A New Artificial Fish Swarm Algorithm for Dynamic Optimization Problems

[1]Danial Yazdani
Department of Electrical, Computer and IT Engineering, Qazvin Branch, Islamic Azad University, Iran
d_yazdani@qiau.ac.ir

[2] Mohammad Reza Akbarzadeh-Totonchi
Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University of Mashhad, Iran
akbarzadeh@ieee.org

[3]Babak Nasiri
Department of Electrical, Computer and IT Engineering, Qazvin Branch, Islamic Azad University, Iran
nasiri.babak@qiau.ac.ir

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

Abstract— **Artificial fish swarm algorithm is one of the swarm intelligence algorithms which performs based on population and stochastic search contributed to solve optimization problems. This algorithm has been applied in various applications e.g. data clustering, neural networks learning, nonlinear function optimization, etc. Several problems in real world are dynamic and uncertain, which could not be solved in a similar manner of static problems. In this paper, for the first time, a modified artificial fish swarm algorithm is proposed in consideration of dynamic environments optimization. The results of the proposed approach were evaluated using moving peak benchmarks, which are known as the best metric for evaluating dynamic environments, and also were compared with results of several state-of-the-art approaches. The experimental results show that the performance of the proposed method outperforms that of other algorithms in this domain.**

Keywords- dynamic optimization problems; artficial fish swarm algorithm; moving peaks benchmark; dynamic environments.

I. INTRODUCTION

Artificial fish swarm algorithm (AFSA) is one of the algorithms inspired both from the nature and swarm intelligence algorithms. AFSA was presented by Li Xiao Lei in 2002[1]. This algorithm is an approach based on swarm behaviors that was inspired from social behaviors of fish swarm in the nature. This algorithm has some characteristics such as high convergence rate, insensibility to initial values, flexibility and high fault tolerance. AFSA has been utilized in optimization applications e.g. PID controller parameters setting[2] multi-objective optimization[3], global optimization[4] , neural network learning[5], data clustering[6], color quantization[7] and etc.

In many real world problems, uncertainty is obvious and clear. Swarm intelligence and evolutionary algorithms could be contributed to solve these kinds of problems. Uncertain problems could be divided into four categories: existence of noise in evaluation function, disturbance in design variables, approximation in fitness function and time-dependent fitness function. In this paper, time-dependent fitness function has been considered, which belongs to the most general types of uncertainties.

Up to now, different methods have been presented for solving dynamic problems by means of evolutionary processing[8] and swarm intelligence methods[9]. Three main challenges of evolutionary and swarm intelligence algorithms that cause inability in direct use of these methods for optimization in dynamic environments are memory limitations, losing diversity and existing several potential optimums in the problem space. When the environment is changed, current obtained solutions in memory are not valid anymore. Thus it should be either forgotten completely or should be evaluated again. In addition, since most evolutionary algorithms and swarm intelligence methods converge to a point due to their characteristics, group diversity is lost in the environment and in case of changes in the environment, convergence to a new optimal point would be impossible or time-consuming and slow. There are various methods for generating or maintaining group diversity in the environment that will be discussed as follows.

The simplest way for reacting against environment change is considering any change as an entrance of a new optimization problem which must be solved again. In case of lack of time limitations, this solution is an appropriate option. However, frequently this time is rather short for re-optimization. A natural attempt to accelerate optimization process after a change is using related information of the search space before change in order to improve the search after change. For instance, if it is supposed that a new optimal point is near the previous optimal one, it can limit the search space to the previous optimal neighbor point. The case whether reusing previous information is suitable depends on change characteristics.

If the change is broad and there is little similarity between the environments before and after change, restarting the optimization algorithm can be the only option and any new use of collected information based on the problem space before change will be misleading. In most real world problems, changes are hoped to be rather smooth, which means there should be a relation between the problem environment before change and the problem environment after change, in order to be able to use previous information for improving optimization process. The fundamental question discussed here is that what information should be kept and how should this information be used to accelerate the search process after changes in the environment.

Another challenge is that in most optimization algorithms, after the algorithm converges to a point, it loses its diversity which leads to decreased compatibility with changes in the environment. In fact, when algorithms agents converge to a point, they lose their ability of following this point after its displacement. Therefore, beside transferring information among optimization algorithms agents before and after the changes, we should search ways to increase diversity and compatibility of the algorithm after environment change so that these agents could follow the goal point after displacement and converge fast to the new position of the goal point.

