A new Algorithm Based on Improved Artificial Fish Swarm Algorithm for Data Clustering

D. Yazdani[1] and B. Saman[2] and A. Sepas-Moghaddam[3] and F. Mohammad-Kazemi[4] and M.R. Meybodi[5]

[1]Young Researchers Club and Elites,
Mashhad Branch, Islamic Azad University, Mashhad, Iran
d.yazdani@miau.ac.ir

[2]Shirvan Branch, Islamic Azad University, Shirvan, Iran
barat.saman@shirvaniau.ac.ir

[3]3Department of Computer Engineering and Information Technology,
Qazvin Branch, Islamic Azad University, Qazvin, Iran,
sepasmoghaddam@qiau.ac.ir

[4]4Department of Computer Science, Payame Noor University
Mashhad, Iran
fmkazemi@pnu.ac.ir

[5]Department of Computer Engineering and Information Technology
Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

ABSTRACT

Artificial Fish Swarm Algorithm (AFSA) is one of the state-of-the-art swarm intelligence approaches that is widely used for optimization purposes. On the other hand, data clustering is an unsupervised classification technique which has been addressed by researchers in many disciplines and in many contexts. The contribution toward this study is twofold. First, weak points of standard AFSA including lack of using previous experiences of AFs during optimization process, lack of existing balance between exploration and exploitation and high computational load were investigated in order to present a New Artificial Fish Swarm Algorithm (NAFSA). For resolving the weak points, functional behaviors and the overall procedure of AFSA have been improved. In addition, some parameters are eliminated and several supplementary parameters are added. Subsequently, a hybrid clustering algorithm was proposed based on NAFSA and k-means approaches. This combination leads to maximum utilization of the involved approaches for data clustering. The proposed methods were evaluated on several datasets and its efficiency was compared with that of several state-of-art algorithms in this domain. Results showed high efficiency of the proposed algorithm.

Keywords: Artificial Fish Swarm Algorithm, Data Clustering, K-means, Swarm Intelligence, Optimization.

Mathematics Subject Classification Number: 68T01.

1 Introduction

Artificial fish swarm algorithm (AFSA) is amongst state-of-the-art swarm intelligence approach, which was proposed by Li $X_i$ao Lei in 2002 (Lei, Shao and Qian, 2002). AFSA is inspired by the collective movement of the fish and their various social behaviors in nature. Framework of AFSA is based on population, random search, and behaviors. AFSA has been used for optimization purposes e.g. data mining (Zhang, Shao, Li and Sun, 2006), image segmentation (XiaoLi, Ying, JunTao and JiQing, 2010), neural network learning (Tsai and Lin, 2011) (Shen, Guo, Wu and Wu, 2011), color quantization (Yazdani, Nabizadeh, Mohamadzadeh Kosari and Nadjaran Toosi, 2011), global optimization (Rocha, Martins and Fernandes, 2011) (Yazdani, Nadjaran Toosi and Meybodi, 2011), data clustering (Jiang and Zhu, 2011) (Cheng, Jiang and Yuan, 2009) , multi-objective optimization (Jiang and Zhu, 2011), PID controller parameters optimization (Cheng and Hong, 2012) etc.

Nevertheless, AFSA has not been extensively considered by researchers, due to its high complexity in comparison with other algorithms in this domain, particularly PSO, whereas the results of AFSA are not better than those of PSO. The reasons for such inefficiency are high structural complexity of the algorithm, high computational complexity, lack of appropriate balance between exploration and exploitation, and lack of using previous experience of the swarms individuals in order to improve optimization process. These issues led us to improve the structure of AFSA in order to conquer the weaknesses. In this paper, we aim to propose a novel AFSA algorithm, so called NAFSA, to resolve the weaknesses of the standard AFSA. For this purpose, different stages of AFSA are modified in order to remove the weak points and consequently, improve efficiency of the algorithm. The modifications include reducing structural complexity as well as computational complexity of the algorithm, determining a balance between exploration and exploitation, and utilizing previous experience of the swarms individuals in order to improve optimization process.

On the other hand, Clustering is an unsupervised classification technique in which datasets that are generally represented as vectors in multi dimensional space, based on a similarity criterion, are divided into some clusters. Where predefined number of clusters is K and there are N m-dimensional data points, clustering algorithm assign each data point to the clusters, regarding the similarity between data points. Data clustering which is categorized amongst NP problems has been widely used in data classification (Memarsadeghi and O'Leary, 2003), data compression (Emre Celebi, 2011), data mining (Pizzuti and Talia, 2003), pattern recognition (Wong and Li, 2008), machine learning (Yang, Song and Zhang, 2006) , image segmentation (Vannoorenberghe and Flouzat, 2006) and so forth. The importance of data clustering in various sciences led to introducing various methods for data clustering (Hartigan, 1975).

The term "k-means" was first used by James MacQueen in 1967 (MacQueen et al., 1967), although the primary idea was proposed by Hugo Steinhaus in 1956 (Steinhaus, 1956). k-means clustering algorithm is one of the foremost and straightforward clustering approaches

that has been widely used in different applications. K-means method starts with K cluster centers and divides a set of objects into K subsets. This is one of the most well-known and functional clustering techniques, since it can be implemented in linear time. However, there are several significant weak points in k-means method. One of these weaknesses is extensive sensitivity of the algorithm to initial values of cluster centers. Furthermore, there are several local optimums in objective function of k-means, so that the method could not guarantee to pass local optimums. Therefore, by selecting inappropriate initial position of cluster centers in the problem space, the algorithm converges to a local optimum.

In the second contribution of this paper, we propose a hybrid approach based on k-means and NAFSA, called K-NAFSA, in order to improve the convergence rate by conquering the extreme sensitivity of k-means to the chosen initial solution. In the proposed approach, k-means has been applied as a behavior for artificial fishes in NAFSA. This combination leads to maximum utilization of the involved approaches for performing data clustering. The experiments have been conducted on five artificial datasets and seven real datasets including datasets of Iris, CMC, WDBC, Sonar, Glass, Wine and Pima using four evaluation criteria including Sum of Intra Cluster Distances, Offline SICD, Error rate and Average number of fitness evaluations. Efficiency of the proposed method has been compared with that of standard AFSA, k-means, PSO and KPSO.

The rest of this paper is organized as follows: Section 2 is dedicated to a review AFSA. The principles of the proposed algorithms are discussed in Section 3. Experimental results and analysis are presented in Section 4. Section 5 concludes the paper.

2 ARTIFICIAL FISH SWARM ALGORITHM

2.1 ARTIFICIAL FISH SWARM PROCEDURE

AFSA is one of the swarm intelligence methods and evolutionary optimization techniques. The framework of this algorithm is based on functions that are inspired from social behaviors of fish swarm in the nature. In water world, fish can find areas that have more foods, which is done with individual or swarm search by fishes. According to this characteristic, artificial fish (AF) model is represented by prey, free move, swarm, and follow behaviors. AFs search the problem space by those behaviors. Food consistency degree in water area is AFSA objective function. Finally, AFs reach to a point which its food consistency degree is maxima (global optimum).

As it is observed in Fig. 1, AF perceives external concepts with sense of sight. Current position of AF i shown by vector $X = (x_1, x_2, ..., x_n)$. The $visual$ is equal to sight field of AF and X_v is a position in $visual$ where the AF intends to move. Then if X_v has better food consistency than current position of AF, it goes one $step$ toward X_v which causes change in AF position from X to x_{next}, but if the current position of AF i better than X_v, it continues searching in its $visual$ area. Food consistency in position X is fitness value of this position and

is presented with $f(X)$. The $step$ is equal to maximum length of the movement. The distance between two AFs which are in X_i and X_j positions is shown by $Dis_{ij} = ||X_i - X_j||$ (Euclidean Distance).

AF model consists of two parts of variables and functions. Variables include X (current AF position), $step$ (maximum length step), $visual$ (sight field), try_number (the maximum test interactions and tries), $bulletin$ and $crowd\ factor\ \delta\ (0 < \delta < 1)$. Also functions consist of prey behavior, free move behavior, swarm behavior and follow behavior. In each $step$ of the optimization process, AF looks for locations with better fitness values in problem search space by performing these four behaviors based on the algorithm procedure. Standard AFSA pseudo code is shown in Algorithm 1. In the following subsection, the behaviors of AFSA will be discussed. More details could be found in (Lei et al., 2002).

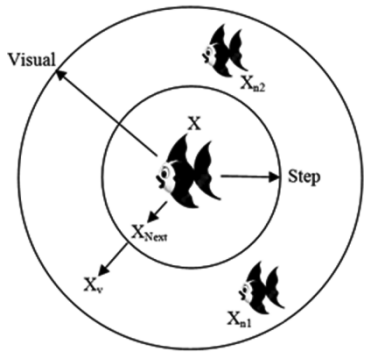

Figure 1: Artificial Fishes and the Environment.

2.1.1 Free move behavior

In nature, when fish is not successful in finding food, it moves freely. In AFSA, when an AF could not move toward a place with more food, moves a random $step$ in problem space by Eq. (2.1):

$$X_{i,d}(t+1) = X_{i,d}(t) + Step \times Rand_d(-1,1) \tag{2.1}$$

Where, $X_{i,d}$ is component d of AF i's position in D-dimensional space and $1 \leq$d$\leq D$. Rand function generates a random number with uniform distribution in the range of $[-1,1]$.

