Applied Soft Computing 62 (2018) 987–1002

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Tackling global optimization problems with a novel algorithm – Mouth Brooding Fish algorithm

Ehsan Jahani*, Mohammad Chizari*

Faculty of Engineering, University of Mazandaran, P.O. Box 47416-95447, Babolsar, Mazandaran, Iran

ARTICLE INFO

Article history:
Received 2 May 2016
Received in revised form
20 September 2017
Accepted 21 September 2017
Available online 28 September 2017

Keywords:
Mouth Brooding Fish algorithm
Nature-inspired algorithm
Evolutionary algorithm
Optimization algorithm

ABSTRACT

Nowadays, due to the fact that difficulty of global optimization problems in different fields is increasing, various methods have been introduced to solve such problems. This paper proposes a novel global optimization algorithm inspired by Mouth Brooding Fish in nature. Meta-heuristics based on evolutionary computation and swarm intelligence are outstanding examples of nature-inspired solution techniques. Mouth Brooding Fish (MBF) algorithm simulates the symbiotic interaction strategies adopted by organisms to survive and propagate in the ecosystem. The proposed algorithm uses the movement, dispersion and protection behavior of Mouth Brooding Fish as a pattern to find the best possible answer. This algorithm is evaluated by CEC2013&14 benchmark functions for single objective optimization and the proposed algorithm competes with the advanced algorithms (CMA-ES, JADE, SaDE, and GL-25). The results demonstrate that the proposed algorithm is able to construct very promising results and has merits in solving challenging optimization problems.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

In the past few decades, nature-inspired computation has attracted more attention [1]. Many real-world engineering optimization problems are substantially very complicated and quite difficult to solve. However, nature serves as a fertile source of concepts, principles, and mechanisms for designing artificial computation systems to tackle complicated computational problems [2]. In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from allowed set of data and computing the function value. The generalization of optimization theories and techniques to other formulations comprises a large area of applied mathematics [3]. Furthermore, optimization includes finding "best available" values of some objective function given a defined domain (or a set of constraints), including a variety of different types of objective functions and different types of domains. A lot of optimization algorithms based on nature have been proposed. Among them, global optimization algorithms such as Genetic Algorithm (GA) [4–9], Particle Swarm Optimization (PSO) [10–14], Differential Evolution Algorithm (DE) [15–19], etc. have received much attention and are widely used in many fields such as function optimization [20–22], civil engineering [23–28], signal processing [29], classification and machine learning [30,31] due to their advantages in obtaining global optima and their rapid convergence. Optimization algorithms have advantages and disadvantages compared to each other and may show different performances when solving discrete and continuous problems. Therefore, many scientists try to develop or create new artificial or evolutionary algorithms in order to surpass the performance of such methods.

In the present paper, an algorithm for global optimization, called the Mouth Brooding Fish (MBF) algorithm is presented for the first time. In nature, mouth brooding fishes employ their mouth as a barrier against the surrounding dangers threatening their children. MBF algorithm has been inspired by nature; it is based on the behavior and the distance of movement and dispersion of the children around the mother's mouth. In the section of Mouth Brooding Fish algorithm, and subsection of basic concepts, the initial and general concepts are developed from the natural behavior of the fish; MBF algorithm features, the way the concepts are manipulated within the algorithm; pseudo code discussion, coding comparison and discussion between MBF and well known PSO algorithm; and tuning parameters, the experimental process to determine which control parameters are robust or sensitive and perform suggestion for the users; are presented. In the section of Comparison with Advanced Algorithms, the efficiency of this algorithm is compared with the advanced algorithms (CMA-ES, JADE, SaDE, and GL-25), which have widespread applications in global optimization problems due to their advantage of accuracy and speed in obtaining global optima. Moreover, in this section, we use CEC2013&14

* Corresponding authors.
E-mail addresses: e.jahani@umz.ac.ir (E. Jahani), m.chizari@stu.umz.ac.ir (M. Chizari).

https://doi.org/10.1016/j.asoc.2017.09.035
1568-4946/© 2017 Elsevier B.V. All rights reserved.

base functions for single objective optimization and the outcomes are compared, and the conclusions are presented. In the section of MBF Algorithm Out-Performance, the proposed algorithm out-performance measured by a Chess Rating System for Evolutionary Algorithms (CRS4EAs) [32].

2. Literature review

2.1. Exploring the past

Evolution-inspired algorithms such as Genetic Algorithm (GA), that models the mechanism of genetic evolution, are popular and simple optimization tools so that they are preferred in a wide range of structural problem areas for the last two decades [33]. GA is a population-based algorithm, so the optimization process starts with a set of random solutions. We have a population of individuals that are typically used in GA [34]. These individuals are represented by a string of characters and are known as chromosomes. Each individual of the population is evaluated with a specific fitness function which shows how well the trial solution performs the required task [35]. Then, according to the principles of natural selections, three main operators evolve the initial population: selection, crossover, and mutation. The selection operator chooses proper individuals based on the fitness evaluation and ranking in order to be recombined by the crossover operator. Eventually, the mutation operator randomly chooses individuals and mutates new ones in order to maintain the diversity of the whole population [36]. This process iterates, and in each generation, the members with higher fitness values have a higher chance to crossover with other members. Therefore, the overall fitness of all members is increased until the desired output is achieved. Genetic algorithms have been applied to many hard optimization problems, but still, applications of GA in very complicated problems have not gained an appreciable edge and causes unacceptable delays in the optimization process.

Swarm-inspired algorithms such as Artificial Bee Colony (ABC) algorithm [37], Ant Colony Optimization (ACO) [38,39], Amoeboid Organism Algorithm (AOA) [40], and Particle Swarm Optimization (PSO) algorithm are popular nowadays. PSO which is the most popular one, imitates the collective behavior of birds or fish schooling [41]. Swarm intelligence, a new kind of evolutionary computing technique, has attracted much optimization problems interest in the recent years [42]. A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles) [43]. These particles are moved around in the search space according to a few simple formulas. The movements of the particles are guided by their own best known position in the search space as well as the entire swarm's best known position. When improved positions are being discovered, these will then come to guide the movements of the swarm [44]. The process is repeated, and by doing so, it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

In evolutionary computation, Differential Evolution (DE) is a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality [45]. DE is also a population-based optimization algorithm and was developed to optimize a real parameter real-valued functions [46]. Although it is more efficient than genetic algorithms, its algorithm complexity and stability to find the best answer is not suitable enough for a wide range of problems.

2.2. Why a new meta-heuristics

In recent years, the field of combinatorial optimization has witnessed a true tsunami of novel meta-heuristic methods, most of them based on a metaphor of some natural or man-made process [47], however, the impetus of creating such algorithms was provided by the growing needs to solve optimization problems that are very difficult or have many local minima, constraints or stochasticity [48]. There will be no progress in any fields if the world wants to remain in its past. For instance in the field of smart phones or cars, people are always looking for more simplicity, comfort, and speed, even though there is not much difference between the old and new models. Engineers are always trying to increase both accuracy and speed in solving real world optimization problems, therefore, new algorithms to achieve these goals are created for various applications. Three general important factors in solving complex optimization problems using algorithms are speed, accuracy, and a number of function evaluation [49]. The number of function evaluation is also named NFE. NFE is an important factor of optimization algorithms because the usual problems function is really complicated such as a dam or a bridge, which their function evaluation is very time-consuming [50]. Therefore, It should be noted that comparing on an equal number of iterations is mostly inappropriate for meta-heuristic algorithms (only in special cases) [51] because the number of function evaluation for each generation of algorithms might be different.

3. Mouth Brooding Fish algorithm

3.1. Basic concepts

In the sea, many species play the numbers game and broadcast thousands of eggs (with no thought to parental care) in the hope that some small percentage (a very small number) will actually survive to adulthood; members of the cichlids family devote considerable energy in rearing their youngsters. Rather than expend energy and resources in the form of huge numbers of eggs, these fish spawn relatively few offspring but offer them protection until such time that they are fully functional. Once the parents abandon their progeny, life is indeed tough. However, because they are functionally more mature, larger, somewhat more experienced and faster swimmers, they have a better chance of survival.