Another challenge in solving dynamic problems is that there are several potential optimums in these problems. In fact, there are some peaks in dynamic environments whose width, height and location may be varied after environment change in different problems. Therefore, in such problems, each of these peaks can be modified to the global optimum after environment change. So, the algorithms which are designed for solving these problems should have the ability to cover all peaks so that whenever one of them is transformed to a global optimum, they can find it fast.

Multiswarm is an appropriate way to resolve this problem. In [9], there were some predetermined groups that consist of some agents and every group has to cover one peak. The problem with this method was that if the number of peaks is not equal to the number of groups, algorithm's efficiency was decreased. In[10], the number of groups was determined adaptively according to the number of found peaks in the problem space.

Generally, algorithms in which the number of groups is determined with respect to the number of found peaks in the problem space have more efficiency than those with a constant number of groups. The best condition is that each peak is covered by one group of agents, but sometimes more than one group may converge to a peak. To solve this problem, exclusion was used in [8]. In this method, when two swarms get close more than a specific distance called r_{excl}, the group with the worse position would be reinitialized. Therefore, in each peak, one group can reside at most.

In this paper, a modified AFSA is proposed for optimization in dynamic environments. All requirements of dynamic environments were satisfied in proposed algorithm. In this algorithm, basic behaviors mechanisms of standard AFSA have been changed which are: Prey, Follow, Swarm and Free_move. Also procedure of basic algorithm is completely changed. The proposed algorithm has been applied on different configuration of moving peaks benchmark (MPB) [11] that is one of the most famous benchmarks of dynamic environments. Performance of proposed algorithm is compared with ten other algorithms which were presented for optimizing MPB. Comparing criterion is offline_error which is the main comparing criteria of designed algorithms for dynamic environments [11]. Experimental results show that the proposed algorithm has a suitable efficiency. In following, this paper is organized as: in section two, AFSA algorithm will be described briefly. The proposed algorithm will be discussed in section three. In section four, experiments and their results will be discussed and last section will present the conclusion.

II. ARTIFICIAL FISH SWARM ALGORITHM

In water world, fishes 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. The environment, which AF lives in, substantially is solution space and other AF's domain. Food consistence degree in water area is AFSA objective function. Finally, AFs reach to a point which its food consistence degree is maxima (global optimum).

As it is observed in figure 1, AF perceives external concepts with sense of sight. Current position of AF is shown by vector $X=(x_1, x_2, \ldots, x_n)$. The *visual* is equal to sight field of AF and X_v is a position in *visual* where the AF wants to go. Then if X_v has better food consistence 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 is better than X_v, it continues searching in its *visual* area. Food consistence in position X is fitness value of this position and is shown 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) and *crowd factor* δ (0<δ<1). Also functions consist of prey behavior, free move behavior, swarm behavior and follow behavior.

In each step of optimization process, AF looks for locations with better fitness values in problem search space by performing these four behaviors based on algorithm procedure[13].

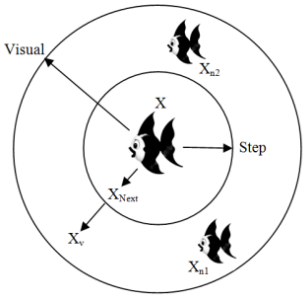

Figure 1. Artificial Fish and the Environment Around It.

III. THE PROPOSED ALGORITHM

In this section, a new algorithm based on artificial fish swarm algorithm is presented for optimization in dynamic environment. In proposed algorithm, parameters, behaviors and AFSA procedure are modified to be appropriate for optimization in dynamic environments. In the proposed algorithm, prey, follow and swarm behaviors are done for AFs which have main differences with prey, follow and swarm behaviors in Standard AFSA[5]. The reason to perform these changes is to adapt AFSA for working in dynamic environments. In following, after description of modified artificial fish swarm algorithm (MAFSA), a new algorithm based on MAFSA will be presented for dynamic environments.

A. Modified AFSA algorithm

First, we discuss about MAFSA behaviors.