2.1.2 Prey behavior

In nature, every fish constantly looks for food or places with more food. If X_i is the current position of AF i, we choose position X_j in $visual$ of AF i randomly. $f(X)$ is the food consistency in position X or in other word its fitness value. Position X_j is calculated by Eq. (2.2):

$$X_{j,d}(t+1) = X_{i,d}(t) + Visual \times Rand_d(-1,1) \tag{2.2}$$

Algorithm 1: Standard AFSA

begin
  for *each AF* i **do**
    initialize x_i
  end
  $bulletin$ = arg min f(x_i) **repeat**
    for *each AF* i **do**
      Perform $SwarmBehaviour$ on x_i(t) and compute $x_{i,Swarm}$
      Perform $FollowBehaviour$ on x_i(t) and compute $x_{i,Follow}$
      if *f*($x_{i,Swarm}$) > *f*($x_{i,Follow}$) **then**
        $x_i(t+1)$ = $x_{i,Follow}$
      else
        $x_i(t+1)$ = $x_{i,Swarm}$
      end
    end
    if *f*($x_{Best-AF}$) <= *f*($bulletin$) **then**
      $bulletin$ = $x_{Best-AF}$
    end
  until *stopping criterion is met*
end

Afterward food density in X_i is compared with current positions, if $f(X_i) \geq f(X_j)$, AF i moves forward one $step$ from its current position to X_j, which is performed by Eq. (2.3):

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \frac{\vec{X}_j - \vec{X}_i(t)}{Dis_{i,j}} \times Step \times Rand(0,1) \tag{2.3}$$

X_i is a D-dimensional vector and $Dis_{i,j}$ is Euclidean distance between vectors X_i and X_j . $X_j - X_i$ generates a transfer vector from X_i to X_j and when divided by $Dis_{i,j}$, a vector with unit length is created from X_i toward X_j. Rand function generates a random number which causes AF i moves as much as a random percent of $Step$ toward position X_j. Nevertheless, if $f(X_i) < f(X_j)$, we choose another position X_i by Eq. (2.2) and evaluate its food density to understand whether forward condition is satisfied or not. If after try_number times AF does not succeed in satisfying forward condition, the concerned AF performs free move behavior and moves a $step$ in problem space randomly.

2.1.3 Swarm behavior

One of the properties of fish as a swarm is that they always try to move beside other swarms members. It causes fish swarm does not scatter and the generality of swarm is kept. In AFSA, in order to keep swarms generality, in each of the iterations, AFs try to move toward a central

position. A central position of swarm is given by Eq. (2.4):

$$X_{Center,d} = \frac{1}{N} \sum_{i=1}^{N} X_{i,d} \tag{2.4}$$

As it could be seen in Eq. (2.4), component d of X_{center} vector is the arithmetic average of all swarm AFs′component d. Let N be the population size, nc be the number of AF in $Visual$ field around of X_{center}. If $f(X_{center}) \leq f(X_i)$ and $\delta > (nc/N)$, central position has better food consistency than current position and population density in its neighborhood is not much, so that AF i moves toward central position by Eq. (2.5):

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \frac{\vec{X}_{Center} - \vec{X}_i(t)}{Dis_{i,Center}} \times Step \times Rand(0,1) \tag{2.5}$$

If $n_c = 0$ or the condition of moving toward central position is not satisfied, prey behavior is performed for AF i.

2.1.4 Follow behavior

In the process of fish swarm moving, when a fish or a number of fishes find food, neighbor fishes follow them to reach food. If X_i is the current position of AF i, it checks neighbor X_n, if nn is the number of AFs in the $Visual$ of AF n, if $f(X_n) \leq f(X_i)$ and $\delta > (nn/N)$, i.e. position X_n has a better food consistency than the current position of AF i and population density in its neighborhood is not much, therefore, AF i moves one $step$ toward AF by Eq. (2.6):

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \frac{\vec{X}_n - \vec{X}_i(t)}{Dis_{i,n}} \times Step \times Rand(0,1) \tag{2.6}$$

If AF i has no neighbors or none of its neighbors satisfy following condition, prey behavior would be performed for AF i.

2.2 COMPARISON BETWEEN AFSA AND PSO

Since both PSO and AFSA algorithms are amongst swarm intelligence algorithms, there is a population of individuals in them that compose together a swarm. Members of these swarms cooperate with each other to reach a common objective and their interactions with each other lead to creating a general behavior among them. Both PSO and AFSA algorithms work based on random search and probability rules. In fact, particles or AFs are randomly distributed in problem space and after that also their movements are completely dependent on numbers that are generated randomly. It may be presented that these algorithms are similar or even AFSA is one of the PSO versions, but the instruction of these two algorithms and the type of movements of swarm members in both algorithm have fundamental differences. Unlike AFSA, various versions of PSO have been presented until now, but in all of them, the framework of PSO has been kept and has not been changed.

According to various studies we have done on PSO and AFSA algorithms, the most prominent difference between these approaches is the difference in movements of particles and AFs. In PSO, particles are completely dependent to their past experience and to the swarms past experience for the next movement. In PSO algorithm, in order to determine particles next position, every particle needs (a) the best position that has been experienced so far, (b) the best position that all members of swarm have experienced so far, (c) the previous velocity that was the displacement vector of its previous motion and (d) its current place and it does not need the current position of other swarm particles. However, in AFSA, AFs do not use of their previous experiences and other swarm members′and also their previous movements for determining their next position. Indeed, according to Eqs. (2.1)-(2.6), for determining the next position, AFs depend on their current position and other swarm members.

2.3 WEAK POINTS OF STANDARD AFSA

2.3.1 Lack of using previous experiences of AFs during optimization process

In standard AFSA, the best found position is recorded on $bulletin$. The reason for such recording is to save current positions, since even an AF that has found the best position so far, can lose its position in the next iteration by performing free move behavior and be involved in a worse position. Consequently, AFs are not able to perform an acceptable exploitation around current best found position in order to find better positions. Therefore, exploitation performance of AFs decreases. AFs in AFSA do not use the best recorded position on the $bulletin$ for their next movements. This is a considerable weak point in AFSA, since the best result found by swarm so far is not applied for improving swarm member positions. Therefore, convergence rate of AFSA is decreased.

2.3.2 Lack of existing balance between exploration and exploitation

In AFSA, AFs search the problem space by performing AFSAs behaviors in their $Visual$ space. Also, AFs move toward their goal based on random percentage of $Step$ in each iteration. Setting the initial value for $visual$ and $step$ parameters has a considerable effect on the quality of the final result, because the values of these two parameters remain fixed and are equal to their initial value up to the end of the algorithm execution. If we consider large initial values for these two parameters, AFs can move faster toward goal, since they can search larger space around themselves and move with larger $step$ length in each iteration. In this condition, a large value of $Visual$ parameter leads to increasing the search boundary of AFs to search further distances that causes searching spaces out of local optimums in which they are trapped. As a result, they could leave the local optimums.

However, some disadvantages are involves by using large values of $Visual$ and $Step$ parameters. In this case, exploitation ability of AFs decreases. In fact, the algorithm appropriately performs exploration. However, by approaching the goal, AFs are unable to perform an acceptable exploitation, since $Visual$ parameters value is larger than the space where AF should search. Therefore, probability of finding positions with better fitness decreases. Also, because

of large Step, AFs may pass global optimum after approaching it which leads to decreasing accuracy and stability of algorithm.

If we consider small values for these two parameters, the algorithm can perform acceptable exploitation. But in this case, AFs move slowly toward goal and their ability to pass local optimums decreases and exploration ability of the algorithm is degraded. Therefore, by adjusting the initial values of $Visual$ and $Step$ parameters AFs are able to appropriately perform just one of the exploration and exploitation tasks.

2.3.3 Wasting high computational load

In AFSA optimization process swarm and follow behaviors are independently performed for each AF in each iteration. Subsequently, new position of an AF i determined with respect to one of these two behaviors at the end of each iteration, which leads to better result. However, considerable computational cost is consumed for executing both swarm and follow behaviors, where only one of them contributes to determine new position of AFs. Therefore, consumed computational load for performing one of these behaviors has no effect on the movement of AF and it is wasted. One of the most important criteria for comparing optimization algorithms is the number of fitness evaluations during optimization process. As a result, other optimization algorithms outperform AFSA, in term of this criterion.

High computational complexity is involved in AFSA. All neighbors of AFs must be considered for executing follow behavior. For this reason, Euclidean distances between all AFs are computed, so that a huge computational load occurs in each iteration. Also, each AF performs two fitness evaluations for follow and swarm behaviors in a single iteration in the best case and this number is $2 \times (try\_number + 1)$, in the worst case.

Finally, there are several parameters, functions and conditions in AFSA which makes its structure complex. Consequently, implementation of AFSA is difficult.

2.3.4 Weak points of AFSAs parameters

In AFSA, AFs in each of their movements change maximum as long as their step. Value of $step$ is equal or less than $Visual$ value. In prey and follow behaviors, AFs move toward the position corresponded to their $Visual$ which is calculated regarding a random percentage of $step$ parameter by Eq. (2.3) and Eq. (2.6). However, probably AFs moves to an undesirable position. For example, suppose Euclidean distance between destination position and current position of AF to be 2 and parameter value of $step$ to be 10. Random generated number by Rand function in Eq. (2.3) and Eq. (2.6) is however larger than 2, which means corresponding AF gets further from goal position.

$Crowd\ factor$ parameter controls swarm diversity. Smaller values of this parameter cause increasing the distance between AFs and bigger values of it leads to decreasing distance between them. It is worth to note that distance between AFs or swarm density considerably influence exploration and exploitation abilities. Although adjusting exploration and exploitation significantly depend on determining $Visual$ and $step$ parameters value, $crowd\ factor$ value is also affect them. Totally, specifying appropriate initial values of $crowd\ factor$, $Visual$ and $Step$ parameters is especially difficult for various applications and final result is highly dependent on the vales of these parameters. According to these conditions, solving problems required both high exploration and exploitation abilities. Thus, it is very difficult or even impossible to conquer the problems by utilizing standard AFSA.

