International Journal of Computational Intelligence and Applications
Vol. 15, No. 2 (2016) 1650010 (23 pages)
© World Scientific Publishing Europe Ltd.
DOI: 10.1142/S1469026816500103

A Novel Approach for Optimization in Dynamic Environments Based on Modified Artificial Fish Swarm Algorithm

Danial Yazdani
Young Researchers and Elite Club, Mashhad Branch
Islamic Azad University, Mashhad, Iran
d_yazdani@mshdiau.ac.ir

Alireza Sepas-Moghaddam*
Department of Electrical and Computer Engineering
Instituto Superior Técnico
Universidade de Lisboa Lisbon, Portugal
alireza.sepas-moghaddam@tecnico.ulisboa.pt;
sepasmoghaddam@gmail.com

Atabak Dehban
Institute for Systems and Robotics
Instituto Superior Técnico
Universidade de Lisboa Lisbon, Portugal
adehban@isr.ist.utl.pt

Nuno Horta
Instituto de Telecomunicações
Instituto Superior Técnico
Universidade de Lisboa Lisbon, Portugal
nuno.horta@lx.it.pt

Received 13 December 2014
Accepted 07 March 2016
Published 8 June 2016

Swarm intelligence algorithms are amongst the most efficient approaches toward solving optimization problems. Up to now, most of swarm intelligence approaches have been proposed for optimization in static environments. However, numerous real-world problems are dynamic which could not be solved using static approaches. In this paper, a novel approach based on artificial fish swarm algorithm (AFSA) has been proposed for optimization in dynamic environments in which changes in the problem space occur in discrete intervals. The proposed algorithm can quickly find the peaks in the problem space and track them after an environment change. In this algorithm, artificial fish swarms are responsible for finding and tracking peaks and several behaviors and mechanisms are employed to cope with the dynamic environment.

*Corresponding author.

Extensive experiments show that the proposed algorithm significantly outperforms previous algorithms in most of tested dynamic environments modeled by moving peaks benchmark.

Keywords: Artificial fish swarm algorithm; dynamic optimization problems; swarm intelligence; evolutionary algorithms; moving peaks benchmark.

1. Introduction

The study of applying evolutionary algorithms for optimization in dynamic environments is an active research topic and has increasingly attracted interest from the evolutionary computation community. In Ref. 1, Nguyen defined dynamic optimization problems as follows: "Given a dynamic problem f_t, an optimization algorithm G to solve f_t, and a given optimization period $[t^{\text{begin}}; t^{\text{end}}]$, f_t is called a dynamic optimization problem (DOP) in the period $[t^{\text{begin}}; t^{\text{end}}]$ if during $[t^{\text{begin}}; t^{\text{end}}]$ the underlying fitness landscape that G uses to represent f_t changes and G has to react to this change by providing new optimal solutions." The most prominent aim in static optimization problems is finding the global optimum. However, in DOPs, tracking the global optimum should be also considered.

So far, different optimization approaches have been proposed, including swarm intelligence methods for optimization in dynamic environments.[2,3] The designed algorithms based on swarm intelligence approaches include some mechanisms in their structure to solve the particular challenges faced in dynamic environments. Due to the lack of appropriate time between the occurrences of two consecutive environment changes, which contribute to further complications of DOPs, the need for powerful and efficient optimization techniques is imminent.

Optimization algorithms which are proposed to be performed in dynamic environments are typically extended versions of those in static environments. For instance, evolutionary algorithms,[4,5] particle swarm optimization (PSO),[2,6] ant colony optimization,[7,8] differential evolution[9,10] and artificial fish swarm algorithm (AFSA)[11] can be mentioned.

There are various types of optimization problems in real-world environments in which different challenges are involved. In fact, designing efficient optimization problems in dynamic environments is dependent on the particular challenges of the problem. In this paper, a novel algorithm based on AFSA has been proposed for optimization in dynamic environments that has been modeled by moving peaks benchmark (MPB).[12,13] There are some assumptions regarding the proposed approach as follows: (1) The approach is applied to unconstrained multi-modal problems (2) The problem space is continuous (3) The changes in the space take place in a discrete in time (4) The dimension and domain of the search space are constant after environment changes. In the proposed algorithm, various mechanisms have been employed in order to solve particular challenges of dynamic environments. Several mechanisms are novel and some of them are the extended versions of the previously used mechanisms.

The rest of this paper is organized as follows: In Sec. 2, related work is discussed. Section 3 dedicated to the proposed algorithm. Section 4 the results of the extensive experiments of the proposed algorithm and its comparison with other approaches from the literature. Finally, Sec. 5 concludes this paper.

2. Related Work

Multi-swarm is considered as a well-known solution for designing DOPs. Several algorithms for DOPs based on multi-swarm approach have been proposed in the literature. In Ref. 14, a method called shifting balance genetic algorithm (SBGA) has been proposed in which a number of small subpopulations were responsible for global search in the problem space, and a large subpopulation was responsible for tracking the peaks. Another approach was presented in Ref. 15, called self-organizing scouts (SOS), which utilized a big subpopulation for global search and a number of small subpopulations for tracking changes. This strategy has also been proposed with other meta-heuristic methods such as genetic algorithm in Ref. 16 and differential evolution in Ref. 17. In Refs. 18 and 19, two methods similar to SOS were proposed, respectively called fast multi-swarm optimization (FMSO) and multi-swarm PSO (mPSO), in which a parent type explored the search space to discover existing promising areas in the environment, and a series of child types performed local search. Another approach was to use a population for both local search and global search simultaneously. In Ref. 20, a population was used for performing global search, and after discovering an optimum, the population was divided into two subpopulations. The first and second subpopulations were responsible to track optimum changes and conducting global search, respectively. In Refs. 21–23 a speciation-based PSO (SPSO) approach was proposed for optimization in dynamic environments. Also in Ref. 24, a regression-based PSO approach (RSPSO) was presented in order to enhance the convergence rate using speciation-based methods. In that approach, every subpopulation was considered as a hypersphere and was developed through a certain radius of the best solution. In Ref. 25, a method called SPSO was proposed in which every cluster was divided into two. The first cluster was responsible for exploitation and the second one was in charge of exploration. In that research, Gaussian local search and differential mutation have been used in order to improve diversity in the environment. In Ref. 26, a method based on clustering was proposed for developing subpopulations, and in Ref. 27, this method PSO with composite particle (PSO-CP) has been improved, in which some simplifications, e.g., eliminating the learning procedure, and reducing the number of phases for clustering from two phases to only one phase was made. In Ref. 28, two multi-population methods, called multi-sawarm optimization (mQSO) and multi-changed particle swarm optimization (mCPSO), were proposed. In the former one, quantum particles and in the latter one, charged particles were used to generate diversity. The number of solutions in this technique was equal for every subpopulation, and the number of subpopulations was also initialized. In Ref. 29, an approach for enhancing this method was proposed by adapting the number of subpopulations, which was called AmQSO. It has significantly improved the performance of the algorithm. Finally, a method for optimization in dynamic environments was proposed based on composite particles in Ref. 30 which demonstrated a suitable efficiency.

The AFSA is one of the algorithms inspired from the nature and swarm intelligence algorithms.[31] This algorithm 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 used in optimization applications such as neural network learning,[27,32] color quantization,[33] global optimization,[34,35] data clustering,[36–38] multi-objective optimization,[39] PID controller parameters optimization,[40] image segmentation,[41,42] etc. A multi-swarm algorithm (mNAFSA)[43] was proposed to conquer particular challenges of dynamic environment by proposing modified multi-swarm mechanism for finding and covering potential optimum peaks. In Ref. 11, Yazdani *et al.* proposed a modified AFSA (MAFSA) for designing optimization algorithms in dynamic environments. In the MAFSA algorithm, parameters, behaviors and the standard AFSA procedure were modified to be appropriate for optimization in dynamic environments. In this algorithm, several behaviors were performed on artificial fish (AF).