1) Prey behavior

This behavior is an individual behavior and each AF performs it without considering other swarm members. Along performing this behavior, each AF does a local search around itself. By performing this behavior each AF attempts *try_number* times to replace to a new position with better fitness. Let suppose AF i is in position X_i and wants to execute prey behavior. Following steps are done in prey behavior:

- AF i considers a target position in *visual* by equation (1), then evaluates its fitness value. *d* is dimension number and Rand generates a random number with uniform distribution in [-1, 1]:

$$X_{T,d} = X_{i,d} + Visual \times Rand_d(-1,1) \quad (1)$$

- If fitness value of position X_T is better than current position of AF i, position will be updated by equation (2):

$$\vec{X}_i = \vec{X}_T \quad (2)$$

Steps a and b are performed *try_number* times. By executing above steps, in the best case, an AF can update its position at most *try_number* times and move toward better positions. In the worst case, none of AF's attempts to find better position will be succeed. Then in this situation after performing prey behavior, there will be no replacement at all.

2) Follow behavior

AFs in Standard AFSA in case of not finding better positions in performing standard prey behavior, move one step randomly [11, 12] and lose their previous position. But, in MAFSA as it was discussed in 3-1-1, in prey behavior if an AF is not able to move to better positions, it won't move at all and will keep its previous position. This causes that the best AF (according to fitness value) of swarm locates in the best found position by swarm member so far. The reason is that in prey behavior in MAFSA, an AF displace if only moves to a better position. In follow behavior, each of AFs moves one step toward the best AF of swarm by using equation 3:

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \frac{\vec{X}_{Best} - \vec{X}_i(t)}{Dis_{i,Best}} \times [Visual \times Rand(0,1)] \quad (3)$$

X_i is position vector of AF i which performs follow behavior and x_{best} is position vector of the best AF in swarm. Therefore, AF i moves at most to *Visual* extent in each dimension toward the best AF of swarm. In fact, after finding more food by a fish, other swarm members follow after it to reach more foods. Following the best AF of swarm causes convergence rate increase and helps to keep integrity of AFs in a swarm. This behavior is a group behavior and interactions between swarm members are done globally.

3) Swarm behavior

This function is a group behavior too and is performed globally among members of swarm. In swarm behavior, first of all, central position of swarm is calculated in terms of arithmetic average of all swarm members position in every dimension. Central position of swarm X_{center} is obtained by:

$$X_{Center,d} = \frac{1}{N} \sum_{i=1}^{N} X_{i,d} \quad (4)$$

As it is observed, component d of vector X_{center} is the arithmetic mean of component d of all AFs of swarm. For AF i, move condition toward central position is checked, i.e. $f(X_{Center}) \geq f(X_i)$ and if this condition is satisfied, next position of AF i is obtained by:

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \frac{\vec{X}_{Center} - \vec{X}_i(t)}{Dis_{i,Center}} \times \left[\vec{V}isual \times Rand(0,1)\right] \quad (5)$$

Equation (5) is used for all AFs that have worse positions than central position so they move toward X_{Center}. But for the best AF locating in X_{Best}, if fitness value of X_{Center} is better than X_{Best}, next position of the best AF is obtained from:

$$\vec{X}_{Best} = \vec{X}_{Center} \quad (6)$$

The reason of using equation (6) for the best AF is that it may be located in worse position than its current position by moving toward X_{Center} by using equation (5), because it is possible to have worse positions in the way ending to X_{Center} from X_{Best}. Therefore, it may cause to lose the best position found by all members of swarm so far. This problem is removed by using of (6) for the best AF. The reason of not using (6) for all AFs is that changing position of swarm fishes to a similar position leads to extreme decrease in diversity of swarm and considerable decrease of convergence rate.

IV. MAFSA PROCEDURE

In MAFSA, for every AF, prey, follow and swarm behaviors are performed in each iteration. In Standard AFSA executing one of the standard swarm and standard follow behaviors didn't affect on AFs movement and huge amount of computations were wasted.