In AFSA, $Visual$ and $step$ are two scalar parameters. There are numerous applications for which AFSA needs AFs with large dimensions. If we consider the same interval range for all dimensions of search space, a specific amount for $Visual$ and $Step$ parameters could be considered. Nevertheless, in most real applications e.g. data clustering, this issue could not be considered, due to this fact that the interval ranges are not the same. For example, suppose dimension i of all data vectors in the range of [1,2] and dimension j for these data vectors in the range of [40,100]. In this situation, if we consider small values for $Visual$ and $Step$ parameters which are appropriate in dimension i, search process encounter complicatedness in dimension j, since the small values lead to considerable degradation in convergence rate in dimension j. On the other hand, if we consider large values for $Visual$ and $Step$ parameters which are appropriate for dimension j, search process encounter inefficiency in dimension i, since these values are excessively large for searching in dimension i which leads to searching in a space out of interval range of the dimension. Therefore, it is impossible to determine a specific value for $Visual$ and $Step$ parameters values.

2.4 AFSA CONFIGURATION FOR DATA CLUSTERING

In order to configure AFSA for data clustering, first, a fitness function for data clustering must be determined on which AFSA performs optimization process. In this paper, in order to find optimal values of cluster centers, Eq. (2.7) is used. In fact, Eq. (2.7) is the fitness function which must be optimized by AFs. Eq. (2.7) calculates sum of intra cluster distances (SICD), which was used as the similarity criterion between the members of a cluster. It is worth mentioning that the best cluster is the one whose sum of intra cluster distances is minimum. Therefore, AFSA minimizes the fitness function presented by Eq. (2.7):

$$J(C_1, C_2, ..., C_k) = \sum_{i=1}^{k} (\sum_{X_j \in C_i} ||Z_i - X_j||) \tag{2.7}$$

In Eq. (2.7), sum of the Euclidean distance between each data vector in a cluster and center of the cluster is calculated. To be more elaborate, there are K clusters $(C_i(1 \leq \text{i} \leq K))$ and N data vectors $(X_j(1 \leq \text{j} \leq N))$ which must be clustered regarding distance from each cluster centers $(Z_i(1 \leq \text{i} \leq K))$. Euclidean distance of data vectors corresponded to a cluster from the

center of cluster is less than the distance from other clusters centers. Thus, AFSAs objective is to determine cluster centers in order to minimize Eq. (2.7). Since data vectors and cluster center vectors are d-dimensional and there are K clusters, each AF should represent K cluster centers in d-dimensional space, consequently it has $K \times d$ components in its vector. Fig. 2, shows a vector of an AF that contains K d-dimensional cluster centers. It should be noted that particles vector in PSO is also the same as Fig. 2.

$$\left(Z_{1,1}, Z_{1,2}, ..., Z_{1,d}, Z_{2,1}, Z_{2,2}, ..., Z_{2,d}, ..., Z_{K,1}, Z_{K,2}, ..., Z_{K,d}\right)$$

Figure 2: Structure of an AF Position in Data Clustering Problem Space.

Therefore, in AFSA, first we randomly set initial values of AFs of the swarm in problem space, so that each of AFs involve K random initial cluster centers which would be improved by performing AFSAs behaviors in problem space.

3 PROPOSED ALGORITHM

3.1 NEW ARTIFICIAL FISH SWARM ALGORITHM (NAFSA)

In this section, New Artificial Fish Swarm Algorithm (NAFSA) is presented, in which we try to resolve the weak points of standard AFSA. In NAFSA, $step$ and $crowd\ factor$ parameters are eliminated and a new parameter called $contractionfactor$ is added. In addition, modifications are applied on some of other parameters. Prey and free move behaviors are combined with some changes in a specific parameter, so called individual behavior, and there is another behavior called swarm behavior which performs instead of follow and swarm behaviors. In the following, NAFSA is discussed in details.

Suppose that there are N AFs as a swarm in D-dimensional space. Position of AF i is represented by $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,D})$. In NAFSA, AFs have a $Visual$ vector that is represented as $(v_1, v_2, ..., v_D)$. Each of the dimensions of $Visual$ vector is determined with respect to the interval range of the dimension of search space. Therefore, in a different manner of standard AFSA, NAFSA can properly specify $Visual$ value in every dimension for problems involving different interval range in their various dimensions. $Visual$ is the environment of an AF surrounding in D-dimensional space where the AF searches which could be changed in each step. When we consider $Visual$ parameter value in all dimensions with regard to a large search space, both AFs′search space and their movement $step$ become larger. So, convergence rate to global optimum, avoidance capability of being trapped in local optimums and consequently exploration ability of swarm are improved. But as it was discussed, their exploitation ability decreases. On the other hand, if we consider $Visual$ parameter value in its all dimensions with regard to a small search space, exploitation ability increases and exploration ability is deteriorated. In fact, there is no balance between exploration and exploitation.

To remove this problem, first we have to consider large amounts for $visual$ vector components. Therefore, AFs are able to perform an exploration with high performance. $Visual$ vector components value should decrease simultaneously with moving toward global optimum to decrease gradually the exploration ability and increase exploitation by approaching to global optimum. As a result, the algorithm is able to explorer the global optimum range with more accuracy. For this purpose, a parameter called $contractionfactor$ (CF) is added to NAFSA structure. CF is a positive number less than 1 that could be considered constant or be generated by a function. Previously, $inertia\ weight$ was presented for balancing the exploration and exploitation in PSO (Shi and Eberhart, 1998). CF in NAFSA has a similar application to $inertia\ weight$ in PSO. Previously, some approaches for modifying the $Visual$ and $Step$ parameters were presented (He, Belacel, Hamam and Bouslimani, 2009) (Tian and Tian, 2009). However, the approaches are completely different from the presented method in this paper. Here, we used a random function to generate CF in each iterations that is represented by Eq. (3.1). Previously, Eberhart and shi presented random $inertia\ weight$ for PSO in (Shi and Eberhart, 1998).

$$CF = CF_{min} + (CF_{max} - CF_{min}) \times Rand(0,1) \tag{3.1}$$

Eq. (3.1) generates CF as a random number in the range of [CF_{min} , CF_{max}]. Thereby, in each iteration, the value of component i for $visual$ vector is calculated by Eq. (3.2):

$$Visual_d(t+1) = Visual_d(t) \times CF \tag{3.2}$$

Note that $Visual$ vector value is the same for all AFs. At the beginning of algorithm execution, AFs′ positions are initialized randomly in problem space. Then, every AF executes a swarm behavior and an individual behavior.

3.1.1 Individual behavior

This behavior is composed of prey and free move behaviors. In this case, AF i which is in position $X_i(t)$ tries a number of times to move toward better positions. In each try, AF considers position $X_j(t)$ around itself by Eq. (2.2), then evaluates food consistency (fitness value) of that point and if $f(X_i) \geq f(X_j)$, next positions of AF i are obtained by Eq. (3.3).

$$\vec{X_i}(t+1) = \vec{X_j} \tag{3.3}$$

Since position $X_j(t)$ is in $Visual$ range of AF i, the distance which the AF i moves in each dimension by means of Eq.(3.3) would be less than or equal to the value of $Visual$ vector on the same dimension. If $f(X_i) \geq f(X_j)$, AF i displaces by Eq. (3.3) and tries to move toward better positions from its new positions by Eq. (2.2) and Eq. (3.3) until this process is performed for a number of times. But, if $f(X_i) < f(X_j)$, AF does not move toward X_j and tries again from its previous position to find a better position. So, in a single execution of individual behavior,

an AF can move up to try a number of steps toward better positions. If an AF i unsuccessful in all attempts for finding a better position and moving toward that, it moves toward a random position in its $visual$ field that for AF i is done by Eq. (3.4).

$$X_{i,d}(t+1) = X_{i,d}(t) + Visual \times Rand_d(-1,1) \tag{3.4}$$

In NAFSA, every AF tries a number of times to move toward better positions by performing individual behavior, but when an AF fails in all attempts, by performing Eq. (3.4), moves toward a random position in its $visual$ field and loses its previous position. Performing this movement could result in obtaining a worse position of AF in problem space. However, executing of this movement on AFs is necessary, since it causes diversity in swarm and AF could search other positions in later iterations.

In NAFSA, it tries to maintain the diversity of swarm by performing Eq. (3.4) on AFs, but it would not be applied for the best AF in the swarm. Therefore, the best AF does not lose its position, even when it could not find a better position in its neighborhood. So, best AF displaces just when it could find a better position in its Visual. In this condition, the best place that has been found in the previous iterations by swarm is the position of the best AF of the swarm. Because in each iteration, the best AF of swarm displaces only when moves toward a better position. In NAFSA, there is no more need to $bulletin$, since constantly the best found position by swarm so far is the position of the best AF of swarm.

3.1.2 Group behavior

In group behavior, two goals are followed, first goal is to keep AFs as a swarm and the second one is the movement of AFs toward the best AFs position. In group behavior, first, central position of swarm is calculated by Eq. (2.4). Movement condition toward center position, i.e. $f(X_{Center}) \leq f(X_i)$ is controlled for AF i and if it is satisfied, next position of AF i is obtained by Eq. (3.5):

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \frac{\vec{X}_{Center} - \vec{X}_i(t)}{Dis_{i,Center}} \times [\vec{Visual} \times \vec{Rand}(0,1)] \tag{3.5}$$

In Eq. (3.5) $Visual$ is a vector and Rand function generates a D-dimensional vector of separated random numbers. It is worth to note that in order to multiply two vectors, the corresponding components are multiplied. If $f(X_{Center}) > f(X_i)$, AF i cannot move toward central position and instead of that by Eq. (3.6) moves one $step$ toward the best AFs position of swarm, i.e. X_{Best}:

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \frac{\vec{X}_{Best} - \vec{X}_i(t)}{Dis_{i,best}} \times [\vec{Visual} \times \vec{Rand}(0,1)] \tag{3.6}$$

Therefore, each AF that is in a worse position, compared to central position, moves one $step$ toward the central position and if its position is better than central position of swarm, it moves one $step$ after the best AF of the swarm. Thus, for maintaining the swarm, all AFs move toward a better position by performing group behavior. Thereby, in NAFSA, the best found position by swarm is used for the movement of other AFs that causes the increment of the convergence rate of the swarm. In group behavior, first, AFs try to move toward central position, since swarm diversity decreases fast and it is possible to AFs are trapped in local optimums, in case of moving all AFs toward the best AF in every iteration. Therefore, group behavior mechanism is designed in order to move AFs as a swarm in each iteration which avoids fast diversity decrement.