In Ref. 44, the idea of hibernation was applied in a PSO optimization algorithm, in which a parent swarm explores the search space and child swarms exploit promising areas found by the parent swarm. In Ref. 45, a new PSO algorithm for dynamic environments was proposed to adapt exclusion radios and utilize a local search on best swarm to accelerate progress of algorithm and adjust inertia weight adaptively. Cellular PSO,[46] a new hybrid model of PSO and cellular automata, was proposed to find global optima quickly after the change in environment. PSO-CP[47] proposed to address dynamic optimization problems by partitioning the swarm into a set of composite particles based on their similarity. In Ref. 48 a new technique was presented that can be used with most evolutionary algorithms that improve their convergence speed. In Ref. 49, a new multi-strategy ensemble PSO (MEPSO) for dynamic optimization was proposed that included two new strategies, Gaussian local search and differential mutation. Woldesenbet *et al.*[50] proposed a new dynamic evolutionary algorithm that uses variable relocation to adapt already converged or currently evolving individuals to the new environmental condition. In Ref. 51, an algorithm based on firefly algorithm was proposed for multi-modal optimization in dynamic environment. CDEPSO[52] was a bi-population hybrid collaborative model of crowding-based differential evolution and PSO for dynamic optimization problems. In Ref. 53, a novel multi-swarm cellular PSO algorithm was proposed by clustering and local search, where the search space was partitioned into cells and a local search is applied to improve the solutions in the each cell. Yazdani *et al.*[54] proposed a novel algorithm for optimization in dynamic environments based on PSO in which a novel mechanism has been used to increase the ability of local search around optimum with focusing on best found peak in each environment. In Ref. 55, a speciation-based firefly algorithm was investigated to enhance the population diversity in order to generate several populations in different areas in the landscape without knowing the number of optima in each landscape.

3. The Proposed Algorithm: Parent–Child AFSA

In this section, the proposed algorithm for optimization in dynamic environments is presented. The proposed approach presented in this paper is a modified version of MAFSA algorithm,[11] where we added some new mechanisms to MAFSA in order to overcome the particular challenges of dynamic environments and improve its performance. There is only one type of swarm in MAFSA, where it initiates another swarm after its convergence. However, in parent–child AFSA (PCAFSA), the swarms have been divided into parent, non-best child, and best child swarms with different configurations. Parent swarms are responsible for finding undiscovered peaks in an appropriate time, where child swarms cover the peaks and subsequently, track them after an environment change. The first added mechanism to MAFSA is migration to increase the speed of finding undiscovered peaks. Born mechanism is the second additional mechanism to MAFSA for solving the challenges of existing potential optimums and unknown number of peaks in the problem space. The last added mechanism is exclusion which is proposed to solve the challenge of convergence of two swarms to one peak. In addition to the added mechanisms, some modifications have been performed in the structure of MAFSA to solve the challenges of diversity loss and outdated memory.

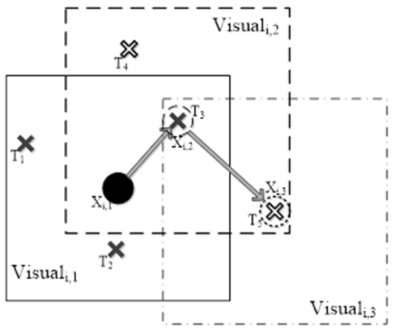
Since the proposed algorithm utilizes parent–child mechanisms, it is called PCAFSA. In what follows, the behaviors, mechanisms and procedure used in the PCAFSA algorithm are discussed in detail to solve challenges in dynamic environments. In this algorithm, prey, follow and swarm behaviors are performed on AF and several mechanisms are employed to face particular challenges of dynamic environments.

3.1. *Prey behavior*

This behavior is an individual behavior, where each AF does a local search around itself without considering other swarm members. By performing this behavior, each AF attempts *try_number* times to replace to a new position with a better fit. Suppose AF_i is in position X_i and wants to display prey behavior. The following steps are performed in prey behavior:

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

$$T_d = X_{i,d} + \mathrm{Visual} \times \mathrm{Rand}_d(-1, 1). \tag{1}$$



(a)

Prey behavior ()

```
for each AF i
    for counter=1 to try_number
        Obtain $\vec{X}_T$ with Eq. (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
```

(b)

Fig. 1. (a) The schematic of prey behavior and (b) pseudo-code of prey behavior.

(b) If the fitness value of position X_T is better than the current position of AF$_i$, the position will be updated by Eq. (2):

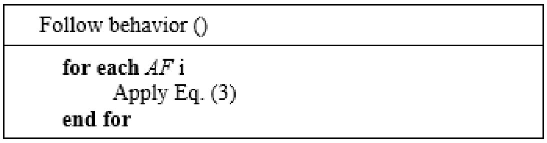
$$\mathbf{X}_i = \mathbf{T}. \tag{2}$$

Steps (a) and (b) are performed *try_number* times. By executing the above steps, an AF can update its position at most *try_number* times in the best case and move toward better positions. In the worst case, none of the AF's attempts to find a better position will succeed. In this situation, after performing the prey behavior, there will be no replacement at all. The schematic of prey behavior for ith AF in two-dimensional space is shown in Fig. 1(a).

Visual space is a D-dimensional cubic space in which ith AF performs a search process by Eq. (1). As can be seen, ith AF which is placed in X$_{i,1}$ position finds a better position in the third execution of Eq. (1) and moves to this position by Eq. (2). Again, the AF performs this process from its new position (X$_{i,2}$). This procedure is performed up to *try number* times. In Fig. 1(a), it is considered that *try number* is equal to 5 and the AF improved its position two times. Pseudo-code of prey behavior is shown in Fig. 1(b).

3.2. *Follow behavior*

In standard AFSA, in case of not finding better positions by standard prey behavior, AFs move one step randomly and so lose their previous positions. But in PCAFSA, if an AF is not able to move to better positions in prey behavior, it will not move at all and will keep its previous position. This causes the best AF (according to the fitness value) of the swarm to be located in the best position found by the swarm member so far. The reason is that in prey behavior in the proposed algorithm, an AF displaces if only it moves to a better position. In the following behavior, each of AFs moves one



```
Follow behavior ()
    for each $AF$ i
        Apply Eq. (3)
    end for
```

Fig. 2. Pseudo-code of follow behavior.

step toward the best AF of swarm using Eq. (3):

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

where X_i is the position vector of AF_i which performs the follow behavior and X_{Best} is the position vector of the best AF in the swarm. Therefore, AF_i can move at most as much as its visual value in each dimension towards the best AF of the swarm. In fact, after finding more food by a fish, other swarm members follow it to reach more food. Following the best AF of the swarm makes the convergence rate increase and helps to keep the integrity of AFs in a swarm. This behavior is a group behavior and interactions among swarm members take place globally. Pseudo-code of follow behavior is shown in Fig. 2.