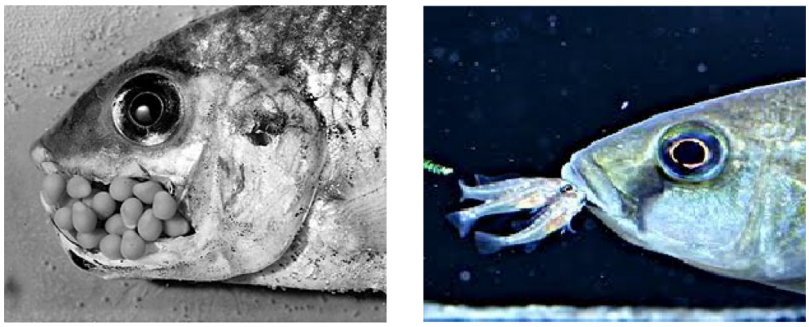
While many underwater creatures have strategies to protect themselves from harm, such as camouflage, not all have methods for protecting their young, too. Mouthbrooders, however, are well-known for their ability to take care and protect their offspring, largely due to a very unusual technique. Mouthbrooders protect their young by using their mouths as a shelter, as shown in Fig. 1a.

Species of fish defined as mouthbrooders include cichlids, sea catfish, cardinalfish, Bagrid catfish, pikeheads, jawfishes, gouramis and arowanas. African cichlids, which are maternal mouthbrooders, are an example of fishes that continues to protect their young even after the eggs have been hatched. Once released, the cichlid uses distinct behavioral cues when danger is present to let her young know it is time to swim back and seek protection. When they grow and the mother's shelter has not enough room for all of them, at the time of danger, some of the weak cichlids are left behind and have to face up with the danger and nature alone (Fig. 1b).

3.2. MBF algorithm features

3.2.1. Primary requirements

MBF algorithm has been inspired by the process of Mouth Brooding Fish life cycle. Like all other algorithms, MBF has a number of controlling parameters that are set by the user. These controlling parameters work like setting volumes that adjust the algorithm to the problem in order to reach a faster convergence and find the best possible answer. MBF algorithm has 5 controlling parameters which are determined by the user. These 5 controlling parameters are the number of population of cichlids (nFish), mother's source



(a) The way of protection (b) Left out cichlids

Fig. 1. Pictures of Mouth Brooding Fish.

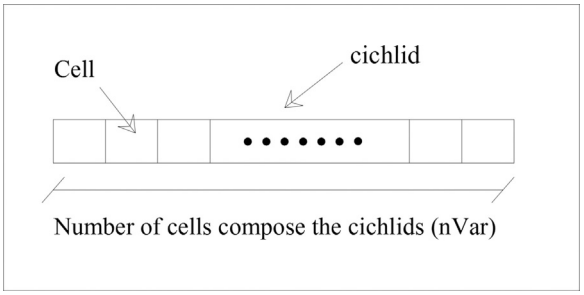


Fig. 2. Variable representation in cichlids.

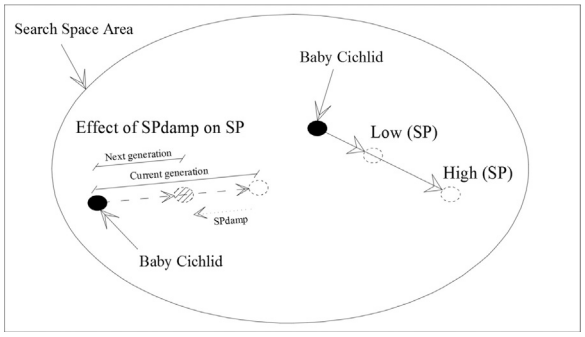


Fig. 3. Effects of SP and SPdamping in cichlids movements.

point (SP), the amount of dispersion (Dis), the probability of dispersion (Pdis), and mother's source point damping (SPdamp). In order to choose the best possible values for the controlling parameters, one may consider the problem and the results of tuning parameters. Although we must assume the controlling parameters as a constant to make the MBF algorithm capable for comparing with CMA-ES, JADE, SaDE, and GL-25. MBF algorithm is a population-based algorithm, therefore, the number of population (nFish) is one of the controlling parameters. The number of population (nFish) shows that how many fish will go through the operation of solving a problem in Mouth Brooding Fish algorithm. As it recognizable choosing a number of the population less than 5 is not logic, although, it depends on a user requirement and could be any positive number more than 2.

Every fish created in the algorithm is composed of some cells that represent the variables of the problem shown in Fig. 2. The main base of Mouth Brooding Fish algorithm is the movements of cichlids around their mother and the effects of nature or danger on these movements. MBF algorithm procedure consists of 4 main parts in order to find the best possible results for the problems.

3.2.2. The main movements of each cichlid

The proposed algorithm uses the movements of cichlids as the main factor, therefore, using all effects on cichlids movement could help the MBF algorithm to find the best possible answer for the problem.

1. The main movements of each cichlid
2. The additional movements of left out cichlids
3. Crossover with roulette wheel selection
4. Shark attack or effects of danger on cichlids movement

1. The first factor is the effect of mother's power or source point (SP) on movements of cichlids. This factor only affects the amount of movements and does not have any effects on the direction of movement. SP is one of the controlling parameters which has to be selected between 0 and 1. As can be seen in Fig. 3, by increasing the amount of SP the displacement between movements increases and it means that the cichlids have a strong mother and must keep up with her in the sea.

   It should be noted that increasing the mother's source point or power (SP) does not always lead to find the best possible answer, however, it only helps the MBF algorithm to solve local optima problems easier. However, the high value of (SP) would not help the MBF algorithm to solve problems with multi-local optima. The first effect of movements is defined as follows:

$$1st.Effect = SP \times Cichlids.Movements \quad (1)$$

   where *SP* is the mother's source point and Cichlids. Movements are the last movements of cichlids. In nature by spending the ages the power of mother will decrease and therefore this would affect the movement of the mother and their children. In MBF algorithm (SPdamp) mother's source point damping plays this part in each iteration and helps the Mouth Brooding Fish to better adapt to nature life. Fig. 3 shows (SPdamp) effects on cichlids movements and in end of each iteration the new Source point of the mother (SP) is calculated as follows:

$$SP = SP \times SPdamp \quad (2)$$

   where *SP* is mother's source point that changes for the next iteration and SPdamp is mother's source point damp and varies between 0.85 and 0.95.
2. The second factor of movements is the best position of cichlid which is found in the past iterations. Each cichlid likes to move to the best position that they get through passed iterations which are different from the current position and the best position of same cichlids as it is shown in Fig. 4. This movements effect can be controlled by the user through the other controlling parameter Dispersion (Dis) which is between 1 and 2. It should

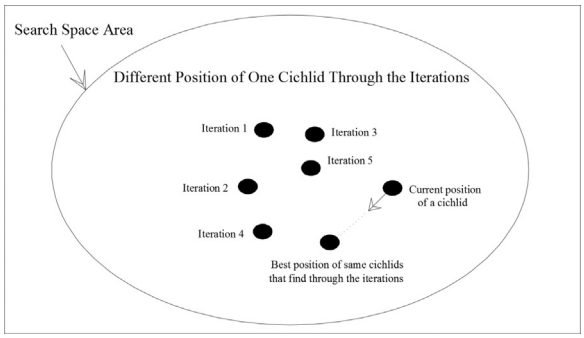


Fig. 4. Effects of best position of each cichlid on movements.