In group behavior, central position of swarm may have better fitness value than the best AF of swarm. performing Eq. (3.5) on an AF causes moving toward central position, but worse positions may exist along the way between AF position and central position. Thereafter, applying Eq. (3.5) for the best AF can make its position worse and then possibly cause losing the best found position. Consequently, if the central position fitness is better than the best AF, next position of the best AF i determined by Eq. (3.7).

$$\vec{X}_{Best}(t+1) = \vec{X}_{Center} \tag{3.7}$$

Eq. (3.7) is only considered for the best AF with condition $f(X_{Center}) < f(X_{Best})$. Other AFs, after considering move condition toward central position use Eq. (3.5). Using Eq. (3.7) for moving toward central position causes locating all AFs on the same point of search space which means completely losing diversity of the swarm.

3.1.3 NAFSA procedure

In NAFSA, individual and group behaviors are done for every AF in each iteration. In a similar manner of standard AFSA in which executing one of the swarm behavior and follow behavior had no effect on the movement of AFs and thus, considerable amount of computational cost were wasted, both individual and group behaviors are effective in moving AFs in NAFSA. In NAFSA, first, all AFs execute individual behavior and base on the outcome move to new positions. Subsequently, all of them with respect to their new position perform group behavior. Hence,, each AFs performs an individual search based on local information in every stage of the optimization process and performs a group movement based on global information of the swarm. All computations of NAFSA are used for the movement of AFs toward goal, so that computational load is not wasted. NAFSA pseudo code is presented in Algorithm 2.

In NAFSA, there is no preference for performing individual or group behaviors, since, these two behaviors are performed alternately on AFs during the execution iterations of NAFSA.

3.2 HYBRID ALGORITHM FOR CLUSTERING: K-NAFSA

In this section, a hybrid clustering algorithm, so called K-NAFSA, was proposed based on NAFSA and k-means approaches. Several advantages including particularly high computational speed and efficient local search are involved in K-means. However, its convergence is extremely sensitive to the chosen initial solution. The goal of the hybrid approach is to improve convergence rate, by conquering the mentioned problem. In the proposed approach, k-means has been applied as a behavior for AFs. In fact, after executing group and individual behaviors on AFs, we perform an iteration of k-means on a specified percentage of AFs.

Algorithm 2: NAFSA

```
1  begin
2      for each AF i do
3          initialize x_i
4      end
5      bulletin = arg min f(x_i) repeat
6          for each Artificial Fish i do
7              Perform IndividualBehaviour
8          end
9          for each Artificial Fish i do
10             Perform GroupBehaviour
11         end
12         Update Visual by equation (3.2)
13     until stopping criterion is met
14 end
```

Suppose AF i performed individual and group behaviors and placed in position X_i. Afterwards, it wants to perform an iteration of k-means. Position X_i includes K clusters centers for d-dimensional data. First, distances of data vectors from K clusters centers corresponded to X_i are obtained by Eq. (3.8) and each of the data vectors is allocated to a cluster which its Euclidean Distance from its center is smaller than the other clusters center. Therefore, cluster centers composing vector X_i are updated by Eq. (3.9) that cause changing the AF i's position to a new position. Indeed, one stage of k-means algorithm is performed on AF i. K-NAFSA pseudo code is shown in Algorithm 3.

$$Dis(X_p, Z_j) = \sqrt{\sum_{i=1}^{d}(X_{pi} - Z_{ji})} \tag{3.8}$$

where, X_p is data vector, Z_j is the center of cluster j and d is the number of dimensions of data vectors and cluster center vectors.

$$Z_j = \frac{1}{n_j} \sum_{\forall X_p \in C_j} X_p \tag{3.9}$$

where, n_j is the number of data vectors that are assigned to cluster j and C_j is a subset of all data vectors corresponded to cluster j. The obtained cluster center using Eq. (3.9) is the average vector of the assigned data vectors to the cluster. Steps Eq. (3.8) and Eq. (3.9) are reiterate until the completion criterion is satisfied. It should be noted that k-means algorithm would be performed only on some limited AFs which are randomly selected in each iteration for avoiding precocious convergence. Therefore, K-NAFSA combines advantages of NAFSA and k-means algorithms while avoiding their weak points in data clustering problems. The experimental results show that K-NAFSA is not sensitive to the chosen initial solution and high convergence rate, high ability in passing local optimums, and high accuracy are involved in this approach.

Algorithm 3: K-NAFSA

```
1  begin
2      for each AF i do
3          initialize x_i
4      end
5      bulletin = arg min f(x_i) repeat
6          for each Artificial Fish i do
7              Perform IndividualBehaviour
8          end
9          for each Artificial Fish i do
10             Perform GroupBehaviour
11         end
12         for some Artificial Fish i do
13             Perform one iteration of k-means
14         end
15         Update Visual by equation (3.2)
16     until stopping criterion is met
17 end
```

4 EXPERIMENTS

Experiments are performed on 12 datasets and efficiency of standard k-means, PSO, hybridized algorithm of PSO and k-means called KPSO (Van der Merwe and Engelbrecht, 2003), standard AFSA, NAFSA and K-NAFSA were compared on these datasets. In all of the methods, the objective function is Eq. (2.7), which calculates SICD. In the following of this section,

first, used datasets will be discussed. Afterwards, configuration and parameters settings of different algorithms will be described and finally the experimental results will be studied.

4.1 DATASETS

In this paper 12 datasets are used amongst which seven datasets consist standard real dataset of UCI (Frank and Asuncion, 2010) including datasets of Iris, CMC, WDBC, Sonar, Glass, Wine and Pima and 5 datasets are generated artificially. Brief specifications of datasets including name, size, number of attributes, the number of classes and the number of available data in every class are presented in Table 1 and comprehensive characteristics of the artificial datasets are discussed subsequently.

Table 1: Characteristics of the Used Data Sets

Name	No. of classes	No. of attribute	Size (size of classes in parentheses)
Art1	4	2	400(100,100,100,100)
Art2	4	2	600(150,150,150,150)
Art3	7	2	350(50,50,50,50,50,50,50)
Art4	5	3	250(50,50,50,50,50)
Art5	7	10	140(20,20,20,20,20,20,20)
Iris	3	4	150(50,50,50)
Pima	2	8	768(500,268)
Wine	3	13	178(48,71,59)
Glass	6	9	214(70,17,76,13,9,29)
CMC	3	9	1473(629,334,510)
Sonar	2	60	208(111,97)
WDBC	2	30	569(357,212)

In this paper, it is tried to select datasets for experiments to consist of different characteristics of the problem space e.g. dimension of samples, variation of the types of samples′attributes, existence of boundary samples, existence of common samples, number of samples, arrangement and distribution of samples, ranges′length of various dimensions of samples, number of classes and size of classes.

1. Artificial dataset 1 (Art1): in this artificial dataset, there are four classes and each of the classes contains 100 samples involving two attributes. Samples′order of this dataset in 2-dimensional space is done randomly with uniform distribution. The first and the second attributes of the first class′ data are in the range of [0,10]. First attribute of the second class′ data is in the range of [10,20] and their second attribute is in range of [0,10]. The first attribute of the third class′ data is in the range of [0,10] and their second attribute is in the range of [10,20]. The first and the second attributes of the fourth class′ data are in the range of [10,20]. Samples of this dataset are shown in Fig. 3-a. As it could be observed in this dataset, there are common boundary samples.
2. Artificial dataset 2 (Art2): in this artificial dataset, there are four classes and every class includes 150 samples involving two attributes. The order of the samples of this dataset in 2-dimensional space is done randomly with uniform distribution. The first and the second attributes of the first class′ data are in the range of [0.8,9.2]. The first attribute of the second class′ data is in the range of [10.8,19.2] and the second one is in the range of [0.8,9.2]. The first attribute of the third class′ data is in the range of [0.8,9.2] and the second one is in the

range of [10.8,19.2]. For the fourth class′ data, the first and the second attributes are in the range of [10.8,19.2]. The samples of this dataset are shown in Fig. 3-b. Data vectors′order in this dataset is such that there are no boundary data.
3. Artificial dataset 3 (Art3): in this artificial dataset, there are 7 classes and every class has 50 samples invloving 2 attributes. The distributions of different attributes of every class′samples are: Class1 $\sim$ Uniform(25,40), Class2 $\sim$ Uniform(40,55), Class3 $\sim$ Uniform(55,70), Class4 $\sim$ Uniform(70,85), Class5 $\sim$ Uniform(85,100), Class6 $\sim$ Uniform(100,115), Class7 $\sim$ Uniform (115,130). This dataset is represented in Fig. 3-c.
4. Artificial dataset 4 (Art4): in this dataset, there are 5 classes and every class has 50 samples involving 3 attributes. The distributions of different attributes of every class are: Class1 $\sim$ Uniform(25,40), Class2 $\sim$ Uniform(40,55), Class3 $\sim$ Uniform(55,70),Class4 $\sim$ Uniform(70,85), Class5 $\sim$ Uniform(85,100). This dataset is represented in Fig. 3-d.
5. Artificial dataset 5: in this dataset, there are 7 classes and each of them contains 20 samples involving 10 attributes. The distributions of various attributes of every class are: Class1 $\sim$ Uniform(10,20), Class2 $\sim$ Uniform(20,30), Class3 $\sim$ Uniform(30,40), Class4 $\sim$ Uniform(40,50), Class5 $\sim$ Uniform(50,60), Class6 $\sim$ Uniform(60,70), Class7 $\sim$ Uniform(70,80).