3.3. *Swarm behavior*

This function is also a group behavior and is performed globally among members of the swarm. In swarm behavior, first of all, the central position of the swarm is calculated in terms of the arithmetic average of the positions of all swarm members in every dimension. The *central position* of the swarm is obtained by Eq. (4):

$$X_{\text{Center},d} = \frac{1}{N} \sum_{i=1}^{N} X_{i,d}, \tag{4}$$

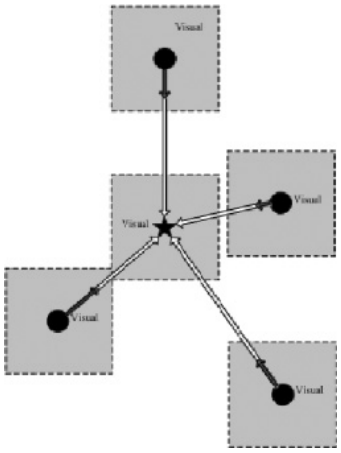
where N is equal to the *population size*. As it is observed, component d of vector X_{center} is the arithmetic mean of component d of all AFs of the swarm. For AF_i, the move condition toward the central position is checked, i.e., $f(X_{\text{Center}}) \geq f(X_i)$ and if this condition is satisfied, the next position of AF_i is obtained by Eq. (5):

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + \frac{\mathbf{X}_{\text{Center}} - \mathbf{X}_i(t)}{\text{Dis}_{i,\text{Center}}} \times [\mathbf{Visual} \times \text{Rand}(0,1)]. \tag{5}$$

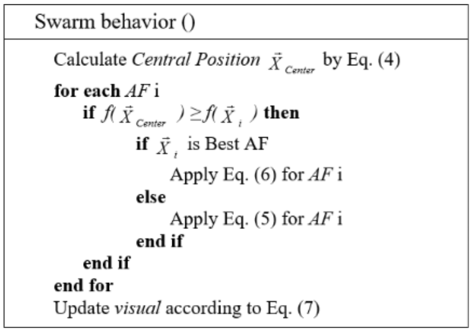
Equation (5) is used for all AFs that have positions worser than the central position, so they move towards X_{Center}. For the best AF located in X_{Best}, if the fitness value of X_{Center} is better than X_{Best}, the next position of the best AF is obtained by (6):

$$\mathbf{X}_{\text{Best}} = \mathbf{X}_{\text{Center}}. \tag{6}$$

The reason for using Eq. (6) for the best AF is that it may be located in a position worse than its current position by moving toward X_{Center} using Eq. (5), because it



(a)



```
Swarm behavior ()
Calculate Central Position $\vec{X}_{Center}$ by Eq. (4)
for each AF i
    if $f(\vec{X}_{Center}) \geq f(\vec{X}_i)$ then
        if $\vec{X}_i$ is Best AF
            Apply Eq. (6) for AF i
        else
            Apply Eq. (5) for AF i
        end if
    end if
end for
Update visual according to Eq. (7)
```

(b)

Fig. 3. (a) The schematic of swarm behavior (b) Pseudo-code of swarm behavior.

is possible to have worse positions in the way ending in X_{Center} from X_{Best}. Therefore, it may cause to lose the best position found by all members of the swarm so far. This problem is sorted out by using Eq. (6) for the best AF. The reason for not using Eq. (6) for all AFs is that changing the position of swarm fishes to a similar position leads to an extreme decrease in the diversity of the swarm and a considerable decrease in convergence rate. The schematic of swarm behavior is shown in Fig. 3(a). In this figure, the central position and the best AF are illustrated, respectively, by a plus and a star. As can be seen, the best AF is directly placed in the central position using Eq. (6), in case of a better situation for the central position, whereas other AFs move one step in their visual toward the central position using Eq. (5).

PCAFSA performs the optimization process iteratively using the functions that were explained as its behavior. Algorithm agents which are also called AFs, in a similar manner of standard AFSA, try to move toward better solutions in the problem space in each iteration. At the end of each iteration of swarm behavior in PCAFSA, the visual value is updated for AFs. In this paper, a specified random number generator function is used in Eq. (7) to determine the value of visual.

$$\text{Visual}(t+1) = \text{Visual}(t) \times (L_{\min} + (\text{Rand} \times (1 - L_{\min}))), \tag{7}$$

where visual is obtained randomly in each iteration based on its value at the previous iteration. L_{Low} is the lower limit of visual change percentage in comparison with the previous iteration. Rand is the random number generator function with uniform distribution in [0, 1]. So, visual value in each iteration is in [visual$(t-1) \times L_{\min}$, visual$(t-1)$] randomly. Pseudo-code of swarm behavior is shown in Fig. 3(b).

3.4. *Solving the challenge of change detection in an environment*

One of the challenges that optimization algorithms in dynamic environments encounter is detecting an environment change. In fact, the change must be detected without any prior knowledge after the environment change. The designed mechanisms in this domain are completely dependent on the range of changes.

In PCAFSA, to discover changes in the environment, the best AF of each swarm must be evaluated at the end of each iteration. In case that any changes in the obtained values, compared to the stored ones, the environment has been changed.

3.5. *Solving diversity loss challenge*

Regarding the structure of the proposed algorithm, diversity loss occurs after convergence. In this situation, all AFs are placed close to each other and visual also decreases extremely after a while. To solve this problem, the diversity loss is initially allowed to occur with the aim of increasing the accuracy of the result, before an environment change. After detecting a change in the environment diversity is created amongst the AFs for increasing convergence speed toward the new position of the goal. For increasing diversity between AFs, the position of the best AF in the swarm is kept and other AFs are randomly distributed around it with a uniform distribution in a space with a radius of r_{div} in each dimension, using Eq. (8):

$$X_{i,j} = X_{\text{Best},j} + (\text{rand}(-1,1) \times r_{\text{div}}), \tag{8}$$

where, jth component of the ith AF in the swarm is randomly calculated regarding the jth position of the best AF in the swarm (X_{best}) and r_{div} parameter. The Rand function generates a random number in the range of [−1,1] with a uniform distribution. After detecting the positions of AFs, the visual value is also adjusted for AFs to search a bigger space around itself by displaying prey behavior and move toward the new position with longer steps.

3.6. *Solving outdated memory challenge*

As it was mentioned, the position of the best AF in the swarm remains unchanged and the positions of the other AFs are randomly determined after detecting an environment change. Then, the fitness of all AFs is evaluated and their fitness values in the new environment are stored in the memory. Thus, the stored fitness values in the memory are valid.

3.7. *Born mechanism: solving the challenges of existing potential optimums and unknown number of peaks*

As it was stated before, there are several peaks in dynamic environments, where each peak could be transformed to a global optimum; therefore, each peak is a potential optimum. Thus, the algorithm must cover all peaks to find the optimum peak in an appropriate time after each environment change. The Born mechanism is utilized in the proposed algorithm in order to cover the peaks. Hence, there are several AF swarms in the problem space which are executed simultaneously and independently for performing the optimization process.

