 8.672872 | 200.9418 | 85.40342 |
| **Tent Chaotic Initialization and Chaotic Mutation** | 0.108714 | 1.36138 | 0.06508 | 7.559361 | 0.155474 | 0.866726 | 0.62547 | 1.437635 | 9.049991 | 199.632 | 100.6933 |
| **Improved Tent Chaotic Initialization and Chaotic Mutation** | 0.111415 | 1.240809 | 0.061477 | 7.584114 | 0.160296 | 0.862592 | 0.656478 | 1.210202 | 8.438322 | 157.9184 | 103.7806 |
| **Logistic Chaotic Initialization and My Mutation Fitness Inertia** | 0.097455 | 0.708525 | 0.097438 | 8.157449 | 0.134996 | 0.88428 | 0.441458 | 0.895737 | 9.949707 | 201.578 | 63.00842 |
| **Tent Chaotic Initialization and My Mutation Fitness Inertia** | 0.097646 | 0.734431 | 0.091778 | 8.145782 | 0.13536 | 0.883967 | 0.379758 | 0.963951 | 9.836498 | 200.8427 | 63.674 |
| **Improved Tent Chaotic Initialization and My Mutation Fitness Inertia** | 0.098071 | 0.72548 | 0.089612 | 8.139588 | 0.136023 | 0.8834 | 0.388664 | 0.98166 | 9.940688 | 175.3747 | 64.20408 |

## Table 2 - Experimental Results for Glasses Dataset

| | Average Compactenss | RMSSTD Index | RS Index | Inter Cluster Diameter | Intracluster Seperation | S_Dbw Index | SD Index | Quantization Error | Time Elapsed | First Generation Fitness | Last Generation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Original Fitness** | 0.802567278 | 1118.302 | 0.856196 | 455.5644114 | 0.007094259 | 0.599769 | 0.166982 | 95.8776833 | 22.54962469 | 5050575.727 | 1975521.762 |
| **Improved Fitness 1** | 0.807729052 | 1059.866 | 0.86371 | 459.8207908 | 0.006306175 | 0.578644 | 0.16297 | 96.28718432 | 23.10193022 | 2449816.279 | 1916372.065 |
| **Improved Fitness 2** | 0.825189677 | 1305.522 | 0.832121 | 463.2379107 | 0.007668306 | 0.707079 | 0.215045 | 101.8701994 | 23.221082 | 3430082.563 | 2621152.289 |
| **Fuzzy Fitness** | 0.371313 | 0.675164 | 0.031077 | 7565.162 | 8.825639 | 0.977771 | 0.076162 | 0.067968 | 15.06768 | 259725.2 | 15339 |
| **Logistic Chaotic Initialization** | 0.36685 | 0.653913 | 0.030977 | 7562.698 | 8.832951 | 0.977753 | 0.11244 | 0.06568 | 15.3205 | 22817.84 | 15134.77 |
| **Tent Chaotic Initialization** | 0.368367 | 0.651773 | 0.030196 | 7552.309 | 8.948316 | 0.977462 | 0.172624 | 0.067494 | 14.2878 | 25636.72 | 14943.42 |
| **Improved Tent Chaotic Initialization** | 0.365591 | 0.678273 | 0.030552 | 7544.481 | 8.928507 | 0.977512 | 0.166174 | 0.067739 | 14.20467 | 20358.98 | 15228.82 |
| **Logistic Chaotic Initialization and Chaotic Mutation** | 0.374921 | 0.76006 | 0.028971 | 7527.244 | 9.271838 | 0.976647 | 0.192757 | 0.08239 | 19.00069 | 22552.83 | 16035.15 |
| **Tent Chaotic Initialization and Chaotic Mutation** | 0.375107 | 0.812286 | 0.028502 | 7529.675 | 9.399596 | 0.976325 | 0.182442 | 0.095346 | 19.15919 | 24723.67 | 16323.08 |
| **Improved Tent Chaotic Initialization and Chaotic Mutation** | 0.373269 | 0.780264 | 0.02928 | 7538.677 | 9.442049 | 0.976219 | 0.21127 | 0.089261 | 20.1676 | 20576.64 | 16500.48 |
| **Logistic Chaotic Initialization and My Mutation Fitness Inertia** | 0.365378 | 0.633084 | 0.031494 | 7572.288 | 8.756787 | 0.977944 | 0.174469 | 0.061222 | 32.50004 | 22878.34 | 14482.46 |
| **Tent Chaotic Initialization and My Mutation Fitness Inertia** | 0.366854 | 0.63793 | 0.030746 | 7574.631 | 8.822555 | 0.977779 | 0.222214 | 0.063857 | 27.77738 | 24435.31 | 14608.91 |
| **Improved Tent Chaotic Initialization and My Mutation Fitness Inertia** | 0.368372 | 0.659466 | 0.030865 | 7577.388 | 8.834127 | 0.97775 | 0.149226 | 0.064141 | 27.63489 | 20623.69 | 14876.52 |