4.2 PARAMETER SETTINGS

For clustering process of each dataset, clusters centers for AFs in standard AFSA, NAFSA, K-NAFSA, for particles in PSO and KPSO, and also for the centers in k-means are randomly initialized in the range of data values in each of dimensions, belongs to the data vectors in dataset. Thus, generating invalid clusters centers is prevented. For standard AFSA, NAFSA, K-NAFSA, PSO and KPSO, the problems dimension (D) for each of datasets is calculated by multiplying the number of clusters (K) and the number of data vectors dimension (d) in the corresponding dataset. The number of iterations for all methods is considered $10 \times D$ (Kao, Zahara and Kao, 2008). In PSO and KPSO, the configuration of all parameters is performed based on (Kao et al., 2008), in which c_1 and c_2 are considered 2, $inertia\ weight$ is calculated by $[w = 0.5 + rand/2]$ in every iteration and population size is supposed to be $5 \times D$ for all datasets. In KPSO, first, k-means is executed until no changes are observed in intra-cluster distance for two consecutive iterations. After that, cluster centers obtained by k-means are matched to a particle in the initial solution of PSO and other particles are randomly initiated. For standard AFSA, the number of AFs (population size) is equal to $4 \times D$ and for NAFSA and K-NAFSA is $2 \times D$. In the light of diverse experiments, increasing the number of AFs from the mentioned values in standard AFSA, NAFSA and K-NAFSA, not only does not affect the efficiency of algorithms, but also leads to increasing computational load.

Parameter value of try_number has considerable effect on search ability for AFs. If we assume low value for this parameter, search ability for AFs decreases significantly and the algorithm cannot achieve suitable results. If this value is assumed high, the probability of finding better positions increases. However, by extreme increments of this value, no improvement is reached in the results and it leads to increasing the number of fitness evaluation and computational load. Thereby, we should determine an appropriate threshold for try_number parameter.

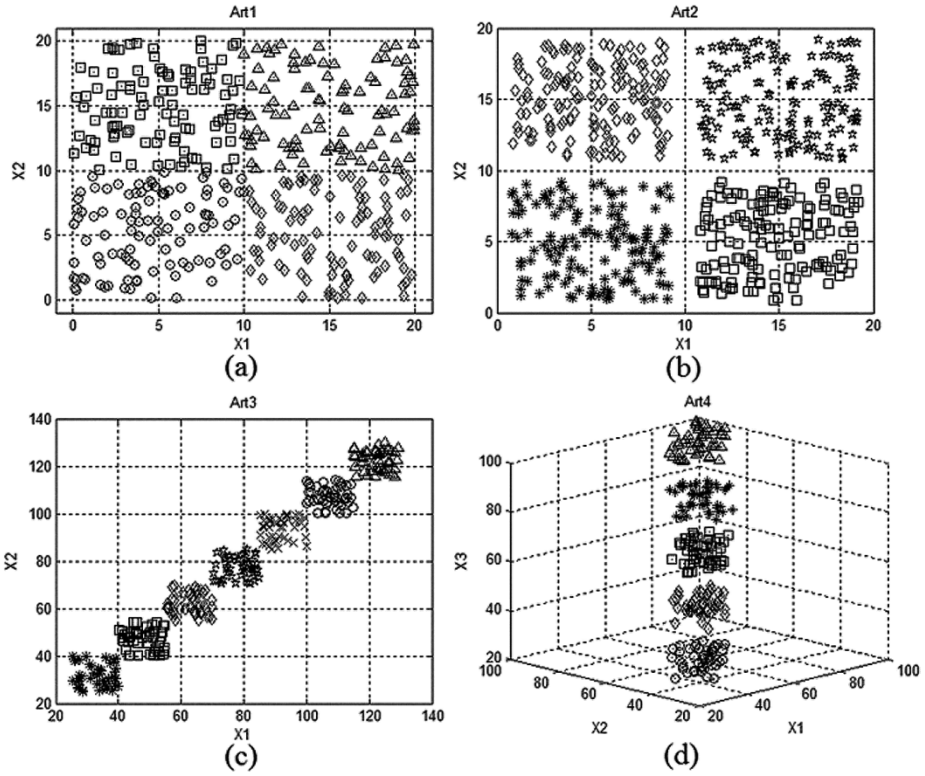

Figure 3: Representation of a)Artificial Dataset1 b)Artificial Dataset2 c)Artificial Dataset3 d)Artificial Dataset4

On the basis of several experiments done on 12 datasets in Table 1, we concluded that the appropriate range of try_number for standard AFSA, NAFSA and K-NAFSA is $5 \leq try\_number \leq 8$. In this paper, for the simplicity of the comparisons, try_number value is assumed 8 for all datasets (for standard AFSA, NAFSA and K-NAFSA). According to the discussions of previous sections concerning standard AFSA, obtained results of standard AFSA is very sensitive to initial values of $Visual$ and Step, so that the value of these two parameters must be determined in order to establish a trade-off between exploration and exploitation. Since the range of search space can be different in each of the problem dimensions in data clustering problem, therefore, determining an appropriate value for $Visual$ and $Step$ parameters is a complex process or even impossible. In this paper, the value of $Visual$ parameter is equal to 10 percent of the range of changes average in all dimensions of data vectors. $Visual$ value is calculated by Eq. (4.1):

$$Visual = 0.1 \times \frac{1}{d} \times \sum_{i=1}^{d} (Max(x_i) - Min(x_i)) \tag{4.1}$$

Where, d is the number of dimensions of data vectors in dataset and X_i is the set of component ith of all data vectors. Based on the done experiments, specification of $Visual$ value using Eq. (4.1) can provide the best trade-off between exploration and exploitation in all dimensions of search space in standard AFSA, however, it still is not adequate. In case of existing considerable difference between the length and range of the dimensions in the datasets, an

appropriate value for $Visual$ could not be determined. $Step$ parameter value is assumed less or equal to Visual. In this paper, $Step$ value is 70 percent of $Visual$ value. $Crowdfactor$ value is assumed 0.5 in standard AFSA for establishing a balance between swarm diversity and exploitation. $Visual$ value for NAFSA and K-NAFSA is presented as a vector, whose components' number is equal to the dimensions of search space. As it was mentioned before, the number of dimension of search space for clustering a set of data is calculated by multiplying number of clusters and that of data vectors attributes in the dataset. For determining the values of $Visual$ vector components, first a vector is obtained by Eq. (4.2):

$$V_i = P \times (Max(x_i) - Min(x_i)) \tag{4.2}$$

where, X_i is the set of component i^{th} of all data vectors and the number of vector Vs components is equal to $K \times d$. $Visual$ vector is calculated by merging K vectors, represented by V, where K is the number of clusters in dataset. P value determines the value of $Visual$ in a specific dimension, regarding the percentage of the length of search space range, corresponded to the components values in the same dimension. In NAFSA and K-NAFSA, the initial value of $Visual$ should be considered large for providing an appropriate exploitation at the beginning of the algorithms execution. Then, we decrease its vector values during the execution by Eq. (3.2) to increase the exploitation ability. According to the experimental results, if we assume P value to be 0.2 in Eq. (4.2) and CF_{max} and CF_{min} to be 1 and 0.9, respectively, a balance between exploration and exploitation is established in NAFSA and K-NAFSA which leads to acceptable results. In K-NAFSA, k-means is executed on AFs which are selected randomly in each iteration. If the number of AFs on which we implement k-means algorithm is a large amount, it may converge those AFs to a local optimum.

4.3 PERFORMANCE EVALUATION METRICS

Performance of the six algorithms is evaluated and compared using the following metrics:
Sum of Intra Cluster Distances (SICD): The distance between each data vector within a cluster and the corresponding cluster center is calculated and summed up. As it discussed, Eq. (2.7) is used for calculating this metric which has to be minimized.

Offline SICD: The average SICD obtained by algorithm in all iterations. For standard AFSA, PSO, KPSO, NAFSA and K-NAFSA methods, the average intra cluster distance during iterations is equal to the average fitness value of the best swarm member in all the iterations which is calculated by Eq. (4.3):

$$Offline\_ICD = \frac{1}{Max\_Itr} \sum_{Itr=1}^{Max\_Itr} (f(Swarm_{Best}(Itr))) \tag{4.3}$$

Max_Itr is the number of final iteration. $SwarmBest(Itr)$ in PSO and KPSO is the Gbest fitness value in iteration Itrth, in standard AFSA, is $bulletin$ fitness value in iteration Itrth and in NAFSA and K-NAFSA is the position of the best AF in iteration Itrth , where $f(SwarmBest(Itr))$

is SICD for the best AF. For k-means algorithm, we replace SICD of clusters′centers in iteration Itr with $f(SwarmBest(Itr))$ in Eq. (4.3). Offline SICD is a metric for comparing exploration ability of the algorithms. In fact, approaching global optimum in fewer numbers of iterations by an algorithm leads to less offline SICD. If the result of two methods based on SICD is close to each other, the method involving lower offline SICD, has more ability in exploration and converges toward solution in fewer number of iterations.

Error rate: It is defined as the number of misplaced points amongst all of the points in the dataset which are not correctly clustered. This metric is calculated by Eq. (4.4):

$$Err = \frac{1}{N}\sum_{i=1}^{N}(if(Class(i) = Cluster(i))\ then\ 0\ else\ 1)) \times 100 \tag{4.4}$$

where, N is the total number of the points in dataset and Class(i) represents the class number which point i belongs to and $Cluster(i)$ represents the cluster number to which point i is assigned.