Each swarm in PCAFSA is responsible for one peak in the problem space. The mechanism used in this algorithm for controlling swarms is in the form of parent–child, in which there are different swarms as parents and children. *Parent* swarms are responsible for finding peaks and their parameters are adjusted in a way that they can find the peaks quickly. On the other hand, *child* swarms cover a peak and subsequently track it, and their parameters are adjusted accordingly. In the beginning of the algorithm, only *parent* swarms exist in the search space and there is no *child* swarm in the problem space. At first, the AFs of *parent* swarms are randomly initialized and start the searching process. After convergence of a *parent* swarm to a peak, it generates a *child* swarm and replaces the *child* swarm with itself on the peak. If the Euclidean distances between the position of the best AF of a *parent* swarm in the mth iteration and $m+n$th iteration are less than a threshold called r_{conv}, a *parent* swarm has converged to a peak which means a peak is found. After creation and replacement of a *child* swarm in the peak found by the *parent*, the *child* swarm must cover the peak and track it after environment changes. The *child* swarm is also responsible for exploitation. After placing the new *child* swarm in the peak, the *parent* swarm is re-initialized in the environment and starts a search to find a new peak. The processes of generation and placement of a new *child* swarm and re-initialization of a *parent* swarm are carried out after each convergence of the *parent* swarm to a new peak. *Parent* swarms in the environment search for the peaks which have not been previously found and once they find a new peak, it is covered by a *child* swarm. Consequently, the number of the existing swarms in the environment corresponds to the found peaks and it is expected that all of the peaks be covered by the *child* swarms after some time. Hence, *parent* swarms solve the challenge of unknown number of peaks by finding uncovered peaks and generating *child* swarms to cover them. In addition, by placing *child* swarms on the peaks, the algorithm monitors the peaks and rapidly finds the peaks which have been transformed to a global optimum, after an environment change. Thus, the unknown number of peaks challenge is also addressed. The pseudo-code for generating *child* swarms by the converged *parent* swarms in Born mechanism is demonstrated in Fig. 4(a).

3.8. *Exclusion mechanism: solving the challenge of convergence of two swarms to one peak*

In the proposed method, it is possible that a *parent* swarm converges to a peak to which it has previously converged and a *child* swarm has been placed on it. In this situation, the algorithm is re-initialized. In fact, the peaks which have been found previously are covered by a *child* swarm. Thus, if the Euclidean distance of the best AF of a *child* swarm and the best AF of a *child* swam is less than a threshold which is called r_{excl}, the *parent* swarm converges to a peak which was previously found. The

```
Born():
if Dis($X_{\text{Parent,j},\text{BestAF}}(m)$ , $X_{\text{Parent,j},\text{BestAF}}(m+k)$) < $r_{\text{conv}}$ then
    Create new Child_Swarm i
    foreach AF k in Child_Swarm i
        $X_{\text{i,k}}=X_{\text{Parent,j}}$
    endfor
    $Visual_{\text{i}} = Visual_{\text{Parent,j}}$
    Randomize all AF in Parent_swarm j
    Reinitialize $Visual_{\text{Parent,j}}$
endif
```

(a)

```
Exclusion()
foreach child_swarm i
    foreach child_swarm j
        if Exclusion are enable for child_swarm i & j and distance($bestAF_{\text{i}}$ and $bestAF_{\text{j}}$) < $r_{\text{excl}}$ then
            if f($bestAF_{\text{i}}$) ≤ f($bestAF_{\text{j}}$) then
                kill child_Swarm i
            else
                kill child_Swarm j
            endif
        endif
    end for
end for
foreach child_swarm i
    foreach parent_swarm j
        if distance($bestAF_{child,\text{i}}$ and $bestAF_{\text{Parent,j}}$) < $r_{\text{excl}}$ then
            re-initialize Parent_swarm j
        endif
    endfor
endfor
```

(b)

Fig. 4. Pseudo-codes of (a) Born mechanism and (b) exclusion mechanism.

value of r_{excl} in the proposed algorithm is determined by Eq. (2). Thus, by performing each iteration of the algorithm, Euclidean distances between the best AF of each *parent* swarm and the best AF of all *child* swarms are calculated in the problem space and in case the distance is less than r_{excl}, the *parent* swarm is re-initialized.

In addition, it is possible that a *parent* swarm converges before reaching a peak; hence, it generates a *child* swarm to be replaced in the position. Thus, the *child* swarm could move toward a peak which is already covered by another *child* swarm. In this situation, there are two *child* swarms in one peak. For solving this challenge, Euclidean distances between the best AF of all *child* swarms are calculated at first and two swarms amongst AFs for which the Euclidean distances are less than r_{excl}, are selected as the swarms that are placed in a particular peak. In this situation, the swarm in which the fitness of the best AF is worse, compared to another swarm, is eliminated. The mechanism used for solving this challenge is called exclusion. The pseudo-code of the exclusion mechanism is demonstrated in Fig. 4(b).

3.9. *Migration mechanism: solving the convergence speed challenge*

One of the most important challenges in this domain is the short time between two successive environment changes. This limitation leads to several problems, i.e., changing optimum positions that are not already found, losing goals after several environment changes and thus, increasing in errors and degrading efficiency.

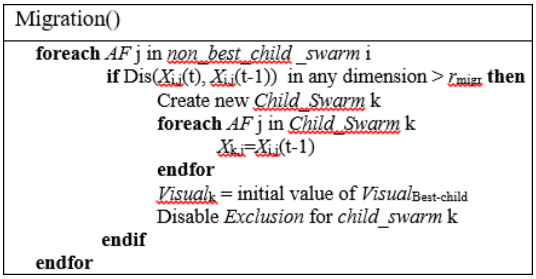
PCAFSA also benefited from a suitable convergence speed. Increasing diversity is another approach for increasing the convergence speed of the proposed algorithm. Distributing AFs around the position of the best AF in the swarm and re-adjusting visual values lead to covering the new optimum position with a high probability and finding it by the swarm with a high speed. *Parent* swarm is utilized in order to increase the speed of finding and covering peaks. Adjusting parameters in this swarm is appropriately performed for finding the peaks with a higher speed. On the other hand, adjusting the parameters of *child* swarm is performed regarding their tasks. The reason for using two types of swarms in PCAFSA is the fact that adjusting the parameters of *parent* and *child* swarms with equal values leads to a decrease in the convergence speed of PCAFSA and consequently, degrading the algorithm efficiency. Thus, by using two types of swarms, each swarm could be independently adjusted to perform its own task with a higher speed.

After a change in the environment, each *child* swarm tracks the peak in which it has been placed by performing a local search. This issue is more important for the swarm placed in the highest peak than that for the other peaks. In fact, the current error value and the efficiency of the algorithm are calculated based on the swarm whose fitness value of its best AF is better, compared to other swarms. As a result, other *child* swarms have no effect on determining results in the current environment. However, it is important to perform a local search for them. Indeed, if these swarms do not perform a local search, their distances from their corresponding peaks are high or in some cases they may lose it, after several environment changes. Thus, performing local search for all *child* swarms is mandatory after an environment change.

After a swarm approaches its goal by performing a local search, the search continues for improving the accuracy of the results. As it was mentioned, the *child* swarm placed in the highest peak determines the results; therefore, this swarm must perform a more precise local search to improve the results. For this purpose, the value of *try-number* corresponding to this swarm is considered greater, compared to other swarms. Thus, AFs of the best swarm perform more searches in each iteration for reaching better positions by using prey behavior, and so the local search ability and convergence speed are enhanced.

After each environment change, the visual parameter values of *child* swarms are reset in order to increase diversity. Determining the value of this parameter after detecting an environment change is considerably important for algorithm efficiency. After an environment change, the *child* swarms are divided into two swarms: best and non-best. The best swarm is a swarm whose best AF fitness is better, compared to other *child* swarms. The current error value is determined by the best swarm in the environment; hence, this swarm must converge to the optimum with a high speed. For this purpose, the visual value of this swarm must correspond to the maximum movement step of peaks. On the other hand, the visual value of other *child* swarms would be determined large, regarding the length of the problem space, so that a *child* swarm migrates from a peak to a better one. This situation occurs when the adjacent peak that is in the range of the AFs visual is better than the peak in which the related swarm is placed. Therefore, *child* swarms could support the *parent* swarms, in terms of finding better peaks. This support is more effective, when *parent* swarms have not yet found all of the peaks. Thus, *child* swarms could cover better peaks.