## Table 3 - Experimental Results for Wine Dataset

| | Average Compactenss | RMSSTD Index | RS Index | InterCluster Diameter | IntraclusterSeperation | S_Dbw Index | SD Index | Quantization Error | Time Elapsed | First Generation Fitness | Last Generation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Original Fitness** | 0.384359844 | 16.26664 | 0.95903 | 29.859773099 | 0.328203376 | 0.209055 | 0.358218 | 9.115279962 | 42.882138896 | 296137.5703 | 25175.99927 |
| **Improved Fitness 1** | 0.368616109 | 9.034572 | 0.977245 | 29.775511311 | 0.046541199 | 0.111525 | 0.068984 | 9.309635399 | 45.214836881 | 270069.4523 | 15627.29185 |
| **Improved Fitness 2** | 0.371633006 | 8.6812614 | 0.978134725 | 29.683063933 | 0.064927221 | 0.107188996 | 0.092399663 | 10.24850958 | 44.277484645 | 272137.3355 | 17548.71191 |
| **Fuzzy Fitness** | 0.820566 | 0.532982 | 0.013235 | 176460.2 | 1047.964 | 0.865241 | 0.594841 | 0.155865 | 8.032419 | 2403368 | 1801045 |
| **Logistic Chaotic Initialization** | 0.821323 | 0.534041 | 0.013936 | 176376.8 | 1048.69 | 0.865147 | 0.598791 | 0.156165 | 8.164713 | 2247967 | 1802426 |
| **Tent Chaotic Initialization** | 0.823089 | 0.536204 | 0.014324 | 175841.2 | 1051.155 | 0.86483 | 0.619125 | 0.156285 | 8.173612 | 2467197 | 1803601 |
| **Improved Tent Chaotic Initialization** | 0.820888 | 0.535045 | 0.014922 | 176399.9 | 1048.799 | 0.865133 | 0.606911 | 0.156094 | 8.201249 | 2125770 | 1804748 |
| **Logistic Chaotic Initialization and Chaotic Mutation** | 0.820367 | 0.546271 | 0.014132 | 174103.7 | 1051.176 | 0.864828 | 0.634029 | 0.156366 | 12.29655 | 2252493 | 1827479 |
| **Tent Chaotic Initialization and Chaotic Mutation** | 0.821785 | 0.545883 | 0.016378 | 174695.5 | 1054.248 | 0.864433 | 0.673516 | 0.157573 | 12.46668 | 2306278 | 1831915 |
| **Improved Tent Chaotic Initialization and Chaotic Mutation** | 0.820653 | 0.545681 | 0.016378 | 175492 | 1050.885 | 0.864865 | 0.63469 | 0.157543 | 12.33695 | 2101529 | 1835906 |
| **Logistic Chaotic Initialization and My Mutation Fitness Inertia** | 0.821951 | 0.534321 | 0.012481 | 176199.6 | 1049.069 | 0.865099 | 0.597397 | 0.155847 | 17.44971 | 2239463 | 1801295 |
| **Tent Chaotic Initialization and My Mutation Fitness Inertia** | 0.81983 | 0.532155 | 0.013711 | 176897.2 | 1046.981 | 0.865367 | 0.592615 | 0.155635 | 16.84735 | 2411307 | 1801611 |
| **Improved Tent Chaotic Initialization and My Mutation Fitness Inertia** | 0.8218 | 0.536223 | 0.013523 | 175879.1 | 1049.294 | 0.86507 | 0.605835 | 0.156115 | 16.62631 | 2092900 | 1802649 |

**Future Works and Conclusion**

In this paper, we proposed an algorithm to improve the clustering with PSO algorithm with application of chaotic theory and mutation.

In our model, we showed that using Fuzzy C-means fitness function instead of the quantization Error as fitness can make the algorithm more efficient. Then, using chaotic equations instead of random initialization can make the first generation more diverse in the solution space but the problem of premature convergence had occurred. In order to overcome this problem we suggested mutation and fitness based inertia weight which increased the outcome efficiency.