Average number of fitness evaluations: Applied algorithms will be compared according to the average of the number of fitness evaluation during execution that is one of the metrics of algorithms complexity.

4.4 EXPERIMENTAL RESULTS

The obtained results in this research are averages of 50 runs of simulation. The algorithms are implemented using MATLAB 7.6 on a machine with Core i7 2.80 GHz processor and 4GB RAM. Table 2 represents the best, average and standard deviation of SICD resulted from 50 times runs of standard AFSA, PSO, KPSO, k-means, NAFSA and K-NAFSA algorithms on 12 datasets of table As it is observed in Table 2, the efficiency of proposed algorithm (K-NAFSA) is considerably better than that of the other algorithms. The best obtained result of K-NAFSA and its results average in all datasets outperforms others algorithms. Also, standard deviation of the obtained results, excluding two cases, is less than that for other algorithms. Standard deviation value for Art3 and Art5 in K-NAFSA is not less than other algorithms, however, the obtained results of the best SICD and the mean metric of the proposed approach are considerably better, compared to those of other methods. Totally, the standard deviation resulted from K-NAFSA represents its robust to initial points of the clusters centers and high ability of this method, in terms of passing local optimums.

It is also can be seen that, the results of k-means and standard AFSA generally leads to the worst results. In standard AFSA, due to the lack of balance between exploration and exploitation, the robust of the algorithm is low. Also, the problems of $Visual$ and $step$ parameters result in decreasing efficiency. The effectiveness of standard AFSA is significantly declined, when the number of problem dimensions is high (it is obvious in results of Sonar, Art5 and Glass). In

Art3 and Art4, the results of standard AFSA is better than PSO and KPSO, since the length of search space range in different dimensions is the same in these two datasets, therefore, it can search through problem space by assuming an appropriate value of $Visual$ and step. Also, the number of problem dimensions in these two datasets is not high and standard AFSA, gives better results in problems with low dimensions, in contrast with problems with high dimensions. In k-mean method, due to sensitivity of the algorithm to initial cluster centers and its low capability in passing local optimums, the obtained results are not acceptable. Regarding the low ability of this algorithm in passing local optimums, the obtained results of k-means algorithm is substantially worse than those other algorithms.

Table 2: Comparison of SICD for the Six Clustering Algorithms

Dataset	Criteria	K-means	Std-AFSA	PSO	KPSO	NAFSA	K-NAFSA
	Best	1535.66	1541.6	1535.3	1535.15	1534.87	**1534.86**
Art1	Mean	1540.22	1548.65	1539.7	1536.17	1535.15	**1534.86**
	Std.Dev	26.87	3.65	4.23	0.69	0.57	**0.00**
	Best	1938.21	1951.88	1939.7	1937.03	1936.95	**1936.94**
Art2	Mean	1946.75	1975.74	1942.83	1937.63	1936.98	**1936.94**
	Std.Dev	60.44	12.95	3.42	0.33	0.1	**0.01**
	Best	2497.93	2239.38	2062.69	2121.4	1990.46	**1989.89**
Art3	Mean	3227.06	2336.72	2389.03	2454.29	2130.86	**2021.25**
	Std.Dev	570.03	62.23	196.24	184.92	240.65	**121.41**
	Best	1742.7	1944.63	1803.44	1740.68	1741.19	**1739.93**
Art4	Mean	2855.25	2032.41	2169.63	2095.09	1742.88	**1739.93**
	Std.Dev	609.69	55.99	302.05	309.55	1.03	**0.00**
	Best	1910.41	3258.78	2769.86	1590.83	1908.11	**1236.27**
Art5	Mean	2337.45	3377.25	3204.64	2256.48	1924.43	**1470.95**
	Std.Dev	344.7	43.77	341.85	384.89	8.03	**147.2**
	Best	97.32	96.91	97.1	96.78	96.66	**96.65**
Iris	Mean	102.57	112.32	102.26	99.61	96.69	**96.65**
	Std.Dev	11.34	5.46	5.81	7.21	0.03	**0.00**
	Best	52072.24	47899.52	47627.73	47606.02	47661.79	**47561.33**
Pima	Mean	55076.52	47974.22	48153.18	47876.12	48635.55	**47561.62**
	Std.Dev	7790.37	47.87	523.15	284.12	1425.5	**0.11**
	Best	16555.68	16695.47	16307.16	16298.92	16307.02	**16292.32**
Wine	Mean	17662.73	16844.1	16320.67	16307.58	16310.34	**16292.41**
	Std.Dev	1878.07	57.19	9.53	7.23	1.56	**0.05**
	Best	5542.18	5754.29	5560.75	5535.06	5534.1	**5532.21**
CMC	Mean	5543.85	6508.48	5609.85	5538.44	5542.4	**5532.23**
	Std.Dev	1.59	234.34	52.71	2.24	3.73	**0.00**
	Best	215.42	348.93	230.64	212.03	212.7	**210.01**
Glass	Mean	241.03	367.52	258.02	233.28	213.76	**213.68**
	Std.Dev	25.32	8.72	12.24	14.05	1.34	**1.29**
	Best	152647.25	152946.16	149537.73	149480.93	149523.15	**149474.26**
WDBC	Mean	179794.25	153388.98	149830.87	149594.05	149610.87	**149474.4**
	Std.Dev	55222.17	211.59	364.73	198.31	55.77	**0.12**
	Best	234.77	399.66	271.83	234.65	233.79	**233.75**
Sonar	Mean	235.06	432.89	276.68	234.92	233.81	**233.75**
	Std.Dev	0.15	15.47	3.79	0.22	0.01	**0.00**

According to Table 2, results of NAFSA, in terms of all performance metrics have been improved considerably than standard AFSA. One of the reasons of such improvement is establishing a balance between exploration and exploitation. Another reason is the modification of $Visual$ parameter as a vector which enables AFs to search in all dimension of the search space based on the search space range. Also, other changes in parameters, behaviors and procedure of executing AFSA and NAFSA leads to suitable group movements by AFs and also an individual movements in each of the iterations and consequently, improving exploration and exploitation of the algorithm. In Table 2, the results of Standard deviation show the superiority of NAFSA than standard AFSA.

Enriching NAFSA by adding k-means as a behavior causes considerable improvement on the results of NAFSA, in case of data clustering. In K-NAFSA, the convergence rate is significantly improved, since one iteration belongs to k-means is performed on some AFs in each iteration of the algorithm execution. In addition, by performing the algorithm on random AFs, trapping in local optimums is prevented. The advantage of K-NAFSA compared to KPSO is that k-means is performed just on one set of clusters centers, and until converging the algorithm to an optimum. After executing k-means, PSO is responsible regarding leaving local optimums by clusters′centers for reaching better results. However, it is possible that other particles in PSO are not able to pass local optimums to move toward global optimum. As a result, strategy of utilizing k-means in K-NAFSA is better than KPSO. Also, as it is observed in Table 2, NAFSA reaches to better results than PSO in most cases. Fig. 4 illustrates the convergence behaviors of the six algorithms for the datasets presented in Table 1.

In graphs presented in Fig. 4, to observe precise details concerning optimization process of intra-cluster distances, we had to limit vertical component of graph from above. Therefore, the difference between results of the algorithms would be more observable. As it seen in Table 2, the average results of the standard AFSA and k-means on Wine, WDBC and Sonar datasets are significantly worse than those of PSO, KPSO, NAFSA and K-NAFSA, so that standard AFSA and k-means curves in graphs (h), (k) and (i) are in higher values of vertical axis of intra-cluster distances which is not observable.

As can be seen in Fig. 4, k-means has high convergence rate in first iterations, but it converges fast to a local optimum and in the next iterations it cannot reach better results and thus, it remains constant until the end of the iterations. In KPSO, since k-means algorithm is executed until converges to a point, therefore, its convergence rate at the beginning of the algorithm execution is very fast, in a similar manner of k-means. However, in a different manner of k-means, by performing PSO algorithm, the algorithm passes local optimum and thus, better results are obtained. Nevertheless, SICD values in the first iterations (up to 10^{th} iterations) of k-means and KPSO are greater than the other algorithms. Because both algorithms perform k-means procedure only on one solution vector including cluster centers at the beginning and thus, there is a high probability for occurring high SICD values in the vector. On the other hand, in PSO, standard AFSA, NAFSA, K-NAFSA, because of existing N solution vectors (N is equal to the number of particles or artificial fishes), probability of occurring low value for SICD is high. Thereby, SICD value for these algorithms is better than that for k-means and KPSO at the beginning of execution.

SICD value for K-NAFSA is less than that for other algorithms at the beginning of clustering, since not only the number of solution vectors is equal to the number of artificial fishes, but also k-means is performed on some of them which leads to dramatic declining SICD value in the first iteration. So, in graphs shown after first iteration, SICD for K-NAFSA is less than the other algorithms. This issue is more tangible in the case of existing considerable number of artificial fishes i.e. Art5, WDBC and Sonar. Standard AFSA has less convergence rate than NAFSA in

all cases and the gradient of SICD for NAFSA is more than that of standard AFSA.

In Table 3, average offline SICD metric calculated for all algorithms on the mentioned datasets are presented. Offline SICD value is related to both convergence speed and final obtained SICD value. When obtained SICD value is closer to its optimal value in all iterations, offline SICD is smaller. It is possible that average value of SICD of an algorithm be better than that of another algorithm, however, its convergence speed be lower than another algorithm. Thus, its offline SICD would be worse (larger). For example, this situation is happened for PSO and k-means in three cases of Art1, Art2 and Iris. It shows that in lower number of iterations, k-means obtain better SICD than PSO. If both mean value of SICD and value of offline SICD of an algorithm are better than another one, it is obvious that both exploration and exploitation abilities of this algorithm are better. In accordance with represented information in Tables 2 and 3, SICD and offline SICD for K-NAFSA are better than those for other methods which are obviously seen in Fig. 4. In fact, because of using k-means in K-NAFSA algorithm, better values of SICD in fewer number of iterations is reached, compared to other methods.