Migration takes place only once for a swarm in each environment. In case of moving AFs of the migrating swarm with steps greater than r_{migr}, this swarm could not generate another swarm. In addition, the exclusion mechanism between the swarm which has migrated in the current environment and the replaced new swarm is not activated until the next environment change. The pseudo-code of the migration mechanism is illustrated in Fig. 5.



```
Migration()
foreach AF j in non_best_child _swarm i
    if Dis($X_{i,j}$(t), $X_{i,j}$(t-1)) in any dimension > $r_{migr}$ then
        Create new Child_Swarm k
        foreach AF j in Child_Swarm k
            $X_{k,j}$=$X_{i,j}$(t-1)
        endfor
        $Visual_k$ = initial value of $Visual_{Best-child}$
        Disable Exclusion for child_swarm k
    endif
endfor
```

Fig. 5. Pseudo-code of migration mechanism.

3.10. *PCAFSA*: *main procedure*

Finally, after describing all modules of PCAFSA, the pseudo-code of the main procedure is presented in Fig. 6.

4. Experimental Study

In this section, the efficiency of the proposed method on MPB is surveyed. At first, MPB is described. After that, the effects of various values of the proposed algorithm parameters are studied and subsequently, its efficiency is compared with several state-of-the-art algorithms in this domain.

4.1. *Moving peaks benchmark*

In our simulations, we used the MPB originally introduced by Branke[12,13] as a benchmark function. It is extensively used in the literature for evaluating the performance of optimization algorithms in dynamic environments. In this benchmark, there are some peaks in a multi-dimensional space, where height, width and position of the peaks vary when a change occurs in the environment.

In order to measure the efficiency of the algorithms, *offline error* is used, which is the average of the difference between fitness of the best solution found by the algorithm and the fitness of the global optimum[5,12,13]:

$$\text{offline error} = \frac{1}{\text{FEs}} \sum_{t=1}^{\text{FEs}} (\text{fitness}(\text{gbest}(t)) - \text{fitness}(\text{globalOptimum}(t))) \qquad (9)$$

where FEs is the maximum fitness evaluation, and $g\text{best}(t)$ and $\text{globalOptimum}(t)$ are the best positions found by the algorithm and the global optimum at the tth fitness evaluation, respectively. In other words, the value of offline error equals the average of all current errors which is defined in time t as the deviance between the best position found by the algorithm in time t in the current environment and the position of the global optimum in the current environments.

```
PCAFSA
//Initializing  Parent swarm
foreach AF j in Parent_swarm
    initialize $X_{\mathrm{Parent,j}}$ randomly
end for
repeat
 //Parent Execution
    foreach parent_swarm j
       foreach Artificial Fish i ϵ [1 .. N]
          initialize xi
       end for
       repeat:
          Prey Behavior ()
          Follow Behavior ()
          Swarm Behavior ()
       until stopping criterion is met
       Born();
    Endfor

 //Child Execution
    foreach  child_swarm i
       if child_swarm i is Best_child_swarm then
            foreach best_child_swarm Artificial Fish i ϵ [1 .. N1]
               initialize xi
            end for
            repeat:
               Prey Behavior ()
               Follow Behavior ()
               Swarm Behavior ()
            until stopping criterion is met
       else
            foreach best_child_swarm Artificial Fish i ϵ [1 .. N2]
               initialize xi
            end for
            repeat:
               Prey Behavior ()
               Follow Behavior ()
               Swarm Behavior ()
            until stopping criterion is met
            Migration();
       endif
    endfor
    Exclusion();
//Test for Change
    Evaluate Best_AF in each swarm
    if new values are different from saved values then
       re-evaluate all AF in all Parent_swarm
       enable Exclusion for all child_swarm
       foreach child_swarm i
          keep $best\_AF_i$ and randomize other $AF_i$s around it based on $r_{\mathrm{div}}$
       endfor
       determaine Best_child_swarm
       Set $Visual_{\text{Best-child-swarm}}$ based on Shift_ length
       Set $Visual_{\text{non-best-child-swarm}}$ based on search space range
   end if
Until stopping criterion is met
```

Fig. 6. Pseudo-code of PCAFSA.

4.2. Parameter settings

The efficiency of the proposed algorithm using different numbers of *parent* swarms is presented after adjusting the parameters and surveying their effects on the efficiency of the algorithm. In PCAFSA, a *parent* swarm initiates the optimization process. Thus, the efficiency of this swarm is considerably important. In what follows, the effects of different MAFSA configurations on the efficiency of the *parent* swarm are studied.

4.2.1. *Parameter settings for parent swarms*

Parent swarms are responsible in terms of finding the peaks in the problem space. Therefore, the PCAFSA parameters should be defined in such a way that *parent* swarms can converge toward the peaks with a high speed. There are four parameters including: visual, $L_{\min}$, *try-number* and *population size* in MAFSA which define its convergence behavior. The parameters *Try-number* and *population size* depend to each other. Hence, they could not be investigated independently. These parameters determine the search strategy as well as the level of fitness evaluation in each iteration. The value of *try-number* determines the volume of local search around each AF and the size of the population shows the number of AF positions around which the search is performed. In addition, visual and $L_{\min}$ parameters are dependent on each other. The effects of different values of *population size* and *try-number* parameters on the *parent* efficiency are tabulated in Table 1. It is worth mentioning that to perform the experiments presented in Table 1, the value of visual and $L_{\min}$ are, respectively, 20 and 0.8 by default. Also, experiments are performed 100 times to find a peak in MBP for up to 2500 fitness evaluations. It is evident in the course of experiments that

Table 1. Effect of different values of population size and try number parameters on the parent efficiency.

(Population size, Try number)	Final error ± Standard error	(Population size, Try number)	Final error ± Standard error
(2,2)	5.8275 ± 2.6851	(4,2)	1.6048 ± 0.2905
(2,3)	2.4130 ± 1.3661	(4,3)	$3.64\text{e}{-05} \pm 3.88\text{e}{-06}$
(2,4)	$2.42\text{e}{-09} \pm 3.21\text{e}{-10}$	(4,4)	$0.0003 \pm 4.98\text{e}{-05}$
(2,5)	$9.32\text{e}{-08} \pm 1.71\text{e}{-08}$	(4,5)	0.0023 ± 0.0002
(2,7)	$1.14\text{e}{-05} \pm 2.13\text{e}{-06}$	(4,7)	0.0267 ± 0.0036
(2,10)	$0.0006 \pm 7.72\text{e}{-05}$	(4,10)	0.1709 ± 0.0186
(2,15)	0.0143 ± 0.0013	(4,15)	1.0893 ± 0.1108
(2,20)	0.1174 ± 0.0110	(4,20)	2.2069 ± 0.1807
(3,2)	7.7404 ± 4.0610	(5,2)	0.1043 ± 0.1042
(3,3)	$1.90\text{e}{-07} \pm 2.18\text{e}{-08}$	(5,3)	0.0045 ± 0.0005
(3,4)	$9.45\text{e}{-06} \pm 1.69\text{e}{-06}$	(5,4)	$0.0004 \pm 5.76\text{e}{-05}$
(3,5)	$0.0001 \pm 1.84\text{e}{-05}$	(5,5)	0.0197 ± 0.0024
(3,7)	0.0020 ± 0.0002	(5,7)	0.1098 ± 0.0142
(3,10)	0.0306 ± 0.0036	(5,10)	0.4235 ± 0.0382
(3,15)	0.2683 ± 0.0295	(5,15)	1.9447 ± 0.1864
(3,20)	0.8447 ± 0.0811	(5,20)	3.6547 ± 0.3659

the best result is obtained when the values of *population size* and *try number* parameters are 2 and 4, respectively.