Unlike standard AFSA, in MAFSA, all three behaviors influence on movement of AFs and swarm move toward better positions. In MAFSA, first, all AFs perform prey behavior and their position is updated based on this behavior procedure. By executing this behavior, every AF can displace up to *try_number* times. Then, all of them with respect to their new position and other AFs' position which performed prey behavior, execute follow behavior and all members except the best AF of swarm move to a new location in direction of moving toward the best found position by swarm. Then each AF performs swarm behavior. By performing swarm behavior, AFs which are apart from swarm and locate in worse positions than other swarm members, can join to the swarm faster. In fact, this behavior causes faster movement of AFs which are in worse positions. As a result of Performing this behavior, convergence rate would increase and unlike follow behavior it can causes position improvement of best AF.

At the end of each iteration of performing MAFSA algorithm, *Visual* value is updated for AFs. According to various experiments done on AFSA, it is concluded that big value of *Visual* parameter results in convergence rate increase since AFs move with larger steps. But when swarm converges, AFs are not able to do an acceptable local search because after convergence, space where has to be searched for better positions becomes smaller and AFs with large *Visual* by performing prey behavior would have less chance to reach better positions. On the other hand, if *Visual* is small, local search ability increases but because of small move steps, convergence rate decreases so much. Thus, it can conclude that the best condition is that at first *Visual* to be large so AFs to converge fast to their goals. Along with converging of swarm to goal, *Visual* value has to decrease gradually so at last AFs with small *Visual* could reach acceptable results by doing acceptable local search. For this reason, *Visual* value has to be multiplied by a number less than one in each iteration. We can use various mechanisms to determine this number. In this paper, a specified random number generator function is used as following:

$$Visual(t+1) = Visual(t) \times \left(L_{Low} + \left(Rand \times \left(L_{High} - L_{Low}\right)\right)\right) \quad (7)$$

In equation (5), *Visual* in each iteration is obtained randomly based on its value at previous iteration. L_{Low} and L_{High} are lower limit and upper limit of *Visual* change percentage in compare with previous iteration. *Rand* is the random number generator function with uniform distribution in [0 , 1]. So, *Visual* value in each iteration randomly is in $[Visual(t\text{-}1)\times L_{Low} , Visual(t\text{-}1)\times L_{High}]$. For this reason, L_{High} value should be considered less than one and $L_{Low} \leq L_{High}$.

MAFSA pseudo code is represented in figure 2. Here we suppose optimization problem is maximizing problem.

A. MAFSA configuration for dynamic environments

In this part, MAFSA algorithm will be configured for working in dynamic environments which is called dynamic MAFSA (DMAFSA). In MPB, when there is more than one peak in problem space, each peak can modify into global optima after environment change. Hence, all peaks have to be covered by AFs as much as possible so if each of them transforms into global optima, the algorithm could find it fast. Thus, one swarm should be located at each peak and cover it. So, multi-swarm has to be used in DMAFSA and each swarm act independently from other swarms and do according to MAFSA procedure. In this paper, at the beginning, there is only one swarm in problem space. If the swarm converges to a peak, another swarm is generated in problem space.

A swarm has converged when the best AF position is almost constant after some iteration in problem space. It means Euclidean distance of current position of the best AF from its position in some previous iteration to be less than a specified amount. In DMAFSA, whenever all swarms have converged, a new swarm generates in problem space and starts searching. If recently generated

swarm converges to a peak which is covered by another swarm, a race condition occurs. In this condition, the swarm which has better best AF according to fitness value continues its activity and the loser will expel and reinitialize. New generated swarm would converge to a covered peak by another swarm when Euclidean distance of best AF position of new generated swarm form best AF position in one of the other swarms in problem space is less than a specified value r_{excl}.

Thereafter, any new generated swarm has a chance to converge to a peak which no other swarm has resided there yet. But if it converges to a covered peak, with high probability it will be reinitialized. It's because the probability of being placed in a better position than previously residing swarm is very low. If new swarm converges to an empty peak which didn't cover by any swarm yet, another swarm will be generated in problem space. Therefore, the number of available swarms in problem space is always one more than the number of found peaks in problem space. This mechanism of generating new swarms is as like as [10] which was presented for PSO but with some modifications in it.