Table 3: Comparison of Offline SICD for the Six Clustering Algorithms.

Data Set	k-means	Std-AFSA	PSO	KPSO	NAFSA	K-NAFSA
Art1	1557.09±29.31	1557.85±3.95	1566.35±10.89	1544.01±8.81	1540.65±1.58	**1538.14±1.05**
Art2	1973.94±59.97	1995.69±12.27	1998.42±25.16	1954.48±15.97	1949.85±3.39	**1943.68±2.80**
Art3	3265.85±565.52	2542.85±94.84	2735.79±182.40	2758.48±216.28	2295.43±20.58	**2099.31±107.28**
Art4	2890.63±599.90	2259.95±47.03	2477.09±234.99	2281.30±311.28	1981.49±61.62	**1781.29±13.08**
Art5	2292.08±349.35	3459.93±35.51	3428.52±327.32	2267.31±383.64	2087.14±22.81	**1481.72±143.48**
Irish	104.34±11.25	116.25±3.80	114.19±7.58	102.24±9.09	98.98±0.38	**96.95±0.11**
Pima	55649.81±7869.75	49546.35±451.75	49470.65±952.49	48965.17±589.15	54802.5±31573.16	**47589.29±6.21**
Wine	17715.11±1893.44	17019.06±54.56	16367.82±17.99	16348.81±24.70	16336.49±3.35	**16294.32±0.64**
CMC	5568.07±11.59	6614.83±120.28	5717.23±49.78	5548.22±6.38	5638.80±13.53	**5534.85±0.41**
Glass	242.23±25.21	376.97±5.45	273.47±9.93	237.04±15.07	222.68±3.42	**214.44±1.20**
WDBC	180544.61±55121.05	160506.52±2189.58	150686.77±518.3	150050.85±496.44	150566.51±238.80	**149480.50±1.31**
Sonar	235.23±0.17	432.89±15.47	284.97±3.15	235.04±0.17	236.55±0.12	**233.83±0.00**

Table 4 shows the average of error rate and its standard deviation of clustering process by the 6 algorithms on the datasets over 50 runs. It can be seen that, the average of error rate obtained by K-NAFSA on 9 datasets excluding Pima, CMC and Sonar is less than that by the other algorithms However, the average of SICD obtained by K-NAFSA is better than these three datasets.

Table 4: Comparison of Error Rate for the Six Clustering Algorithms.

Data Set	k-means	Std-AFSA	PSO	KPSO	NAFSA	K-NAFSA
Art1	3.45±6.50	8.62±3.46	4.55±2.37	2.97±1.02	2.87±1.33	**2.27±0.06**
Art2	0.68±4.84	2.96±2.63	**0±0**	**0±0**	**0±0**	**0±0**
Art3	31.71±11.63	10.51±6.12	7.40±8.30	11.08±7.41	3.92±6.75	**1.07±4.13**
Art4	30.09±13.01	2.52±6.01	9.05±11.45	7.45±10.48	**0±0**	**0±0**
Art5	41.83±10.03	57.14±1.05	58.09±5.22	39.86±12.06	29.66±2.05	**15.58±9.58**
Irish	16.05±10.10	27.89±9.01	10.64±4.50	12.58±7.67	10.15±0.28	**10.00±00**
Pima	**34.10±0.31**	38.93±0.93	39.40±0.83	39.49±0.82	39.14±0.88	39.19±7.22e-15
Wine	34.38±6.08	29.11±0.43	28.74±0.39	28.59±0.47	28.41±0.28	**28.09±3.61e-015**
CMC	60.08±0.20	**57.72±1.92**	60.26±0.39	60.32±0.22	60.40±0.13	60.55±7.23e-15
Glass	48.30±3.14	54.07±3.24	48.72±1.34	47.80±1.98	50.52±2.04	**47.78±3.65**
WDBC	19.12±9.22	13.28±0.28	**13.18±1.80e-15**	**13.18±1.81e-15**	**13.18±1.81e-15**	**13.18±1.81e-15**
Sonar	44.95±0.97	46.63±1.46e-14	46.60±0.42	**44.89±0.84**	46.23±0.18	46.15±7.29e-15

In Table 5, average number of fitness evaluations used by each of the algorithms in one iteration of their executions is shown. In fact, results presented in Table 5 indicate the average number of calculating SICD by Eq. (2.7) in each iteration. K-means algorithm has the minimum number of fitness evaluations among all 6 algorithms. This algorithm, only calculates SICD once in each iteration, however, its results are not acceptable. In PSO, the number of fitness evaluation is equal to the number of swarms in each iteration. However, in KPSO, first, k-means is performed until it is trapped in local optimums. Thus, the number of iterations is not constant.

Table 5: Comparison of Average Number of Fitness Evaluation for the Six Clustering Algorithms in each Iteration.

Data Set	k-means	Std-AFSA	PSO	KPSO	NAFSA	K-NAFSA
Art1	1	310.37	40	31.33	145.55	157.64
Art2	1	324.07	40	33.92	157.17	159.19
Art3	1	522.03	70	53.22	271.45	281.27
Art4	1	561.08	75	70.32	279.51	302.39
Art5	1	3914.68	350	347.12	1264.82	1382.3
Irish	1	467.61	60	56.63	226.53	232.06
Pima	1	681.49	80	71.74	300.21	309.59
Wine	1	2073.28	195	188.12	727.25	758.84
CMC	1	1460.13	135	127.04	528.94	542
Glass	1	2506.51	270	265.1	992.18	1050.53
WDBC	1	3188.26	300	296.06	1138.77	1200.85
Sonar	1	6135.09	600	589.56	2266.06	2415.47

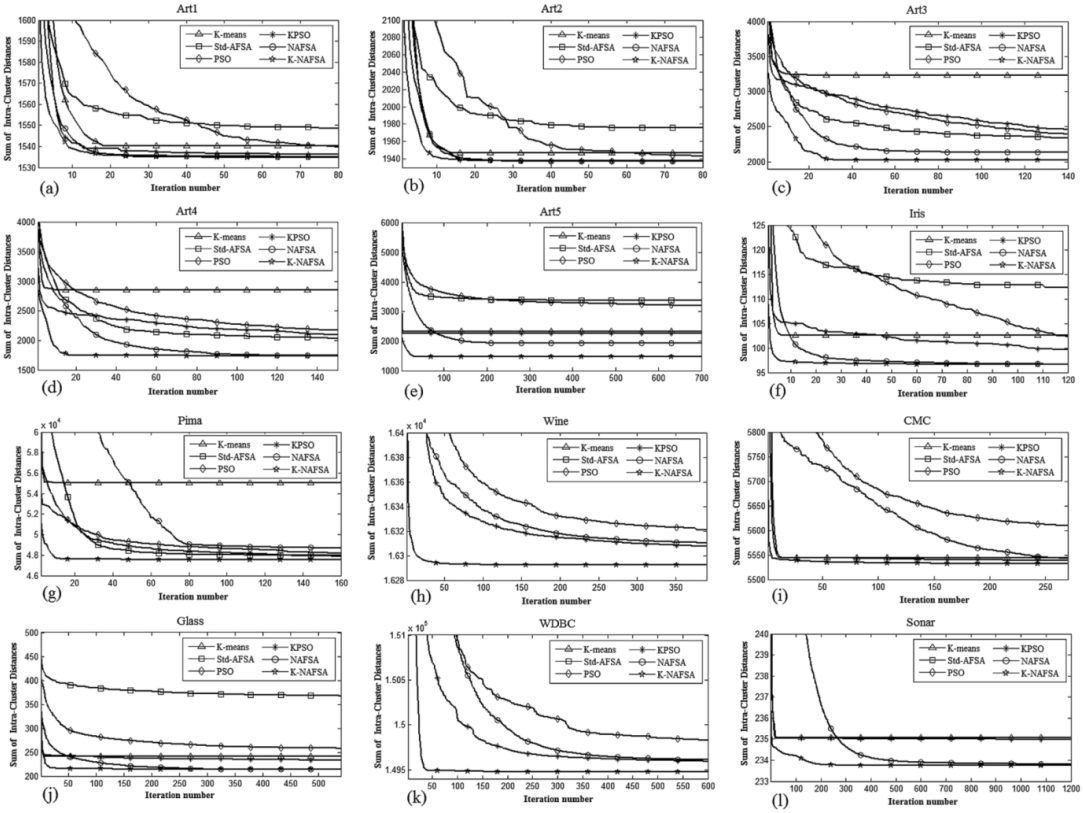

Figure 4: Convergence Behaviors of the Six Algorithms on the Datasets Presented in Table 1.

In standard AFSA, since every AF performs several fitness evaluations in each of the iterations, therefore, the number of fitness evaluations of this algorithm is high, while its results are not acceptable at all. In standard AFSA, every AF, according to the times tried to find a better position in its prey behavior, may have different number of fitness evaluations. When AFs are trapped in local optimums and finding better positions around them is difficult, to find better position attempt try _ number times that leads to an increment on the number of fitness evaluations. Indeed, in accordance with the position of AFs, in standard AFSA, we may have different number of fitness evaluations in each of the iterations.