As it was stated, the value of visual is considered high at first and then it decreases by Eq. (7), in which $L_{\min}$ determines the reduction rate. In Table 2, the effect of using different values of visual and $L_{\min}$ parameters on the *parent* efficiency with *population size* of 2 and *try-number* of 4 is tabulated. Regarding the results of Table 2, *parent* efficiency is improved by setting the visual value to 25 and the $L_{\min}$ value to 0.75, compared to other values for these two parameters.

In addition, it was obvious in the course of experiments that the best values of k and r_{conv} should be 3 and 0.5 for having the best convergence of *parent* swarms. The experiments showed that the best results are obtained when the Euclidean distance of the best AF of the *parent* swarm in mth and $m + 3$th iterations are calculated as a criterion for determining the convergence of the *parent*. In addition, by considering the value of r_{conv} as 0.5, the algorithm efficiency is improved.

4.2.2. *Parameter settings for child swarms*

Child swarms in PCAFSA are categorized as *best* and *non-best child* swarms in which the values of some parameters are different after an environment change, regarding the structure of PCAFSA as discussed in the previous section. There are different parameters for *best* and *non-best child*swarms that should be set in order to achieve the best performance of the proposed algorithm. Comprehensive experiments have been done on MPB with different values of *peaks number*, *change frequency* and *shift severity* in order to determine the best value of these parameters. Table 3 summarizes the best obtained values of PCAFSA parameters for best child and non-best child swarms as well as those for parent swarms.

4.3. *Comparison between PCAFSA and other related methods*

In this part of the paper, we compare the efficiency of PCAFSA and that of other state-of-the-art algorithms in this domain to perform the optimization process on different configurations of MPB. Table 3 summarizes the values of involved PCAFSA parameters for *parent*, *best child* and *non-best child* swarms.

The experiments have been conducted on MPB using a configuration which is presented in Table 4. The experimental results are obtained by the average of 50 executions. Each execution has been performed using different random seeds and it has continued up to 100 environment changes. Some of the presented results of the related works are obtained by implementing the methods and some of them are extracted from the related references.

In Table 5, the efficiency of the proposed algorithm on MPB with different numbers of peaks, a change frequency of 5000 and *shift severity* of 1 is compared with 21 state-of-the-art algorithms in this domain including: mQSO,[2] AmQSO,[25] CLPSO,[42] FMSO,[29] RPSO,[43] mCPSO,[2] SPSO,[44] rSPSO,[45] mPSO,[28] HmSO,[44]

Table 2. Effect of using different values of visual and $L_{\min}$ parameters on the efficiency of *parent* swarms.

Visual	$L_{\min}$	Offline error ± Standard error	Visual	$L_{\min}$	Offline error ± Standard error	Visual	$L_{\min}$	Offline error ± Standard error
5	0.95	0.03 ± 0.003	10	0.95	0.0760 ± 0.0061	15	0.95	0.1235 ± 0.0114
	0.9	5.9594 ± 4.2391		0.9	$0.0003 \pm 2.54\text{e}{-05}$		0.9	$0.0004 \pm 4.14\text{e}{-005}$
	0.8	222.8022 ± 28.0101		0.8	17.3333 ± 6.0815		0.8	0.0232 ± 0.0232
	0.75	232.43 ± 28.9920		0.75	83.7342 ± 13.7900		0.75	19.0188 ± 5.5565
	0.7	301.9345 ± 28.7206		0.7	140.1543 ± 22.2515		0.7	50.8755 ± 12.3201
	0.6	341.1569 ± 30.7649		0.6	205.7751 ± 25.1709		0.6	137.5714 ± 21.9111
20	0.95	0.2004 ± 0.0174	25	0.95	0.2157 ± 0.0197	30	0.95	0.2462 ± 0.0249
	0.9	$0.0005 \pm 6.56\text{e}{-05}$		0.9	$0.0007 \pm 9.99\text{e}{-05}$		0.9	$0.0009 \pm 8.13\text{e}{-05}$
	0.8	$2.42\text{e}{-09} \pm 3.21\text{e}{-10}$		0.8	$3.50\text{e}{-09} \pm 6.25\text{e}{-10}$		0.8	$6.57\text{e}{-009} \pm 1.36\text{e}{-09}$
	0.75	11.9557 ± 6.3010		0.75	$2.56\text{e}{-11} \pm 5.48\text{e}{-12}$		0.75	$5.88\text{e}{-11} \pm 1.22\text{e}{-12}$
	0.7	13.1628 ± 4.3364		0.7	6.8349 ± 4.6837		0.7	0.0375 ± 0.0375
	0.6	109.0391 ± 18.9038		0.6	49.5295 ± 12.5796		0.6	19.6831 ± 5.2877

Table 3. Values of the involved PCAFSA parameters for parent, best child and non-best child swarms.

Parameter	Parent	Best child	Non-best child
Population size	2	2	2
Swarm number	2	1	N/A
Try number	4	10	2
Initiale visaul	25	1× Shift severity	25
Initialize visual after:	Convergence	Environment change	Environment change
L_{min}	0.75	0.75	0.75
r_{div}	N/A	1× Shift severity	1× Shift severity
k	N/A	N/A	N/A
r_{conv}	0.5	N/A	N/A
r_{migr}	N/A	N/A	2× Shift severity
r_{excl}	d_{boa}[2]	d_{boa}[2]	d_{boa}[2]

Table 4. Configuration of MPB.

Parameter	Value
Number of peaks, M	1, 5, 10, 20, 30, 50, 100, 200 (variable)
Change frequency	2500, 5000, 10,000 (variable)
Height change	7.0
Width change	1.0
Peaks shape	Cone
Basic function	No
Shift length, severity	1.0, 2.0, 3.0, 5.0 (variable)
Number of dimensions, D	2, 3, 4, 5, 10, 15, 20
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

PSO-CP,[47] RVDEA,[50] SFA,[51] APSO,[45] CLDE,[10] DynPopDE,[9] DMAFSA,[11] CDEPSO,[52] CPSOL,[53] PSO-AQ,[54] Adaptive-SFA[55] and mNAFSA.[43]

As it could be seen in Table 5, the efficiency of the proposed method outperforms that of other 21 state-of-the-art algorithms in this domain. Setting the parameters of the swarms based on their operations is one of the most prominent reasons for the superiority of the proposed method. In fact, different operations and situations are involved in *parent*, *best child* and *non-best child* swarms and setting the parameters based on the operations and situations leads to improving the efficiency of the proposed method. On the other hand, diversity increase mechanism causes an appropriate diversity increment in swarms after an environment change, which leads to an increase in the convergence speed of swarms toward their new goals. The experimental results show that using MAFSA as the base algorithm is significantly useful and appropriate convergence speed is involved in this algorithm.

Table 5. Comparison between offline error (standard error) of the proposed method and that of 21 related algorithms on MPB with different number of peaks, a change frequency of 5000 and shift severity of 1.