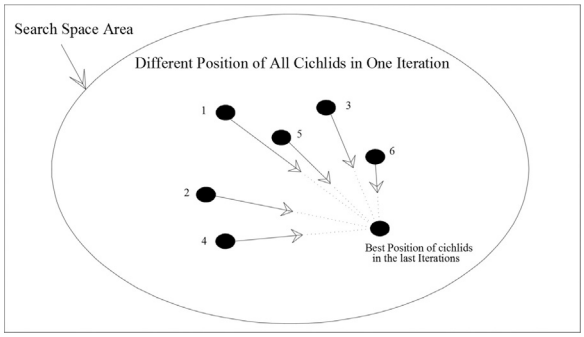


Fig. 5. Effects of the global best position on movements.

be noted that by increasing the amount of (Dis), the effectiveness of this movements increases. The second factor of movements is defined as follows:

$$2nd.Effect = Dis \times (Cichlids.Best - Cichlids.Position) \quad (3)$$

where *Cichlids.Best* is the best position that the cichlid gets through the past iterations and *Cichlids.Position* is the current position of the same cichlid. *Dis* is the amount of dispersion that is one of the controlling parameters which is selected by the user and could increase or decrease the effect of this movement.

3. The third factor of the movement is the tendency of all children to move toward the best possible position of whole cichlids as shown in Fig. 5. As can be seen in Fig. 5 each cichlid of colony tendency to move toward the best position found through the past iterations is calculated as follows:

$$3rd.Effect = Dis \times (Global.Best - Cichlids.Position) \quad (4)$$

where *Global.Best* is the best position found of all cichlids colony through passed iterations and *Cichlids. Position* is the current position for each cichlid. Like the second factor, the effects of this movement can be decreased or increased according to user's interest through the amount of dispersion (*Dis*).

4. The last factor for the base movements of cichlids is the nature trends or force. Nature always has an inclination to the best possible answer although these trends are slow but have their effects on all creatures. In MBF algorithm nature trend or force find through the past generation to the current generation. In the other words, the best position that is found for cichlids in the last and current generation, and has more than 60 percent difference in each cell, will be selected. After the selection of cells that has an enormous difference by using the mother's source point or power (SP) and multiplied by 10 we have a new position for the nature trends as follows:

$$NewN.F.P = 10 \times SP \times NatureForce.Position(SelectedCells) \quad (5)$$

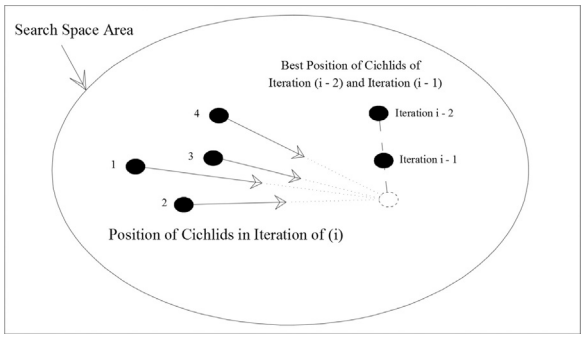


Fig. 6. Effects of nature trend on movements.

where *NatureForce.Position*(*SelectedCells*) is the selected cell from 60 percent difference cells of best position of last and current generation. As shown in Fig. 6, *NewN.F.P* is the position of the new intended place, that nature believes it as the source from which, the best possible results come from. Like the second and third effects of movements, the fourth effect can be controlled by the user through the amount of dispersion (*Dis*) and calculated as follows:

$$4th.Effect = Dis \times (NewN.F.P - NatureForce.Position) \quad (6)$$

where *NatureForce.Position* is the best position of cichlids of the last iteration. The sum of total effects represented the basic movements of each cichlid which is the sum of Eqs. (1), (3), (4), and (6). The effects of nature trends only come to an action when the whole convergence trend (i.e. the best cost of current iteration less than 15 percent better than the best cost of the last iteration) is not in good shape. For more simplicity in describing this concept, the general steps are shown in Algorithm (1).

In the basic movements of cichlids, each child could not move more than Additional surrounding dispersion positive or negative (ASDP or ASDN) which is defined as follows:

$$ASDP = 0.1 \times (VarMax - VarMin), ASDN = -ASDP \quad (7)$$

where *VarMin* and *VarMax* are the minimum and maximum limits of the problems variation respectively. After that, by adding the calculated movement of cichlids to the current position of cichlids, we find a new position for cichlids. If the recent position of cichlids is out of the search space area, the new movement will be added for the cichlids, by using the mirror effect (i.e. by negativing the movement changing the direction of movement) and it is defined as follows:

$$Cichlids.Movements = -Cichlids.Movements \quad (8)$$

where *Cichlids.Movements* is the movements of cichlids before and after of mirror effects. Each position of cichlids is also checked with search space limits (*VarMin* and *VarMax*) therefore no cichlids have left the search space area. The general steps the main movements of each cichlid are shown in Algorithm 1.

3.2.3. The additional movements of left out cichlids

In MBF algorithm, like in nature, the mother keeps and protects some of the cichlids. The mother can keep as many cichlids as its mouth capacity allows and the remaining members, which have to face up with challenges in nature, are named left out cichlids, reverse depends on the mother's source point (SP) (negative power of SP) are calculated as follows:

$$nm = 0.04 \times nFish \times SP^{-0.431} \quad (9)$$

where *nFish* is the population size of cichlids and *SP* is the mother's source point and nm is the number of left out cichlids. These left

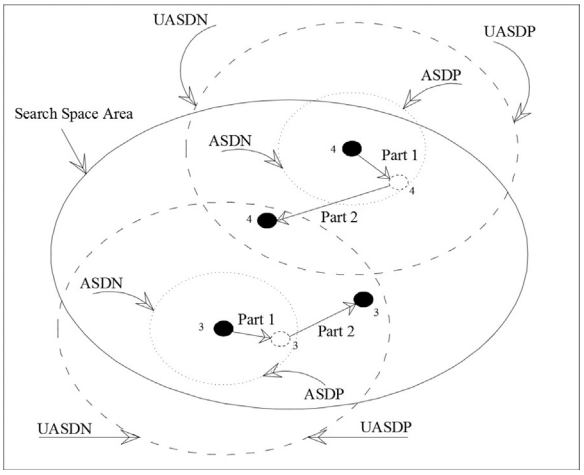


Fig. 7. Left out cichlids movements and limitations.

out cichlids in order to survive from danger have to move further from the basic movement, therefore, the limitation of movement is multiply by 4 as follows:

$$UASDP = 4 \times ASDP, UASDN = -UASDP \tag{10}$$

where *UASDP* and *UASDN* are the ultra-additional surrounding dispersion positive and negative limits for the left out cichlids movements and can be seen in Fig. 7. Left out cichlids have the second part of a movement which is shown in Fig. 6 that for the second part of movement Mouth Brooding Fish algorithm uses another controlling parameter named probability of dispersion (Pdis) and it is between 0 and 1. *Pdis* is used to calculate the number of cells for the chosen left out cichlids.

$$NCC = [nVar \times Pdis] \tag{11}$$

where the number of the cells that are to be changed or *NCC* is found by multiplying the probability of dispersion to *nVar* (number of variables) and rounding it to the nearest whole number. *NCC* is a parameter that significantly affects the direct dispersion of the fish and then a number of the cells equal to *NCC* are randomly selected. The second part of movement, shown in Fig. 7 is obtained from subtracted UASDP or UASDN by the left out cichlids randomly selected cells.

$$LeftCichlids.Position = UASDP \pm Cichlids.P(SelectedCells) \tag{12}$$

where *Cichlids.P(SelectedCells)* are the randomly selected cells of cichlids by the number of NCC. LeftCichlids.Position is the new position of left out cichlids after the second part of movements. As can be seen in Fig. 7 after the second movements cichlids could leave the search space area, therefore, we check the new position again with the search space limits. General steps of the additional movements of left out cichlids are shown in Algorithm 2

3.2.4. Crossover with roulette wheel selection

In nature, the marriage is one of the important operators that help the colony or population to converge through the best possible results. Although when it happens it does not always have good results. Mouth Brooding Fish allowed some of the best cichlids to marry in their population, therefore, in MBF algorithm by using a probability distribution or Roulette Wheel selection (i.e. members with higher points values have a higher chance) we select 1 pairs of parents from each cichlid. The single point crossover by the probability of crossover of 65 percent of the better parent (i.e. the one that has better fitness function result against another chosen parent) and 35 percent of another parent is conducted to generate the new fish and shown in Fig. 8. These newly born cichlids that have new position take the place of their parents in population and their movement would be zero. Before evaluating the newly born fish with fitness function we should check that the new position for the generated children is in the search space area. Crossover with roulette wheel selection pseudo-code is shown in Algorithm 2.