In NAFSA, each AF performs at least $try\_number + 1$ and at most $try\_number + 2$ fitness evaluations in each iteration for group behavior and try_number fitness evaluation for individual behavior. In the case of no success, a fitness evaluation would be done for random move of AF by using Eq. (3.4).In K-NAFSA, for the number of AFs on which k-means is performed, more fitness evaluations than NAFSA is done in each of the iterations. The required number of fitness evaluations for NAFSA and K-NAFSA is much fewer than standard AFSA. In fact, NAFSA and K-NAFSA need less population than standard AFSA for optimization and they do not waste their fitness evaluation, in contrast of AFSA. Evaluation numbers of NAFSA and K-NAFSA is about 4-6 times more than that of PSO and KPSO, but their results are better.

The number of fitness evaluations calculated by algorithms to reach the solution is one of the metrics for comparing algorithms′ complexity. As it was shown in Table 5, the number of performed fitness evaluations by the proposed algorithm is more than PSO, KPSO and k-means algorithms. As discussed previously, the minimum number of fitness evaluations belongs to k-means algorithm, but this method does not reach to acceptable results in comparison with the other algorithms. Among compared algorithms with the proposed algorithms, generally, KPSO algorithm has obtained better results. In Table 6, the average number of fitness evaluations calculated by two proposed algorithms to reach KPSO algorithm result, were shown. In column related to KPSO, the number of fitness evaluations used by this algorithm was represented at $10 \times D$ iterations. The average number of fitness evaluations used by NAFSA and K-NAFSA algorithms until reach to obtained average final result by KPSO, were shown in related columns. Dashed lines corresponding to NAFSA mean this algorithm could not reach KPSO results at $10 \times D$ iterations.

Table 6: Comparison of Average Number of Fitness Evaluations performed by KPSO, NAFSA and K-NAFSA Algorithms.

Data Set	KPSO	NAFSA	K-NAFSA
Art1	**2506.65**	3176.63	2765.22
Art2	**2714.37**	5485.5	4116.18
Art3	7451.91	5937.82	**3268.75**
Art4	10549.15	8578.71	**2305.15**
Art5	242983.56	104680.74	**5266.25**
Irish	**6796.04**	20347.86	9448.57
Pima	11478.91	-	**2774.56**
Wine	73365.71	-	**8326.77**
CMC	34300.72	-	**15032.15**
Glass	143153.89	77099.69	**7994.97**
WDBC	177636.69	-	**37866.39**
Sonar	707472.22	624809.7	**11920.45**

As it is obvious, K-NAFSA algorithm reached KPSO results with much fewer fitness evaluations, when problem dimension were high, but in cases with low dimensions, K-NAFSA algorithm could not reach KPSO result with fewer fitness evaluations. Totally, results of Table 6 show that the proposed algorithm (K-NAFSA) can achieve acceptable results with fewer numbers of iterations, and, consequently, fewer fitness evaluations. In three cases in which K-NAFSA could not obtain KPSO results with less fitness evaluations, the number of fitness evaluations is fewer than two times of KPSO fitness evaluations number. Whereas, in cases with high dimensions, it performs fitness evaluations considerably less than KPSO to reach KPSO results. In cases, in which NAFSA obtains better results than KPSO, the number of fitness evaluations performed on K-NAFSA in high dimensions is less than KPSO.

In the proposed method, NAFSA and k-mean methods are combined as a novel approach. However, it could be wrongly considered that by performing k-means from different initial positions same results are obtained and NASFA does not play an important role in performance of K-NAFSA. To clear the situation, results of K-NAFSA are compared to that of Multiple K-means (MK-means) method in Table 7.

Table 7: The results of the proposed method (K-NAFSA) and MK-means on the 12 datasets.

Data Set	MK-MEANS	K-NAFSA
Art1	1535.65	**1534.86**
Art2	1938.2	**1936.94**
Art3	2431.44	**2021.25**
Art4	1742.7	**1739.93**
Art5	1835.02	**1470.95**
Irish	97.32	**96.65**
Pima	52072.24	**47561.62**
Wine	16555.67	**16292.41**
CMC	5542.18	**5532.23**
Glass	215.24	**213.68**
WDBC	152647.25	**149474.4**
Sonar	234.76	**233.79**

MK-mean algorithm starts its procedure from 5000 random initial positions for each dataset. The mentioned results of this method shown in Table 7 are best result obtained amongst the 5000 initial positions. As can be seen, K-NAFSA outperforms MK-means in all datasets. Thus, important role of NAFSA is obvious in K-NAFSA. One of the reasons for such superiority is exploitation ability of NAFSA than that of K-means. Indeed, convergence speed of K-means in local search is high, however, its accuracy, in case of calculating final results, is not satisfactory. On the other hand, K-means is considered as a movement $step$ in K-NAFSA which is performed on some AFs in each iteration. Thus, NAFSA procedure is considerably important for preventing form local optimums and increasing diversity among solutions in K-NAFSA.

5 CONCLUSION

In this paper, a new artificial fish swarm algorithm called NAFSA was proposed, in which weak points of the standard AFSA have been resolved. Also, a new hybrid algorithm based on NAFSA and k-means, so called K-NAFSA, was presented for data clustering. In the proposed

method, k-means was applied as a behavior for artificial fish. Efficiency of the proposed algorithm was compared to that of k-means, standard AFSA, PSO and KPSO approaches which were applied on 12 datasets. Different performance metrics including Sum of Intra Cluster Distances (SICD), offline SICD, error rate and average number of fitness evaluations were employed in order to performance evaluations and comparisons. Experimental results showed the superiority of the proposed method, in terms of its efficiency and robustness.
Nevertheless, the proposed algorithm is not applicable, when the number of clusters is not predefined. This problem will be considered as a future work.

References

Cheng, Y., Jiang, M. and Yuan, D. 2009. Novel clustering algorithms based on improved artificial fish swarm algorithm, *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, Vol. 3, IEEE, pp. 141–145.

Cheng, Z. and Hong, X. 2012. Pid controller parameters optimization based on artificial fish swarm algorithm, *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference on*, IEEE, pp. 265–268.

Emre Celebi, M. 2011. Improving the performance of k-means for color quantization.

Frank, A. and Asuncion, A. 2010. Uci machine learning repository.

Hartigan, J. A. 1975. An overview of clustering algorithms, *New York* .

He, S., Belacel, N., Hamam, H. and Bouslimani, Y. 2009. Fuzzy clustering with improved artificial fish swarm algorithm, *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, Vol. 2, IEEE, pp. 317–321.

Jiang, M. and Zhu, K. 2011. Multiobjective optimization by artificial fish swarm algorithm, *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, Vol. 3, IEEE, pp. 506–511.

Kao, Y., Zahara, E. and Kao, I. 2008. A hybridized approach to data clustering, *Expert Systems with Applications* **34**(3): 1754–1762.

Lei, L. X., Shao, Z. J. and Qian, J. X. 2002. An optimizing method based on autonomous animate: fish swarm algorithm, *System Engineering Theory and Practice*, Vol. 11, pp. 32–38.

MacQueen, J. et al. 1967. Some methods for classification and analysis of multivariate observations, *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, number 281-297 in *1*, California, USA, p. 14.

Memarsadeghi, N. and O'Leary, D. 2003. Classified information: the data clustering problem, *Computing in Science & Engineering* **5**(5): 54–60.

Pizzuti, C. and Talia, D. 2003. P-autoclass: Scalable parallel clustering for mining large data sets, *Knowledge and Data Engineering, IEEE Transactions on* **15**(3): 629–641.

Rocha, A., Martins, T. and Fernandes, E. 2011. An augmented lagrangian fish swarm based method for global optimization, *Journal of computational and applied mathematics* **235**(16): 4611–4620.

Shen, W., Guo, X., Wu, C. and Wu, D. 2011. Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, *Knowledge-Based Systems* **24**(3): 378–385.

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

Steinhaus, H. 1956. Sur la division des corp materiels en parties, *Bull. Acad. Polon. Sci* **1**: 801–804.

Tian, W. and Tian, Y. 2009. An improved artificial fish swarm algorithm for resource leveling, *Management and Service Science, 2009. MASS'09. International Conference on*, IEEE, pp. 1–4.

Tsai, H. and Lin, Y. 2011. Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior, *Applied Soft Computing* **11**(8): 5367–5374.

Van der Merwe, D. and Engelbrecht, A. 2003. Data clustering using particle swarm optimization, *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, Vol. 1, IEEE, pp. 215–220.

Vannoorenberghe, P. and Flouzat, G. 2006. A belief-based pixel labeling strategy for medical and satellite image segmentation, *Fuzzy Systems, 2006 IEEE International Conference on*, IEEE, pp. 1093–1098.

Wong, A. and Li, G. 2008. Simultaneous pattern and data clustering for pattern cluster analysis, *Knowledge and Data Engineering, IEEE Transactions on* **20**(7): 911–923.

XiaoLi, C., Ying, Z., JunTao, S. and JiQing, S. 2010. Method of image segmentation based on fuzzy c-means clustering algorithm and artificial fish swarm algorithm, *Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on*, IEEE, pp. 254–257.

Yang, X., Song, Q. and Zhang, W. 2006. Kernel-based deterministic annealing algorithm for data clustering, *Vision, Image and Signal Processing, IEE Proceedings-*, number 5 in *153*, IET, pp. 557–568.

Yazdani, D., Nabizadeh, H., Mohamadzadeh Kosari, E. and Nadjaran Toosi, A. 2011. Color quantization using modified artificial fish swarm algorithm, *AI 2011: Advances in Artificial Intelligence* pp. 382–391.

Yazdani, D., Nadjaran Toosi, A. and Meybodi, M. 2011. Fuzzy adaptive artificial fish swarm algorithm, *AI 2010: Advances in Artificial Intelligence* pp. 334–343.

Zhang, M., Shao, C., Li, M. and Sun, J. 2006. Mining classification rule with artificial fish swarm, *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, Vol. 2, IEEE, pp. 5877–5881.