	Number of Peaks							
Algorithm	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)
HmSO	0.87(0.05)	1.18(0.04)	1.42(0.04)	1.50(0.06)	1.65(0.04)	1.66(0.02)	1.68(0.03)	1.71(0.02)
PSO-CP	3.41(0.06)	—	1.31(0.06)	—	2.02(0.07)	—	2.14(0.08)	2.04(0.07)
RVDEA	1.02(−)	—	3.54(−)	3.87(−)	3.92(−)	3.78(−)	3.37(−)	3.54(−)
SFA	0.42(0.07)	0.89(0.09)	1.05(0.04)	1.48(0.05)	1.56(0.06)	1.87(0.05)	2.01(0.04)	1.99(0.06)
APSO	0.53(0.01)	1.05(0.06)	1.31(0.03)	1.69(0.05)	1.78(0.02)	1.95(0.02)	1.95(0.01)	1.90(0.01)
CLDE	1.53(0.07)	1.50(0.04)	1.64(0.03)	2.46(0.05)	2.62(0.05)	2.75(0.05)	2.73(0.03)	2.61(0.02)
DynPopDE	—	1.03(0.13)	1.39(0.07)	—	—	2.10(0.06)	2.34(0.05)	2.44(0.05)
DMAFSA	0.59(0.06)	0.66(0.05)	0.94(0.04)	1.29(0.05)	1.60(0.06)	1.81(0.06)	1.92(0.05)	1.97(0.05)
CDEPSO	0.41(0.00)	0.97(0.01)	1.22(0.01)	1.54(0.01)	2.62(0.01)	2.20(0.01)	1.54(0.01)	2.11(0.01)
CPSOL	1.02(0.14)	0.99(0.15)	1.75(0.10)	1.93(0.11)	2.28(0.10)	2.74(0.10)	2.84(0.12)	2.69(0.08)
PSO-AQ	0.34(0.02)	0.80(0.12)	0.89(0.03)	1.45(0.06)	1.52(0.04)	1.77(0.05)	1.95(0.05)	1.96(0.04)
Adaptive-SFA	0.66(0.05)	0.79(0.06)	0.95(0.05)	1.29(0.07)	1.51(0.06)	1.71(0.04)	1.84(0.04)	1.90(0.05)
mNAFSA	0.38(0.06)	0.55(0.04)	0.90(0.03)	1.25(0.06)	1.47(0.05)	1.68(0.05)	1.83(0.05)	1.84(0.05)
PCAFSA	0.33(0.01)	0.48(0.01)	0.65(0.02)	1.03(0.02)	1.46(0.02)	1.53(0.03)	1.60(0.02)	1.65(0.02)

5. Conclusion

In this paper, a novel algorithm was proposed for optimization in dynamic environments in which changes in problem space have occurred in discrete intervals. The proposed algorithm was able to find the peaks quickly in the problem space and follow them after an environment change. In the proposed algorithm, swarms in the problem space were categorized into *parent*, *best child* and *non-best child* swarms, each of which was configured in a way that it can demonstrate high efficiency in performing its tasks. In the proposed algorithm, all of the AFs performed a search process based on prey, follow, and swarm behaviors. Each swarm has been equipped with some mechanisms, based on its corresponding function, to overcome its particular challenges.

The efficiency of the proposed algorithm has been evaluated on MPB, which is the most well-known benchmark in this domain, and its results were compared with those of the state-of-the-art algorithms. The experimental results and comparative studies showed the superiority of the proposed method. Diverse experiments showed that the proposed algorithm involved a high convergence speed along with high accuracy which is significantly important in designing optimization algorithms in dynamic environments.

A primary knowledge concerning several parameters of the problem space, e.g., number of peaks, shift length and change frequency needed to be determined in almost all previous algorithms in dynamic environments. In the proposed algorithm, we tried to solve such dependencies. Nevertheless, the shift severity parameter must be determined for performing the proposed algorithm. By adding online learning algorithms or self-adaptive mechanisms to the proposed algorithm, the algorithm could be completely independent of the primary knowledge which will be pursued as future works.

References

1. T. Nguyen, Continuous dynamic optimization using evolutionary algorithms, Ph.D. Thesis, The University of Birmingham, UK (2010).
2. T. Blackwell and J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, *IEEE Trans. Evolut. Comput.* **10** (2006) 459–472.
3. S. Yang and C. Li, A Clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments, *IEEE Trans. Evolut. Comput.* **14** (2010) 959–974.
4. C. Li and S. Yang, A general framework of multi-population methods with clustering in undetectable dynamic environments, *IEEE Trans. Evolut. Comput.* **16** (2012) 556–577.
5. Y. Jin and J. Branke, Evolutionary optimization in uncertain environments — A survey, *IEEE Trans. Evolut. Comput.* **9** (2005) 303–317.
6. J. Kennedy and R. Eberhart, Particle swarm optimization, in *proc. IEEE Int. Conf. Neural Networks*, 27 November–1 December 1995, Perth, WA, pp. 1942–1948.
7. D. Angus and T. Hendtlass, Dynamic ant colony optimization, *Appl. Intell.* **23** (2005) 23–38.

8. M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, *IEEE Comput. Intell. Mag.* **1** (2006) 28–39.
9. M. C. Plessis and A. P. Engelbrecht, Differential evolution for dynamic environments with unknown numbers of optima, *J. Glob. Optim.* **55** (2013) 73–99.
10. V. Noroozi *et al.*, CellularDE: A cellular based differential evolution for dynamic optimization problems, in *Adaptive and Natural Computing Algorithms* (Springer, Berlin Heidelberg, Germany, 2011), pp. 340–349.
11. D. Yazdani, M. R. Akbarzadeh-Totonchi, B. Nasiri and M. R. Meybodi, A new artificial fish swarm algorithm for dynamic optimization problems, in *Proc. IEEE Congress on Evolutionary Computation*, 10–15 June 2012, Brisbane, QLD, pp. 1–8.
12. J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, *Evolut. Comput.* **3** (1999) 1875–1882.
13. J. Branke, Moving peaks benchmark (2016) http://people.aifb.kit.edu/jbr/MovPeaks/.
14. A. Uyar and A. E. Harmanci, A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments, *Soft Comput.* **9** (2005) 803–814.
15. S. Yang, Memory-based immigrants for genetic algorithms in dynamic environments, in *Proc. 7th Ann. Conf. Genetic and Evolutionary Computation*, 25–29 June 2005, New York, pp. 1115–1122.
16. H. Richter, Memory design for constrained dynamic optimization problems, *Applications of Evolutionary Computation*, Lecture Notes in Computer Science, Vol. 6024 (Springer, Germany, 2010), pp. 552–561.
17. A. Simões and E. Costa, Evolutionary algorithms for dynamic environments: Prediction using linear regression and Markov chains, *Parallel Problem Solving from Nature — PPSNX*, Lecture Notes in Computer Science, Vol. 5199 (Springer, Germany, 2008), pp. 306–315.
18. H. Richter and S. Yang, Memory based on abstraction for dynamic fitness functions, *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, Vol. 4974 (Springer, Germany, 2008), pp. 596–605.
19. S. Yang and Y. Xin, Population-based incremental learning with associative memory for dynamic environments, *IEEE Trans. Evolut. Comput.* **12** (2008) 542–561.
20. R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments* (Springer-Verlag, New York, 2004).
21. J. Grefenstette, Genetic algorithms for changing environments, *in Proc. 5th Int. Conf. Genetic Algorithms*, Amsterdam, 1992 (Morgan Kaufmann, San Francisco, CA, 1992), pp. 137–144.
22. L. T. Bui, H. A. Abbass and J. Branke, Multiobjective optimization for dynamic environments, in *Proc. IEEE Congress on in Evolutionary Computation*, Edinburgh, 2005, Vol. 3 (IEEE, Piscataway, NJ, 2005), pp. 2349–2356.
23. H. Andersen, An investigation into genetic algorithms, and the relationship between speciation and the tracking of optima in dynamic functions, Ph.D. Thesis, Queensland University of Technology, Brisbane, Australia (1991).
24. F. Oppacher and M. Wineberg, The shifting balance genetic algorithm: Improving the GA in a dynamic environment, in *Proc. Genetic and Evolutionary Computation Conf. (GECCO 1999)*, Vol. 1 (Morgan kaufmann, San Francisco, CA, 1999), 250 (Springer-Verlag, Germany, 2008), pp. 504–510.
25. T. Blackwell, J. Branke and X. Li, Particle swarms for dynamic optimization problems, *Swarm Intelligence* 193–217.
26. H. Cheng and S. Yang, Multi-population Genetic Algorithms with Immigrants Scheme for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks, *Applications*

of Evolutionary Computation, Lecture Notes in Computer Science Vol. 6024 (Springer, Germany, 2011), pp. 562–571.