MAFSA:

for each *Artificial Fish i* □ *[1 .. N]*
  initialize x_i
end for
repeat*:*
  //Searching Food Behavior
for each *AF* i
  for counter=1 to *try_number*
    Obtain $\vec{X}_T$ with equation (1) and Calculate f($\vec{X}_T$)
    if $f(\vec{X}_T) \geq f(\vec{X}_i)$ **then**
      $\vec{X}_i = \vec{X}_T$
    end if
  end for
end for
// Follow Behavior
for each *AF* i
  Apply equation (3)
end for
// Swarm Behavior
Calculate *Central Position* $\vec{X}_{Center}$ by equation (4)
for each *AF* i
  if $f(\vec{X}_{Center}) \geq f(\vec{X}_i)$ **then**
    if $\vec{X}_i$ is Best AF
      Apply equation (6) for *AF* i
    else
      Apply equation (5) for *AF* i
    end if
  end if
end for
Update *Visual* according to equation(7)
until stopping criterion is met

Figure 2. MAFSA Pseudo Code

To discover change in environment, at the beginning of algorithm execution, a random point called *test point* is selected in problem space and its fitness value will be stored. After executing each iteration of algorithm for all active swarms, fitness value of test point is reevaluated. If obtained fitness value is not equal to stored value based on previous fitness evaluation of test point, then environment has been changed. It should be noted that test point position would remain fixed until the end of algorithm.

After detecting change in environment, first, problem of diversity reduction has to be resolved. Indeed, when a swarm converges to a point, the distance of AFs decreases very much from each other and diversity in swarm decreases too much. As a result, after environment change, AFs cannot take advantage of follow and swarm behaviors because distance of artificial fishes from each other has decreased too much. To resolve this issue, after detecting change in environment, the best artificial fish position of the swarm is kept for converged swarms and other AFs of that swarm are distributed randomly in a d-dimensional ball around the best AF position of the swarm. Radius of this d-dimensional ball in each dimension is determined in terms of shift length of peaks in MPB since it is expected that new peak position placed in a d-dimensional ball with the centered of previous peak position before environment change.

Also, *Visual* has decreased with respect to space where the swarm has to search for better positions by using equation (7). Thus, after detecting environment change, *Visual* value should be determined in terms of shift length. Consequently, new peak position would be in the searching neighborhood of AFs and they can move fast toward it. For swarm that has not converged, there is no need to change the position of AFs because diversity has not been lost but its *Visual* value must adjust equal to its initial value to avoid reducing the convergence rate. After adjusting *Visual* and determining AFs' position, their fitness values are reevaluated to remove outdated memory. Then, the algorithm continues searching in new environment.

In DMAFSA, to increase search ability around the best found peak by all swarms, *try_number* amount is considered for AFs of the best swarm more than other swarms. The best swarm is that one which its fitness value of the best AF is better than the other swarms' best AF fitness value. Therefore, AFs of the best swarm would have more opportunity to find better positions by performing prey behavior. The reason which *try_number* amount for all swarms is not increased, is that increasing of local search ability in local optimal peaks cannot affect on obtained result.

Also, by increasing *try_number* in all swarms, the number of fitness evaluations which AFs perform at each iteration increase. It causes that AFs perform less iterations until environment change and the algorithm adaptability becomes slow in comparison with environment change period. Thereafter, by increasing *try_number* just in the best swarm, search ability

increases around optimal peak and on the other side, the algorithm doesn't waste fitness evaluation. Pseudo code of the proposed algorithm DMAFSA is shown in figure 3.

Dynamic MAFSA:

```
//Initializing  first swarm
for each AF j in first_swarm
    initialize X_{1,j} randomly
end for
initialize Test_Point randomly
repeat
   for each  swarm
  //Searching Food Behavior
      for each AF j in swarm i
          for counter=1 to try_number
              obtain  X_T  by equation (1) and evaluate f( X_T )
              if  f( X_T ) ≥ f( X_{i,j} )  then
                  X_{i,j} =  X_T
              end if
          end for
      end for
  // Follow Behavior
      for each AF j in swarm i
          apply equation (3) based on best_AF in swarm i
      end for
 // Swarm Behavior
      Calculate  X_{Center,i}  according equation (4)
      for each AF j in swarm i
          if  f( X_{Center,i} ) ≥ f( X_{i,j} ) then
           if  X_{i,j}  is Best_AF in swarm i  then
              X_{i,j} =  X_{Center,i}
           else
              Apply equation (5) on  X_{i,j}
           end if
      end if
      Update Visual_i according equation (7)
end for
if All swarm are converged then
    Create new_swarm and initialize it
end if
//Exclusion
for each  swarm i
   if distance(bestAF_i and bestAF_{new_swarm}) <  r_excl then
      if f(bestAF_i) ≤ f(bestAF_{new_swarm}) then
          reinitialize Swarm i
      else
          reinitialize new_Swarm
      end if
 end for
//Test for Change
Evaluate Test_Point
if new value is different from last iteration then
     for each swarm i
        if swarm_i is converged
                    keep best_AF_i and randomize others around it
based on  Shift_length
         end if
           Set Visual_i based on Shift_ length
     end for
 end if
Until stopping criterion is met
```