3.2.5. Shark attack or effects of danger on cichlids movement

The important part in each meta-heuristic method that is inspired by nature is that these methods contain all of the single conditions that affect the population. In nature 4 percent of each colony or population of cichlids are attacked by sharks or other natural threats, therefore, MBF algorithm for the 4 percent of the population uses additional movements named shark attack effect.

$$n_{shark} = 0.04 \times nFish \tag{13}$$

where n_{shark} is the number of cichlids that is chosen for shark attack effect. This effect uses the 60 percent cell change difference of natural trend in each generation by saving the cell number and how many times through the generations this cell has the 60 percent difference from the best results of the last generation. As can be seen in Fig. 9 the number of repeated cell change will affect more than the other ones. Shark attack affects 4 percent of cichlids population on position and movements as follows:

$$Cichlids.NewPosition = SharkAttack \times Cichlids.Position \tag{14}$$

where *SharkAttack* is the matrix that holds the number of cells and how many times they have changed and *Cichlids. Position* is the randomly selected cichlids from 4 percent population. The movements of cichlids that are affected by shark attack are equal to the difference between the position of cichlids before and after shark attack. General pseudo-code steps of Shark Attack on cichlids

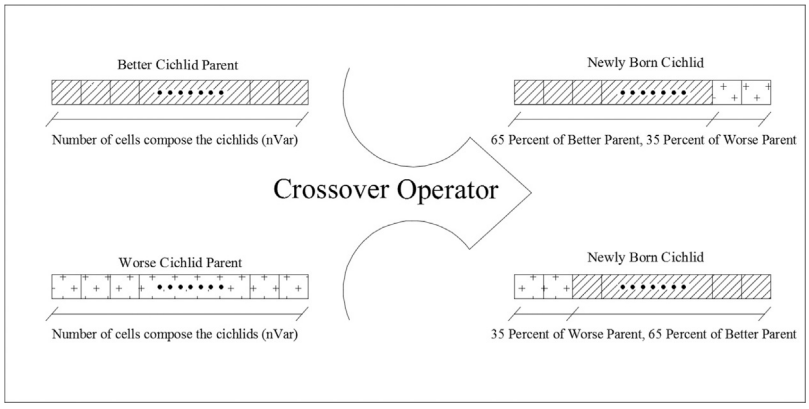


Fig. 8. Crossover operator.

movements are shown in Algorithm 3. Afterward, the new position should be checked to ensure that they are in the search space area. This trend is repeated as many times as the algorithm stops or the desired answer is gained. It should be noted that the stopping criteria are chosen to be the number of function evaluation (NFE), MBF algorithm should check the maximum number of NFE as stopping criteria after each step of evaluating with fitness function in order to properly implement termination condition since MBF consume more fitness evaluations per generation than other meta-heuristics which are used in this paper. Consequently, we use the maximum number of NFE as termination one could see the maximum number of NFE after each step of evaluating with fitness function (see Algorithm 1–3).

3.3. Pseudo code discussion

We performed a comparison between MBF and the well-known PSO algorithm to provide more simplicity in understanding the MBF algorithm pseudo code. It is worth pointing out that in MBF algorithm features section, all of the features had been discussed; therefore, in this section merely the general expression of the proposed algorithm pseudo code is considered. The general steps to be taken in the MBF algorithm are clearly described in Algorithms 1-3.

Algorithm 1. General pseudo-code steps of MBF algorithm (Part-1)

input : Parameters: nFish, SP, SPdamp, Dis, Pdis
FirstCalculation : (nm) number of Left out cichlids (Eq.(9)) and (n_{shark}) number of SharkAttack (Eq.(13))
SecondCalculation: (ASDP, ASDN) (Eq.(7)) and (UASDP, UASDN) (Eq.(10)) the movement limitations of cichlids

Create a number of nFish initial populations of cichlids position;
Assign a value of 0 to the movement of each created cichlid;
Evaluating all cichlids position with the fitness function;
Assign the current cichlids position to their best local position;
Finding the best position of all cichlids and added to the global BestResult;
while *stopping criteria not met (NFE)* **do** `// MBF Main Loop`
  for *All cichlids (nFish)* **do** `// The Main Movements`
    if *the global* $BestResult(it)/BestResult(it-1) < 0.85$ **then**
      Update the movement of Cichlids with 1st effect (Eq.(1)) + 2nd effect (Eq.(3)) + 3rd effect (Eq.(4))of movements;
    else
      Update the movement of Cichlids with 1st effect (Eq.(1)) + 2nd effect (Eq.(3)) + 3rd effect (Eq.(4)) + NatureForce effect (Eq.(6)) of movements;
    Checking the movements limitation with ASDP and ASDN (Eq.(7));
    Update cichlids positions with calculated movements;
    Checking the mirror effect (Eq.(8)) and position limitations (VarMin, VarMax);
    Evaluating all cichlids new position with fitness function;
    Update cichlids Local and global best position;
    Checking the stopping criteria (a maximum number for NFE);

Algorithm 2. General pseudo-code steps of MBF algorithm (Part-2)

```
                    // Continue of Main Loop
for Left out cichlids (nm) do      // Left Out Cichlids Movements
    Calculate the number of cell change NCC (Eq.(11));
    Choosing randomly (NCC) number of the cichlid cells;
    for Number of Cells Change (NCC) do
        UASDP randomly plus minus the selected cell of the left out
         cichlid;
    Calculate the left out cichlid movement with the minus of the new and
     old position;
    Checking the movement limitation with UASDP and UASDN
     (Eq.(7));
    Update the left out cichlids position with calculated movement;
    Checking the mirror effect (Eq.(8)) and position limitations
     (VarMin,VarMax);
    Evaluating the left out cichlids with fitness function;
    Update cichlids Local and global best position;
    Checking the stopping criteria (a maximum number for NFE);
Calculate the roulette wheel probability with cost of calculated from fitness
 function      // Crossover;
for 1 pair of cichlids do
    Roulette wheel selection of 1 pair of parents;
    Use the crossover operator by 65 of better and 35 of worse cichlids
     and vice versa to create new born cichlids;
    Checking the position limitations (VarMin,VarMax);
    Evaluating the newly born cichlids with fitness function;
    Update cichlids Local and global best position;
    Checking the stopping criteria (a maximum number for NFE);
```

Algorithm 3. General pseudo-code steps of MBF algorithm (Part-3)

```
                    // Continue of Main Loop
if Iteration > 1 then      // NatureForce Effect
    Calculate the difference of the global BestResult of previous(it-1)
     and current(it) iteration;
    Evaluate and select cells which the current(it) global BestResult have
     60 percent difference from the previous(it-1) one;
    10 times of SP to those number of cells in difference of global
     BestResult (Eq.(5));
    NatureForce movement equal to calculated position (Eq.(6));
if Iteration > 1 then      // SharkAttack
    Evaluate and select cells which the current(it) global BestResult have
     60 percent difference from the previous(it-1) one;
    SharkAttack matrix compose from the nVar and how many times
     cells are determined from 60 percent difference (Eq.(13));
    Calculate the times of SharkAttack matrix on each cell of current(it)
     and previous(it-1) global BestResult;
    for (n_shark) number cichlids do
        Randomly select a cichlid;
        Calculate the new position of the cichlid (Eq.(14)) ;
        Calculate the cichlid movement from minus of the current position
         and one of the calculated position from SharkAttack matrix to
         cichlid;
        Checking the mirror effect (Eq.(8)) and position limitations
         (VarMin,VarMax);
        Evaluating the new position of SharkAttack cichlids
         (n_shark) with fitness function;
        Update cichlids Local and global best position;
        Checking the stopping criteria (a maximum number for NFE);
end while          // End of Main Loop
Output the best solution;
```

As it can be seen, the part of general pseudo-code steps of the main movements of each cichlid is mentioned in Algorithm 1. Despite the fact that MBF and PSO algorithms don't have anything in common, MBF algorithm uses the first 3 effects of movements to find the best possible answer, as well as PSO algorithm. One of the differences between these two algorithms is the existence of nature force effect on the basic movements in MBF algorithm; the second difference is about the left out cichlids which are shown in Algorithm 2 which is responsible for exploration in whole search space area. Another difference is that, MBF algorithm uses the crossover operator in order to converge easier to the best possible answer (Algorithm 2). Last but not least Shark attack effect (Algorithm 3) on cichlids population that helps the MBF algorithm to predict the best answer, therefore, the proposed algorithm has faster convergence.