Our suggestions to improve the method are as follow:

- Using more chaotic equations
- Improve the fitness with more novel functions
- Using different versions of PSO instead of the PSO we used in this paper
- Using improved FCM instead of simple FCM
- Hybridization of PSO with other Optimization algorithms

## ACKNOWLEDGMENTS

The authors would like to thank officials in Islamic Azad University, Quchan Branch for their financial support of the scientific project.

## REFERENCES

D.W. van der Merwe, and A.P. Engelbrecht, Data clustering using particle swarm optimization, Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, 2003, pp. 215-220 Vol.1.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, From data mining to knowledge discovery in databases. AI magazine 17 (1996) 37.

B.S. Everitt, Cluster Analysis, Heinemann Educational, the Social Science Research Council, London :, 1974.

B.S. Everitt, Cluster Analysis, Heinemann Educational, the Social Science Research Council, London :, 1980.

S. Theodoridis, and K. Koutroumbas, Pattern Recognition, First Edition, Academic Press, 1999.

M. Halkidi, Y. Batistakis, and M. Vazirgiannis, On Clustering Validation Techniques. J. Intell. Inf. Syst. 17 (2001) 107-145.

B. Alatas, E. Akin, and A.B. Ozer, Chaos embedded particle swarm optimization algorithms. Chaos, Solitons & Fractals 40 (2009) 1715-1734.

J. Kennedy, and R. Eberhart, Particle swarm optimization, Neural Networks, 1995. Proceedings., IEEE International Conference on, 1995, pp. 1942-1948 vol.4.

Y. Shi, and R. Eberhart, A modified particle swarm optimizer, Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, 1998, pp. 69-73.

K.T. Alligood, Chaos: An Introduction to Dynamical Systems, Springer-Verlag, 1997.

R. May, Simple mathematical models with very complicated dynamics. Nature 261 (1976) 459-467.

H. Yaoyao, Z. Jianzhong, L. Chaoshun, Y. Junjie, and L. Qingqing, A Precise Chaotic Particle Swarm Optimization Algorithm based on Improved Tent Map, Natural Computation, 2008. ICNC '08. Fourth International Conference on, 2008, pp. 569-573.

W. Li, L. Yushu, Z. Xinxin, and X. Yuanqing, Particle Swarm Optimization for Fuzzy c-Means Clustering, Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, 2006, pp. 6055-6058.

T.A. Runkler, and C. Katz, Fuzzy Clustering by Particle Swarm Optimization, Fuzzy Systems, 2006 IEEE International Conference on, 2006, pp. 601-608.

W. Hao, Y. Shiqin, X. Wenbo, and S. Jun, Scalability of Hybrid Fuzzy C-Means Algorithm Based on Quantum-Behaved PSO, Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on, 2007, pp. 261-265.

J. Woo-seok, K. Hwan-il, L. Byung-hee, K. Kab il, S. Dong-il, and K. Seung-chul, Optimized fuzzy clustering by predator prey particle swarm

optimization, Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, 2007, pp. 3232-3238.

C. Mei, and D. Zhou, An Improved Particle Swarm Optimization With Fuzzy c-Means Clustering Algorithm, Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC '09. International Conference on, 2009, pp. 118-122.

M. Alizadeh, E. Fotoohi, V. Roshanaei, and E. Safavieh, Clustering based fuzzy particle swarm optimization, Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American, 2009, pp. 1-6.

H. Izakian, A. Abraham, and V. Snasel, Fuzzy clustering using hybrid fuzzy c-means and fuzzy particle swarm optimization, Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, 2009, pp. 1690-1694.

L. Hsiang-Chuan, Y. Jeng-Ming, W. Der-Bang, and L. Shin-Wu, Fuzzy C-Mean Clustering Algorithms Based on Picard Iteration and Particle Swarm Optimization, Education Technology and Training, 2008. and 2008 International Workshop on Geoscience and Remote Sensing. ETT and GRS 2008. International Workshop on, 2008, pp. 838-842.

C. Ching-Yi, and Y. Fun, Particle swarm optimization algorithm and its application to clustering analysis, Networking, Sensing and Control, 2004 IEEE International Conference on, 2004, pp. 789-794 Vol.2.

J. Yu, A Novel Chaos PSO Clustering Algorithm for Texture Image Segmentation. Recent Advances in Computer Science and Information Engineering 5 (2012) 269-274.