Figure 3. Pseudo Code of Dynamic MAFSA.

I. EXPERIMENTAL RESULTS

To assess correctness and efficiency of the proposed algorithm, this algorithm was compared with various known algorithms on MPB which is one of the most famous benchmarks of dynamic environments [9,11]. Experiments were done with respect to MPB parameters which are given in table I.

TABLE I. MPB PARAMETERS SETTING

Parameter	Value
Number of peaks, M	Variable between 1 to 200
Change frequency	5000
Height change	7.0
Width change	1.0
Peaks shape	Cone
Basic function	No
Shift length, s	1.0
Number of dimensions,D	5
Correlation Coefficinet,λ	0
peaks location range	[0 – 100]
Peak height	[30.0 – 70.0]
Peak width	[1 – 12]
Initial value of peaks	50.0

The main metric for evaluating the performance of algorithms in this domain is offline error which is indicates the average of the best found position's fitness using algorithms during the running optimization process[9,11]. In other word, the value of offline error is the average of current errors. Current error at time t is the deviance of the best found position using algorithm at time t in the current environment and the position of global optimum in the current environment. The value of offline error is equal or greater than zero, where zero indicates the ideal situation.

Adjustment of parameters in the proposed algorithm is done according to various executions of experiments. Population size in each swarm is 2. *try_number* is equal to 20 for the best swarm and 2 for other swarms. *Visual* value for recently generated swarm is 20. After detecting environment change, *visual* value for converged swarms is considered equal to peaks shift length and AFs of these swarms are located randomly in a d-dimensional ball with radius of shift_length around the best AF of swarm. L_{Low} and L_{High} are adjusted 0.4 and 1 in equation (7), respectively. A swarm has converged, if Euclidean distance between the best AF position of this swarm and its position in 3 previous iterations is less than 0.1. Also, determining r_{excl} value according to [9] has appropriate results. Experiments were repeated independently 50 times and each time they were performed by different random seeds. Every experiment continued until environment changes 100 times. For example when change frequency was 5000, experiments performed 5×10^5 fitness evaluations and during this time the environment changes 100 times.

In table II, the proposed algorithm DMAFSA efficiency is compared with 10 other known algorithms: mQSO[9], mCPSO[9], Adaptive mQSO(AmQSO) [10],

CellularPSO[14], FMSO[15], RPSO[16], SPSO[17], rSPSO[18], mPSO[19], PSO-CP[20] with change frequency 5000, shift length equal 1 and different number of peaks in terms of offline error and Standard error. Given results in table II for some algorithms are brought from their respective paper directly. As it is observed, the proposed algorithm has better efficiency than other algorithms and only in the case that the problem has only one peak, the proposed algorithm is placed in second rank. In the proposed algorithm, because the number of swarms is proportional to the number of found peaks, the algorithm efficiency is acceptable both when number of peaks is low and when the number of peaks is high.

TABLE II. COMPARISON OF OFFLINE ERROR (STANDARD ERROR) OF 11 ALGORITHMS ON MPB PROBLEM WITH DIFFERENT NUMBER OF PEAKS.