3.4. Tuning parameters

Sensitivity analysis is the experimental process by which we determine the relative importance of the various factors of a system. The goal of sensitivity analysis is to provide the

Table 1
Different types of functions for tuning parameters and the results of 30-dimensional problems of MBF algorithm.

F	Function	Related basic functions	MBF Mean ± SD
f_1	Rotated Bent Cigar Function	Bent Cigar Function	8.61E+03 ± 1.38E+04
f_2	Shifted and Rotated Katsuura Function	Katsuura Function	3.71E-01 ± 2.05E−01
f_3	Hybrid Function ($N = 4$)	Griewanks Function Weierstrass Function Rosenbrocks Function Scaffers F6 Function	2.46E+02 ± 3.94E+02
f_4	Composite Function ($N = 5$)	HGBat Function Rastrigins Function Schwefels Function Weierstrasss Function High Conditioned Elliptic Function	4.45E+02 ± 2.27E+01

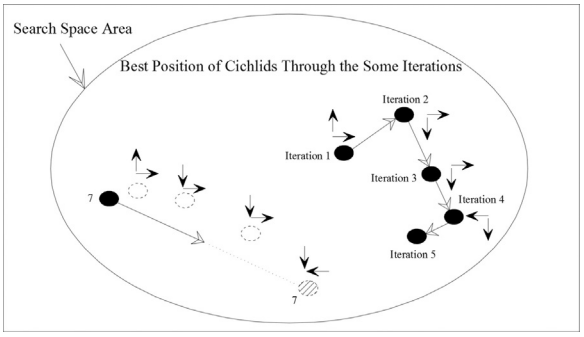


Fig. 9. Effects of shark attack on movements.

researcher/developer with insight and understanding of the key factors algorithmic, subject-based, procedural, environmental, among others that affect the matching performance of the system under study [52]. For every new meta-heuristics, we need to know how sensitive an approach is to its control parameters and which control parameters are robust/or sensitive [53]. However, sensitivity analysis goes through a long process that could take away the main focus of paper from the main subject of proposed algorithm; therefore, this section introduces parameter tuning which is running the MBF algorithm with different settings on control parameters and tries to suggest the best setting for each type of problems. These outcomes are essential for controlling the search processes when using MBF algorithm approach and by using these data, MBF algorithm may generate even better results in terms of accuracy and/or convergence on a global optimization problem.

For each single-objective global optimization algorithms, 4 types of problems could be considered using these functions: unimodal functions, simple multi-modal functions, hybrid functions, and composition function that controlling parameters must be analyzed in these four types of problems. For this experiment, the selected functions are f_1, f_2, f_3, and f_4 from Table 1. To perform tuning parameters, one of the controlling parameters value varies between min and max limit while others are constant at their average value and the stopping criteria would be determined as 5000 number of function evaluations (NFE). The results of the best solutions obtained for 30 dimensional functions are shown in Table 1. We only used the 30 dimensional of problems for controlling parameters, in order to make compact and more to the point discussion of parameter tuning, furthermore keeping the balance of 10, 30 and 50-dimensional of problems. Standard deviation and Error of each controlling parameter for each function are shown in Table 2. In Table 2 Error from the best answer obtained from a certain constant value of controlling parameters for each particular function which is shown in Table 1 calculated as follows:

$$Error = \frac{V - B}{W - B} \times 100 \tag{15}$$

where B is the best answer, obtained from a certain constant value of controlling parameters for each particular function which is shown in Table 1. W is the worst answer derived from all these cases of tuning parameters (i.e. between min and max) for each controlling parameter and function. V is the value of each case of controlling parameter for a certain function. In order to identify the level of impact and parameter tuning approaches for each control parameter in each type of problem, Fig. 10 shows the best outcomes of changing control parameter. According to the results obtained from 4 types of problems, suggested values of controller parameters to achieve the best possible answer of MBF algorithm for the each type of problem are shown in Table 3. Finally, based on accomplished analysis on proposed algorithm, in order, the best possible performance of MBF algorithm values of controlling parameters for any type of problems have been suggested shown in Table 4.

In the following section, the performance of the MBF algorithm is evaluated against CEC2013&2014 benchmark problems, and it is also compared to conventional optimization algorithms and these results have been used to choose the best possible value for the controlling parameters as a constant to fairly compare proposed algorithm with the other algorithms.

4. Comparison with advanced algorithms

4.1. Experiments setting

There are lots of newly developed and popular algorithms to solve recent problems and none of which have not been compared with the performance of MBF algorithm yet. Therefore in this section, we have chosen four (4) advanced algorithms that we had feedback from the paper which is published in Applied Soft Computing (ASC) in 2015 [54] to compete with MBF algorithm:

1. The variances of Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [55], CMA-ES and DE variants are arguably the most successful optimization algorithms current in use [56].
2. The DE variants, we select JADE [57] and SaDE [48] for comparison due to their excellent performance demonstrated in the CEC competitions. In recent years on Real-Parameter Single Objective Optimization Problems Competition at CEC, their variant algorithms (DE) always possess one or two positions in the top ten best-performing algorithms.
3. The global and local real-coded genetic algorithm (GL-25) [58].

Table 2
Tuning parameters results.

Parameter	Value	f_1	*Error*	SD	f_2	*Error*	SD	f_3	*Error*	SD	f_4	*Error*	SD
nFish	5	1.00E+04	100	4.90E+02	1.43E+00	100	1.35E−01	7.69E+02	100	1.72E+02	9.14E+02	100	2.35E+01
	10	9.65E+03	41.67		1.33E+00	16.67		6.47E+02	27.25		8.98E+02	41.21	
	25	9.31E+03	27.78		1.23E+00	11.12		5.26E+02	18.17		8.81E+02	27.48	
	50	**8.96E+03**	**13.89**		**1.14E+00**	**5.56**		**4.05E+02**	**9.09**		**8.64E+02**	**13.74**	
SP	Min=0.001	9.64E+03	41.14	5.84E+02	1.51E+00	100	1.61E−01	8.82E+02	100	2.05E+02	9.30E+02	100	2.81E+01
	0.25	**8.96E+03**	**13.72**		1.33E+00	16.46		7.62E+02	35.87		9.13E+02	54.85	
	0.5	9.30E+03	27.43		**1.14E+00**	**5.49**		5.23E+02	17.94		8.97E+02	41.14	
	0.75	9.98E+03	54.89		1.23E+00	10.97		**4.03E+02**	**8.97**		**8.64E+02**	**13.72**	
	Max=0.999	1.03E+04	100		1.42E+00	21.94		6.43E+02	26.91		8.80E+02	27.47	
Dis	Min=1.001	9.72E+03	100	3.78E+02	1.35E+00	100	1.05E−01	6.72E+02	100	1.33E+02	9.01E+02	100	1.82E+01
	1.25	9.28E+03	30.34		1.23E+00	11.12		5.94E+02	23.26		8.90E+02	35.56	
	1.5	**8.84E+03**	**8.89**		1.17E+00	7.17		5.16E+02	17.44		8.80E+02	26.69	
	1.75	9.06E+03	17.78		**1.10E+00**	**3.59**		**3.61E+02**	**5.86**		**8.58E+02**	**8.98**	
	Max=1.999	9.50E+03	35.56		1.29E+00	14.23		4.39E+02	11.63		8.69E+02	17.78	
Pdis	Min=0.001	**9.11E+03**	**20**	8.51E+02	2.76E+00	100	5.88E−01	9.82E+02	52.32	2.98E+02	9.20E+02	59.34	4.09E+01
	0.25	9.61E+03	40		2.42E+00	80		8.08E+02	39.24		8.96E+02	35.56	
	0.5	1.01E+04	60		2.08E+00	60		**4.58E+02**	**13.08**		**8.72E+02**	**19.78**	
	0.75	1.06E+04	80		**1.39E+00**	**20**		6.33E+02	26.16		9.44E+02	79.12	
	Max=0.999	1.11E+04	100		1.73E+00	40		1.16E+03	100		9.67E+02	100	
SPdamp	Min=0.8	9.09E+03	100	1.67E+02	1.17E+00	100	4.61E−02	4.49E+02	100	5.85E+01	8.70E+02	100	8.03E+00
	0.85	**8.73E+03**	**4.71**		1.14E+00	5.68		4.08E+02	9.30		8.65E+02	14.21	
	0.9	8.85E+03	9.48		**1.08E+00**	**1.95**		3.66E+02	6.20		8.59E+02	9.88	
	Max=0.95	8.97E+03	14.21		1.11E+00	3.78		**3.25E+02**	**3.12**		**8.53E+02**	**4.65**	