S.C.M. Cohen, and L.N. de Castro, Data Clustering with Particle Swarms, Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, 2006, pp. 1792-1798.

A. Sharma, Determining cluster boundaries using particle swarm optimization, world academy of science engineering and technology, 2006, pp. 250-254.

I.W. Kao, C.Y. Tsai, and Y.C. Wang, An effective particle swarm optimization method for data clustering, Industrial Engineering and Engineering Management, 2007 IEEE International Conference on, 2007, pp. 548-552.

A.M.N. K. Premalatha, A New Approach for Data Clustering Based on PSO with Local Search. Computer and Information Science Vol 1, No 4 (2008).

S. Alam, G. Dobbie, and P. Riddle, An Evolutionary Particle Swarm Optimization algorithm for data clustering, Swarm Intelligence Symposium, 2008. SIS 2008. IEEE, 2008, pp. 1-6.

A.A.A. Esmin, D.L. Pereira, and F. de Araujo, Study of different approach to clustering data by using the Particle Swarm Optimization Algorithm, Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on, 2008, pp. 1817-1822.

R. Karthi, Comparative evaluation of Particle Swarm Optimization Algorithms using real world data sets, 2008.

P. Zhenkui, H. Xia, and H. Jinfeng, The Clustering Algorithm Based on Particle Swarm Optimization Algorithm, Intelligent Computation Technology and Automation (ICICTA), 2008 International Conference on, 2008, pp. 148-151.

M.H.K. Mr.V.K.Panchal, Ms. Jagdeep Kaur, Comparative Study of Particle Swarm Optimization based unsupervised clustering techniques. IJCSNS International Journal of Computer Science and Network Security 9 (2009) 132-140.

N.E.-D. Neveen I. Ghali, Mervat A. N., and Lamiaa Bakrawi, Exponential Particle Swarm Optimization Approach for Improving Data Clustering. International Journal of Electrical, Computer, and Systems Engineering 3 (2009) 208-212.

S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, Multi-dimensional Particle Swarm Optimization for dynamic clustering, EUROCON 2009, EUROCON '09. IEEE, 2009, pp. 1398-1405.

S.S. Madeiro, C.J.A. Bastos-Filho, F.B.L. Neto, and E.M.N. Figueiredo, Adaptative clustering Particle Swarm Optimization, Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, 2009, pp. 1-8.

R.K. Johnson, and F. Sahin, Particle swarm optimization methods for data clustering, Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009. ICSCCW 2009. Fifth International Conference on, 2009, pp. 1-6.

N.B. A Khan, S Bodkhe, N GHRCE, An Analysis of Particle Swarm Optimization with data clustering technique for optimziation in data mining. International Journal on Computer Science and Engineering 2 (2010) 1363-1366.

C.-J.H. Cheng-Hong Yang, Li-Yeh Chuang Accelerated Linearly Decreasing Weight Particle Swarm Optimization for Data Clustering, International Multiconference of engineers and computer scentists, Hong Kong, 2010.

B. Naik, S. Swetanisha, D.K. Behera, S. Mahapatra, and B.K. Padhi, Cooperative swarm based clustering algorithm based on PSO and k-means to find optimal cluster centroids, Computing and Communication Systems (NCCCS), 2012 National Conference on, IEEE, 2012, pp. 1-5.

E. Toreini, and M. Mehrnejad, Clustering data with Particle Swarm Optimization using a new fitness, Data Mining and Optimization (DMO), 2011 3rd Conference on, IEEE, 2011, pp. 266-270.

E. Toreini, and M. Mehrnejad, A novel method in fuzzy data clustering based on chaotic PSO, Internet Technology and Secured Transactions (ICITST), 2011 International Conference for, IEEE, 2011, pp. 335-340.

Y. Min, H. Huixian, and X. Guizhi, A Novel Dynamic Particle Swarm Optimization Algorithm Based on Chaotic Mutation, Knowledge Discovery and Data Mining, 2009. WKDD 2009. Second International Workshop on, 2009, pp. 656-659.

Y. Shi, and R.C. Eberhart, Parameter Selection in Particle Swarm Optimization, Proceedings of the 7th International Conference on Evolutionary Programming VII, Springer-Verlag, 1998, pp. 591-600.

Y. Shi, and R.C. Eberhart, Empirical study of particle swarm optimization, Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, 1999, pp. 1950 Vol. 3.

M.L. Repository.

M. Za, and H. Messatfa, A comparative study of clustering methods. Future Gener. Comput. Syst. 13 (1997) 149-159.