ALG.	NUMBER OF PEAKS							
	1	**5**	**10**	**20**	**30**	**50**	**100**	**200**
mQSO	4.92(0.21)	1.82(0.08)	1.85(0.08)	2.48(0.09)	2.51(0.10)	2.53(0.08)	2.35(0.06)	2.24(0.05)
AmQSO	**0.51(0.04)**	1.01(0.09)	1.51(0.10)	2.00(0.15)	2.19(0.17)	2.43(0.13)	2.68(0.12)	2.62(0.10)
CLPSO	2.55(0.12)	1.68(0.11)	1.78(0.05)	2.61(0.07)	2.93(0.08)	3.26(0.08)	3.41(0.07)	3.40(0.06)
FMSO	3.44(0.11)	2.94(0.07)	3.11(0.06)	3.36(0.06)	3.28(0.05)	3.22(0.05)	3.06(0.04)	2.84(0.03)
RPSO	0.56(0.04)	12.22(0.76)	12.98(0.48)	12.79(0.54)	12.35(0.62)	11.34(0.29)	9.73(0.28)	8.90(0.19)
mCPSO	4.93(0.17)	2.07(0.08)	2.08(0.07)	2.64(0.07)	2.63(0.08)	2.65(0.06)	2.49(0.04)	2.44(0.04)
SPSO	2.64(0.10)	2.15(0.07)	2.51(0.09)	3.21(0.07)	3.64(0.07)	3.86(0.08)	4.01(0.07)	3.82(0.05)
rSPSO	1.42(0.06)	1.04(0.03)	1.50(0.08)	2.20(0.07)	2.62(0.07)	2.72(0.08)	2.93(0.06)	2.79(0.05)
mPSO	0.90(0.05)	1.21(0.12)	1.61(0.12)	2.05(0.08)	2.18(0.06)	2.34(0.06)	2.32(0.04)	2.34(0.03)
PSO-CP	3.41(0.06)	-	1.31(0.06)	-	2.02(0.07)	-	2.14(0.08)	2.04(0.07)
DMAFSA	0.55(0.06)	**0.78(0.06)**	**1.01(0.05)**	**1.42(0.06)**	**1.63(0.06)**	**1.84(0.07)**	**1.95(0.05)**	**1.99(0.04)**

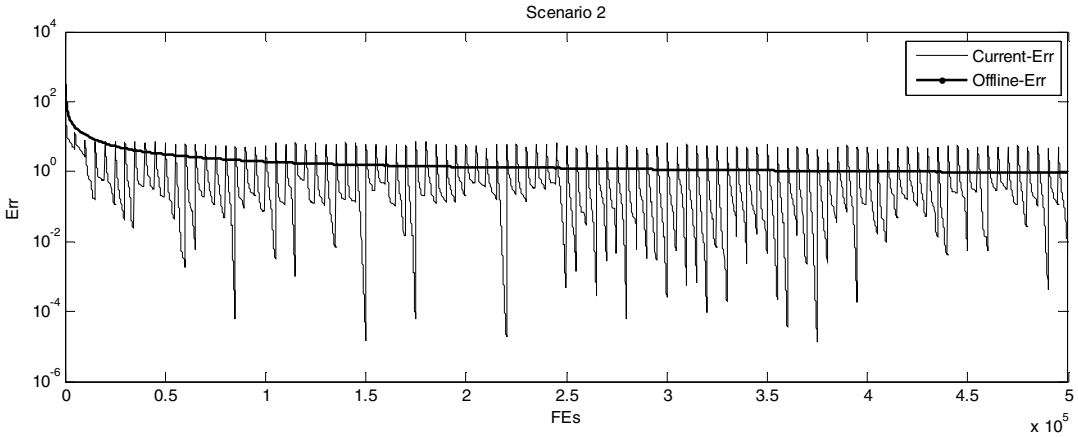

Figure 4. Offline Error and Current Error Obtained from the Proposed Algorithm DMAFSA.

In DMAFSA algorithm, parameters are set such that swarms which locate in local optimums perform fewer fitness evaluations in each iteration. Therefore, since the environment change criterion is number of fitness evaluations, there are more chances for the best swarm to seek better around the global optimum before environment change. In MAFSA designing, it tried to keep integrity of swarm by performing follow and swarm behaviors. On the other hand, with following the best AF of swarm by other swarm members, the algorithm convergence rate has increased much more. Moreover, in prey behavior, AFs can move some steps toward better positions with respect to local search and use all done fitness evaluations to improve their position. In addition, increase in local search ability with decrease in *visual* parameter cause AF to be able to reach more accurate results. All these improvements caused that swarms in the proposed algorithms could converge fast to goal and after reaching to goal, perform an acceptable local search.