Bold values represents the best values.

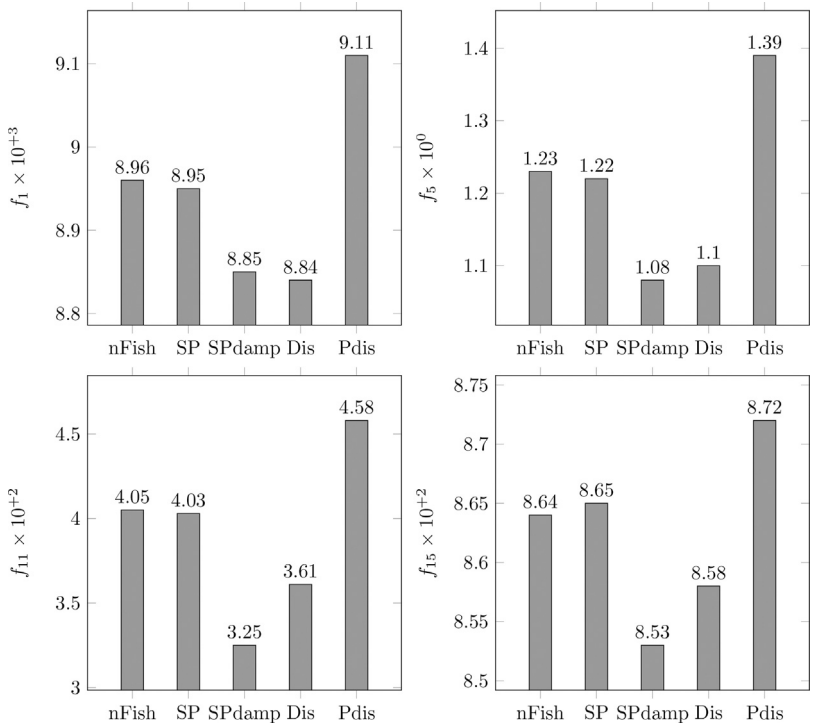


Fig. 10. Best values of each controlling parameters for different types of functions.

Table 3
Suggested values of controlling parameters of each types of problems.

Types of problems	nFish	SP	Dis	Pdis	SPdamp
Unimodal	50	0.3	1.5	0.5	0.85
Simple multi-modal	50	0.6	1.75	0.7	0.9
Hybrid	50	0.75	1.8	0.25	0.95
Composition	50	0.6	1.8	0.2	0.95

Table 4
Best possible performance for any types of problems.

nFish	SP	Dis	Pdis	SPdamp
50	0.6	1.8	0.2	0.95

In order to use a wide range of various tests on proposed algorithm and to benchmark the performance of MBF, we conduct simulations on 25 different benchmark functions which are from the published paper in one of the pioneers and respectful journals in this area Applied Soft Computing [54]. Thus to have an equal condition all parameters were the same (e.g., number of functions, number of independent runs) as in the test in the paper [54]. All benchmark functions locate their global minimum values at zero, and the fitness values smaller than 10^{-8} are considered as 10^{-8} as required in [59,60]. The test contains the 10, 30, and 50 dimensions of the each benchmark function and each function is tested for 51 runs [60]. Each run has the maximum number of $10^4 \times n$ function evaluations as the termination criteria and n is the dimension of the problem. [54].

The 25 benchmark functions are all the base functions from the latest competition on real-parameter single objective optimization problems at CEC 2013 [59] and CEC 2014 [60]. The benchmark functions can be classified into four groups:

- Group I: f_1–f_5 are Unimodal functions.
- Group II: f_6–f_{15} are Multi-modal functions.
- Group III: f_{16}–f_{20} are rotated Multi-modal functions whose base functions belong to Group II functions.
- Group IV: f_{21}–f_{25} are hybrid Multi-modal functions whose base functions belong to Group I–III functions.

The benchmark functions are listed in Table 5 where n is the dimension of the problem and M is a transformation matrix. The comparison criteria are mean and standard deviation of the best solutions obtained over 51 independent runs.

4.2. Numerical results

Tables 6–8 show the statistical analysis on the simulation results and perform a comparison between MBF and other algorithms. As a quick look to the 3 tables of outcomes (10, 30, and 50-dimensional problems) MBF algorithm generally, outperforms all compared algorithms in terms of a statistical test. In Table 6 10-dimensional problems among all 25 functions MBF generates better simulation results with non-similar outcomes in 11 functions in comparison with 1, 0, 0, and 5 functions for CMA-ES, JADE, SaDE, and GL-25, respectively. If the similar results are taken into account, the advantage is more obvious, in 19 functions MBF generates more promising results than all 25 benchmark function while 5, 4, 4, and 13 functions for CMA-ES, JADE, SaDE, and GL-25, respectively. In Table 7 30-dimensional problems among all 25 functions MBF generates better results for non-similar 10 against 1 (CMA-ES), 3 (JADE), 1 (SaDE), and 3 (GL-25) and for similar outcomes 17 (MBF) against 5, 9, 6, and 10 functions for CMA-ES, JADE, SaDE, and GL-25, respectively. The outcomes of 50-dimensional problems are shown in Table 8 and as can be seen for non-similar results in 17 functions MBF generates better outcomes and show the dominance of proposed algorithm as we expected against 1, 2, 1, and 2 functions for CMA-ES, JADE, SaDE, and GL-25, respectively. In another point of view, for 10 functions of group III and IV as expected by increasing the complexity of problems, MBF generates the better results. For 10 dimensional problems MBF generates better simulation results with non-similar outcomes in 3 functions and for 30 dimensional,

Table 5
CEC2013 and CEC2014 base functions for single objective optimization.