27. H. C. Tsai and Y. H. Lin, Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior, *Appl. Soft Comput.* **11** (2011) 5367–5374.
28. M. Kamosi *et al.*, A new particle swarm optimization algorithm for dynamic environments, *Swarm, Evolutionary, and Memetic Computing*, Lecture Notes in Computer Science, Vol. 6466 (Springer-Verlag, Germany, 2010), pp. 129–138.
29. L. Changhe and Y. Shengxiang, Fast multi-swarm optimization for dynamic optimization problems, in *Proc 4th Int. Conf. Natural Computation*, Shandong, 2008 (IEEE Computer Society, LA, 2008), pp. 624–628.
30. R. K. Ursem *et al.*, Multinational GAs: Multimodal optimization techniques in dynamic environments, *in Proc the 2nd Genetic and Evolutionary Computation Conf.*, 2000 (Morgan Kaufmann, San Francisco, 2000), pp. 19–26.
31. X. Li, Z. Shao and J. Qian, An optimization method base on autonomous animates: fish swarm algorithm, *Syst. Eng. Theory Pract.* **22** (2002) 32–38.
32. W. Shen, X. Guo, C. Wu and D. Wu, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, *Knowl.-Based Syst.* **24** (2011) 378–385.
33. D. Yazdani, H. Nabizadeh, E. M. Kosari and A. N. Tossi, Color quantization using modified artificial fish swarm algorithm, in *A1 2011: Advances in Artificial Intelligence* (Springer-Verlag, Germany, 2011), pp. 382–391.
34. A. M. A. C. Rocha, T. F. M. C. Martins and E. M. G. P. Fernandes, An augmented Lagrangian fish swarm based method for global optimization, *J. Comput. Appl. Math.* **235** (2011) 4611–4620.
35. D. Yazdani, A. N. Tossi and M. R. Meybodi, Fuzzy adaptive artificial fish swarm algorithm, in *A1 2010: Advances in Artificial Intelligence* (Springer-Verlag, Germany, 2010), pp. 334–343.
36. W. Zhu, J. Jiang, C. Song and L. Bao, Clustering algorithm based on fuzzy C-means and artificial fish swarm, *Procedia Eng.* **29** (2012), 3307–3311.
37. Y. Cheng, M. Jiang and D. Yuan, Novel clustering algorithms based on improved artificial fish swarm algorithm, in *Proc. 6th Int. Conf. Fuzzy Systems and Knowledge Discovery*, 14–16 August 2009, Tianjin, pp. 141–145.
38. D. Yazdani, B. Saman, A. Sepas-Moghadam, F. Mohammad-Kazemi and M. R. Meybodi, A new algorithm based on improved artificial fish swarm algorithm for data clustering, *Int. J. Artif. Intell.* **11** (2013) 193–221.
39. M. Jiang and K. Zhu, Multiobjective optimization by artificial fish swarm algorithm, in *Proc IEEE Int. Conf. Computer Science and Automation Engineering*, 10–12 June 2011, Shanghai, pp. 506–511.
40. A. Cheng and X. Hong, PID controller parameters optimization based on artificial fish swarm algorithm, in *Proc. 5th Int. Conf. Intelligent Computation Technology and Automation*, 12–14 January 2012, Zhangjiajie, Hunan, pp. 265–268.
41. D. Yazdani, A. Arabshahi, A. Sepas-Moghaddam and M. Dehshibi, A multilevel thresholding method for image segmentation using a novel hybrid intelligent approach, in *Proc. 12th Int. Conf. Hybrid Intelligent Systems*, December 2012, Puna, pp. 137–142.
42. A. Sepas-Moghaddam, D. Yazdani and J. Shahabi, A novel hybrid image segmentation method, *Prog. Artif. Intell.* **3** (2014) 39–41.
43. D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. Meybodi and M. Akbarzadeh-Totonchi, mNAFSA: A novel approach for optimization in dynamic environments with global changes, *Swarm Evolut. Comput.* **18** (2014) 38–53.

44. M. Kamosi, A. Hashemi and M. Meybodi, A hibernating multi-swarm optimization algorithm for dynamic environments, in *Proc. World Congress on Nature and Biologically Inspired Computing*, 19–21 October 2010, Kitakyushu, pp. 370–376.
45. I. Rezazadeh, M. Meybodi and A. Naebi, Adaptive particle swarm optimization algorithm for dynamic environments, *Advances in Swarm Intelligence*, Lecture Notes in Computer Science, Vol. 6728 (Springer, Germany, 2011), pp. 120–129.
46. A. Hashemi and M. Meybodi, Cellular Pso: A Pso for Dynamic Environments, *Advances in Computation and Intelligence*, Lecture Notes in Computer Science, Vol. 5821 (Springer-Verlag, Germany, 2009), pp. 422–433.
47. L. Liu *et al.*, Particle swarm optimization with composite particles in dynamic environments, *IEEE Trans. Sys. Man Cybern. B, Cybern.* **40** (2010) 1634–1648.
48. S. Bird and X. Li, Using regression to improve local convergence, in *Proc. IEEE Congress on Evolutionary Computation*, 25–28, September 2007, Singapore, pp. 592–599.
49. W. Du and B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, *Inf. Sci.* **178** (2008) 3096–3109.
50. Y. G. Woldesenbet and G. G. Yen, Dynamic evolutionary algorithm with variable relocation, *IEEE Trans. Evolut. Comput.* **13** (2009) 500–513.
51. B. Nasiri and M. Meybodi, Speciation based firefly algorithm for optimization in dynamic environments, *Int. J. Artif. Intell.* **8** (2012) 118–132.
52. J. K. Kordestani, A. Rezvanian and M. R. Meybodi, CDEPSO: a bi-population hybrid approach for dynamic optimization problems, *Appl. Intell.* **40** (2014) 682–694.
53. S. Nabizadeh, A. Rezvanian and M. R. Meybodi, Tracking extrema in dynamic environments using multi-swarm cellular pso with local search, *Int. J. Electron. Inf.* **1**(2012) 29–37.
54. D. Yazdani, B. Nasiri, R. Azizi, A. Sepas-Moghadam and M. R. Meybodi, Optimization in dynamic environment utilizing a novel method based on particle swarm optimization, *Int. J. Artif. Intell.* **11** (2013) 170–192.
55. B. Nasiri and M. R. Meybodi, Improved speciation-based Firefly Algorithm in dynamic and uncertain environment, *Int. J. Bio-Inspir. Comput.* (2016). In press.