In figure 4, offline error and current error graphs of the proposed algorithm DMAFSA are shown on MPB for 100 times of environment change with 5000 change frequency, 10 peaks and shift length equal to 1 which called scenario 2 in MPB problem. This graph is plotted with respect to mean of 50 times running the algorithm. As it is observed, after each time of environment change, the proposed algorithm follows the optimum well. The reason of high efficiency of this algorithm in looking after the optimums is adjustment of diversity and visual. As a result of this, new peak location is placed within the area of artificial fish swarm. Also, acceptable decrement amount of current error in any environment shows appropriate local search ability of swarm in proposed algorithm.

II. Conclusion

In this paper, for the first time, a method was proposed for optimization in dynamic environments based on artificial fish swarm algorithm. Results on moving peak benchmark problem were compared with some other known methods. In the proposed algorithm, it was tried to satisfy all requirements of dynamic environments. Experimental results showed that the proposed algorithm has high ability in locating and tracking optimums, appropriate convergence rate and local search ability.

Also, there are some relevant works to pursue in the future. First, some work can be done to set a good initialization before the algorithm start. Second, adaptation of *Visual* with some learning algorithm will also improve efficiency of the algorithm so much.

References

[1]. L. X. Li, et al., "An Optimizing Method based on Autonomous Animals: Fish Swarm Algorithm," presented at the Proc. of Systems Engineering Theory & Practice, 2002.

[2]. Y. Luo, et al., "The Optimization of PID Controller Parameters Based on Artificial Fish Swarm Algorithm," 2007, pp. 1058-1062.

[3]. M. Jiang and K. Zhu, "Multiobjective optimization by Artificial Fish Swarm Algorithm," presented at the IEEE international conference on computer science and automation engineering, 2011.

[4]. A. M. A. C. Rocha, et al., "An augmented Lagrangian fish swarm based method for global optimization," Journal of computational and applied mathematics, vol. 235, pp. 4611-4620, 2011.

[5]. H.-C. Tsai and Y.-H. Lin, "Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior," Applied Soft Computing, vol. In Press, Corrected Proof, 2011.

[6]. S. He, et al., "Fuzzy Clustering with Improved Artificial Fish Swarm Algorithm," 2009, pp. 317-321.

[7]. D. Yazdani, et al., "Color Quantization Using Modified Artificial Fish Swarm Algorithm", Australasian Conference on Artificial Intelligence, pp. 382-391, 2011.

[8]. J. Yaochu and J. Branke, "Evolutionary optimization in uncertain environments-a survey," Evolutionary Computation, IEEE Transactions on, vol. 9, pp. 303-317, 2005.

[9]. T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," Evolutionary Computation, IEEE Transactions on, vol. 10, pp. 459-472, 2006.

[10]. T. Blackwell, et al., "Particle swarms for dynamic optimization problems," Swarm Intelligence, pp. 193-217, 2008.

[11]. J. Branke. (1999). The Moving Peaks Benchmark. Available: http://people.aifb.kit.edu/jbr/MovPeaks/

[12]. J. Branke, Evolutionary Optimization in Dynamic Environments: Kluwer Academic Publishers, 2001.

[13]. M. Zhang, et al., "Mining Classification Rule with Artificial Fish Swarm," 2006, pp. 5877-5881.

[14]. A. B. Hashemi and M. R. Meybodi, "Cellular PSO: A PSO for Dynamic Environments," presented at the Proceedings of the 4th International Symposium on Advances in Computation and Intelligence, Huangshi, China, 2009.

[15]. C. Li and S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," 2008, pp. 624-628.

[16]. X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," Dimensions, vol. 1, pp. 2-2.

[17]. W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," Inf. Sci., vol. 178, pp. 3096-3109, 2008.

[18]. S. Bird and X. Li, "Using regression to improve local convergence," 2008, pp. 592-599.

[19]. M. Kamosi, et al., "A New Particle Swarm Optimization Algorithm for Dynamic Environments," 2010.

[20]. L. Liu, S. Yang and D. Wang, "Particle swarm optimization with composite particles in dynamic environments", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, pp. 1634-1648, 2010.