F	Exact solution	Used function	Search range
f_1	$f(x)=\sum_{i=1}^{n} x_i^2$	Shifted Sphere Function	[−100,100]
f_2	$f(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	Shifted Schwefel's Problem 2.22	[−10,10]
f_3	$f(x)=x_1^2+10^6\sum_{i=2}^{n} x_i^2$	Shifted Bent Cigar Function	[−100,100]
f_4	$f(x)=10^6 x_1^2+\sum_{i=2}^{n} x_i^2$	Shifted Discus Function	[−100,100]
f_5	$f(x)=\sum_{i=1}^{n} ix^4+rand()$	Shifted Quadratic Function with Noise	[−1.28,1.28]
f_6	$f(x)=10n+\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)]$	Shifted Rastrigin Function	[−5.12,5.12]
f_7	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+\cdots$ $20+\exp(1)$	Shifted Ackley Function	[−32,32]
f_8	$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\frac{x_i}{\sqrt{i}}+1$	Shifted Griewank Function	[−600,600]
f_9	$f(x)=\sum_{i=1}^{n-1}[100(x_{i-1}-x_i^2)^2+(x_i-1)^2]$	Shifted Rosenbrock Function	[−30,30]
f_{10}	$f(x)=\sin^2(\pi u_1)+\sum_{i=2}^{n-1}(u_i-1)^2[1+10\sin^2(\pi u_i+1)]+\cdots$ $(u_n-1)^2[1+\sin^2(2\pi x_n)],\ u_i=1+\frac{x_i+1}{4}$	Shifted Levy Function	[−50,50]
f_{11}	$f(x)=\frac{1}{10}[\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2(1+\sin^2(3\pi x_{i+1}))+\cdots$ $(x_n-1)^2(1+\sin^2(2\pi x_n))]+\sum_{i=1}^{n} u(x_i,5,100,4)$ $u(x_i,a,k,m):\{if\ x_i>a\}\rightarrow u=k(x_i-a)^m\ ;\ \{if\ x_i<-a\}\rightarrow u=k(-x_i-a)^m\ ;\ Otherwise\rightarrow u=0\ ;$	Shifted Penalized Function	[−50,50]
f_{12}	$f(x)=g(x_1,x_2)+g(x_2,x_3)+\cdots+g(x_{n-1},x_n)+g(x_n,1)$ $g(x,y)=0.5+\frac{(\sin^2(\sqrt{x^2+y^2})-0.5)}{(1+0.001(x^2+y^2))^2}$	Shifted Schaffers Function F6	[−100,100]
f_{13}	$f(x)=418.9828872724338\times n-\sum_{i=1}^{n} x_i\sin\sqrt{\lvert x_i\rvert}$	Shifted Schwefels Problem 2.26	[−500,500]
f_{14}	$f(x)=\left[\frac{1}{n-1}\sum_{i=1}^{n-1}(\sqrt{v_i}+\sin(50v_i^{0.2}))\sqrt{v_i}\right]^2$ $v_i=\sqrt{x_i^2+x_{i+1}^2}$	Shifted Schaffers Function F7	[−100,100]
f_{15}	$f(x)=\min\{\sum_{i=1}^{n}(x_i-\mu_1)^2,\quad n+s\times\sum_{i=1}^{n}(x_i-\mu_2)^2\}+\cdots$ $10\sum_{i=1}^{n}(1-\cos[2\pi(x_i-\mu_1)])$ $\mu_1=2.5,\quad \mu_2=-\sqrt{\frac{\mu_1^2-1}{s}},\quad s=1-\frac{1}{2\sqrt{n}-8.2}$	Shifted Lunacek Function	[−10,10]
f_{16}	Shifted and Rotated Griewank Function	$f_8(Mx)$	
f_{17}	Shifted and Rotated Rosenbrock Function	$f_9(Mx)$	
f_{18}	Shifted and Rotated Penalized Function	$f_{11}(Mx)$	
f_{19}	Shifted and Rotated Schaffers Function F6	$f_{12}(Mx)$	
f_{20}	Shifted and Rotated Lunacek Function	$f_{15}(Mx)$	
f_{21}	Hybrid Function 1	$[f_1,f_6,f_{13}]$	
f_{22}	Hybrid Function 2	$[f_6,f_8,f_9]$	
f_{23}	Hybrid Function 3	$[f_3,f_7,f_9,f_{11}]$	
f_{24}	Hybrid Function 4	$[f_6,f_7,f_8,f_9,f_{13}]$	
f_{25}	Hybrid Function 5	$[f_1,f_7,f_{10},f_{13},f_{15}]$	

Table 6
Results of 10-dimensional CEC2013 and CEC2014 base functions for each algorithm.

F	CMA-ES Mean ± SD	JADE Mean ± SD	SaDE Mean ± SD	GL-25 Mean ± SD	MBF Mean ± SD
f_1	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_2	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_3	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_4	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_5	2.78E−02 ± 1.87E−02	4.58E−04 ± 2.95E−04	2.29E−03 ± 1.98E−03	4.70E−04 ± 2.66E−04	**1.50E−04 ± 1.00E−04**
f_6	1.60E+01 ± 1.87E−02	2.34E+00 ± 2.48E+00	2.91E+00 ± 2.74E+00	4.61E+00 ± 1.72E+00	**1.00E−08 ± 0.00E+00**
f_7	5.62E−01 ± 2.90E+00	4.51E−01 ± 8.48E−01	9.14E−01 ± 1.16E+00	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_8	9.95E−03 ± 8.75E−03	4.06E−02 ± 6.97E−02	1.15E−01 ± 1.94E−01	2.41E−03 ± 6.32E−03	**1.64E−03 ± 1.96E−03**
f_9	3.13E−01 ± 1.08E+00	5.47E−01 ± 1.39E+00	3.08E+00 ± 7.04E+00	2.34E+00 ± 6.81E−01	**2.06E−01 ± 4.31E−01**
f_{10}	3.82E−02 ± 1.91E−01	4.88E−02 ± 1.14E−01	7.32E−02 ± 3.96E−01	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_{11}	1.72E−03 ± 4.04E−03	3.42E−03 ± 1.50E−02	1.26E−01 ± 5.57E−01	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_{12}	4.24E+00 ± 2.78E−01	2.83E−01 ± 2.46E−01	4.18E−01 ± 1.11E−01	1.16E+00 ± 7.46E−01	**1.11E−01 ± 7.32E−02**
f_{13}	1.61E+03 ± 3.69E+02	3.19E+02 ± 1.81E+02	1.15E+02 ± 1.18E+02	2.67E+02 ± 2.21E+02	**2.23E−03 ± 7.77E−03**
f_{14}	1.36E+01 ± 2.91E+01	3.88E−01 ± 2.51E+00	1.44E+00 ± 3.49E+00	3.17E−03 ± 5.31E−03	**1.00E−08 ± 0.00E+00**
f_{15}	1.36E+01 ± 7.35E+00	6.38E+00 ± 6.57E+00	1.14E+01 ± 6.65E+00	1.19E+01 ± 3.78E+00	**4.68E+00 ± 5.73E+00**
f_{16}	9.85E−03 ± 9.45E−03	2.63E−02 ± 3.09E−02	1.17E−01 ± 1.06E−01	**2.82E−03 ± 4.99E−03**	1.74E−01 ± 1.46E−01
f_{17}	**4.69E−01 ± 1.30E+00**	9.38E−01 ± 1.71E+00	1.75E+01 ± 3.36E+01	4.91E+00 ± 6.39E−01	7.68E+00 ± 2.39E+01
f_{18}	1.70E−03 ± 4.51E−03	1.29E−03 ± 3.58E−03	1.87E−02 ± 3.82E−02	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_{19}	4.21E+00 ± 2.21E−01	2.76E+00 ± 5.53E−01	3.26E+00 ± 3.83E−01	3.09E+00 ± 4.48E−01	**2.70E+00 ± 4.71E−01**
f_{20}	1.52E+01 ± 6.98E+00	7.67E+00 ± 5.05E+00	1.54E+01 ± 8.07E+00	**6.02E+00 ± 3.21E−01**	1.14E+01 ± 4.71E+00
f_{21}	7.07E+02 ± 2.56E+02	2.18E+02 ± 1.52E+02	6.46E+01 ± 9.33E+01	1.74E+00 ± 3.21E+00	**1.00E−08 ± 0.00E+00**
f_{22}	4.56E+01 ± 2.88E+01	1.06E+01 ± 1.74E+01	5.71E+00 ± 7.26E+00	**4.02E−01 ± 4.75E−01**	4.27E+00 ± 7.71E+00
f_{23}	2.13E+01 ± 4.43E+00	3.44E+00 ± 7.53E+00	3.31E+00 ± 7.02E+00	**7.76E−03 ± 4.06E−02**	2.57E+00 ± 4.48E+00
f_{24}	4.07E+02 ± 1.66E+02	1.03E+02 ± 8.79E+01	3.05E+01 ± 6.38E+01	1.15E+00 ± 1.34E+00	**1.06E+00 ± 3.93E+00**
f_{25}	3.22E+02 ± 1.63E+02	1.00E+02 ± 1.05E+02	4.88E+01 ± 7.23E+01	**1.95E−01 ± 4.46E−01**	4.29E+00 ± 6.02E+00

Bold values represents the best values.

Table 7
Results of 30-dimensional CEC2013 and CEC2014 base functions for each algorithm.

F	CMA-ES Mean ± SD	JADE Mean ± SD	SaDE Mean ± SD	GL-25 Mean ± SD	MBF Mean ± SD
f_1	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_2	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_3	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_4	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_5	6.88E−02 ± 2.21E−02	1.39E−03 ± 5.68E−04	8.23E−03 ± 3.22E−03	1.77E−03 ± 5.19E−04	**1.29E−03 ± 1.40E−03**
f_6	5.90E+01 ± 1.56E+01	**7.80E−02 ± 2.70E−01**	1.81E+00 ± 1.59E+00	2.36E+01 ± 6.10E+00	3.03E+00 ± 1.29E+00
f_7	4.34E+00 ± 5.77E+00	8.86E−02 ± 3.14E−01	8.57E−01 ± 8.68E−01	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_8	1.84E−03 ± 4.59E−03	4.81E−03 ± 1.39E−02	4.40E−02 ± 9.22E−02	**1.00E−08 ± 0.00E+00**	1.63E−02 ± 1.97E−02
f_9	**5.47E−01 ± 1.39E+00**	3.72E+00 ± 1.12E+01	3.22E+01 ± 2.99E+01	2.12E+01 ± 9.22E−01	1.57E+01 ± 2.15E+01
f_{10}	5.73E−02 ± 2.31E−01	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_{11}	6.46E−04 ± 2.61E−03	**1.00E−08 ± 0.00E+00**	9.48E−02 ± 3.79E−01	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_{12}	1.37E+01 ± 3.02E−01	9.30E−01 ± 1.10E−01	1.22E+00 ± 1.77E−01	1.04E+01 ± 1.07E+00	**2.91E−01 ± 9.38E−02**
f_{13}	4.89E+03 ± 6.15E+02	2.25E+02 ± 1.86E+02	**2.79E+01 ± 5.60E+01**	3.54E+03 ± 7.98E+02	2.88E+02 ± 7.72E+02
f_{14}	4.21E+01 ± 6.16E+01	6.05E−05 ± 4.32E−04	2.46E−01 ± 8.52E−01	4.01E−01 ± 2.44E−01	**2.08E−05 ± 2.88E−04**
f_{15}	8.02E+01 ± 1.48E+01	3.26E+01 ± 1.57E−01	3.48E+01 ± 2.22E+00	6.03E+01 ± 7.02E+00	**3.25E+01 ± 1.24E+01**
f_{16}	2.32E−03 ± 5.06E−03	1.49E−02 ± 4.05E−02	3.76E−02 ± 8.57E−02	1.61E−05 ± 7.55E−05	**1.06E−05 ± 1.22E−04**
f_{17}	8.60E−01 ± 1.66E+00	**5.47E−01 ± 1.39E+00**	8.07E+01 ± 9.08E+01	6.37E+01 ± 3.32E+01	6.14E+01 ± 7.56E+01
f_{18}	2.80E−03 ± 4.84E−03	8.62E−04 ± 2.98E−03	6.94E−01 ± 2.21E+00	**1.00E−08 ± 0.00E+00**	5.85E−04 ± 1.46E−03
f_{19}	1.38E+01 ± 2.84E−01	1.10E+01 ± 3.24E−01	1.20E+01 ± 4.23E−01	1.33E+01 ± 6.16E−01	**1.08E+01 ± 7.23E−01**
f_{20}	8.54E+01 ± 1.57E+01	7.67E+01 ± 1.09E+01	9.50E+01 ± 1.37E+01	1.17E+02 ± 6.90E+01	**2.71E+01 ± 1.85E+01**
f_{21}	1.76E+03 ± 3.47E+02	3.53E+02 ± 1.93E+02	6.98E+01 ± 9.82E+01	2.41E+02 ± 1.77E+02	**1.34E−04 ± 8.44E−05**
f_{22}	1.55E+02 ± 4.92E+01	**9.81E+00 ± 1.38E+01**	1.32E+01 ± 1.33E+01	1.11E+01 ± 2.28E+00	3.06E+01 ± 2.98E+01
f_{23}	2.88E+01 ± 6.89E+00	3.90E+00 ± 6.57E+00	4.01E+00 ± 4.38E+00	7.43E−01 ± 1.03E+00	**6.01E−01 ± 1.67E−03**
f_{24}	1.30E+02 ± 2.60E+02	2.58E+02 ± 1.52E+02	3.54E+01 ± 5.93E+01	2.38E+01 ± 2.16E+01	**1.04E+01 ± 1.60E+01**
f_{25}	1.18E+03 ± 3.07E+02	1.89E+02 ± 1.35E+02	4.44E+01 ± 6.77E+01	**4.06E+01 ± 4.71E+01**	1.27E+02 ± 6.03E+01

Bold values represents the best values.

the number rises up to the 6 functions and for 50-dimensional problems, the dominance number of MBF goes up to the 8 from 10 functions of group III and IV.

The advantage of the proposed algorithm in generating more accurate results for each function could tell the story itself, however, nowadays it is far better to compare the power and reliability of the newly developed algorithms with other meta-heuristic methods by Chess Rating System for Evolutionary Algorithms (CRS4EAs) [32] which is discussed in next section.

5. MBF algorithm out-performance

In recent years, developing a new method of evolutionary process has become a popular technique in computational intelligence.

Table 8
Results of 50-dimensional CEC2013 and CEC2014 base functions for each algorithm.

F	CMA-ES Mean ± SD	JADE Mean ± SD	SaDE Mean ± SD	GL-25 Mean ± SD	MBF Mean ± SD
f_1	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_2	1.32E−08 ± 1.69E−08	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_3	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_4	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_5	7.88E+00 ± 5.55E+01	2.25E−03 ± 8.17E−04	1.65E−02 ± 6.08E−03	4.22E−03 ± 1.45E−03	**1.71E−03 ± 3.17E−03**
f_6	1.86E+02 ± 2.76E+02	**1.00E−08 ± 0.00E+00**	1.31E+00 ± 1.10E+00	4.98E+01 ± 1.18E+01	2.76E+01 ± 5.03E+00
f_7	1.38E+01 ± 8.17E+00	1.60E−01 ± 3.78E−01	9.60E−01 ± 6.57E−01	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_8	5.80E−04 ± 2.01E−03	8.13E−03 ± 1.70E−02	3.89E−02 ± 6.64E−02	**1.00E−08 ± 0.00E+00**	1.16E−02 ± 2.02E−02
f_9	1.56E−01 ± 7.82E−01	1.02E+00 ± 1.75E+00	7.36E+01 ± 2.81E+01	4.13E+01 ± 1.25E+00	**1.44E−01 ± 3.19E+00**
f_{10}	3.35E+00 ± 1.22E+01	2.44E−03 ± 1.22E−02	3.66E−03 ± 1.93E−02	**1.00E−08 ± 0.00E+00**	**1.00E−08 ± 0.00E+00**
f_{11}	8.62E−04 ± 2.98E−03	2.15E−04 ± 1.54E−03	2.82E−01 ± 9.77E−01	6.57E−02 ± 1.09E−01	**1.00E−08 ± 0.00E+00**
f_{12}	2.39E+01 ± 8.29E−01	1.73E+00 ± 1.96E−01	4.84E+00 ± 1.33E+00	1.91E+01 ± 1.53E+00	**1.10E+00 ± 1.44E+00**
f_{13}	8.45E+03 ± 7.98E+02	1.14E+02 ± 1.18E+02	**4.64E+00 ± 2.32E+01**	7.39E+03 ± 8.76E+02	2.09E+03 ± 2.59E+02
f_{14}	1.90E+02 ± 1.40E+02	**1.00E−08 ± 0.00E+00**	2.96E−01 ± 8.43E−01	8.74E+00 ± 3.56E+00	5.90E−03 ± 5.92E−03
f_{15}	1.53E+02 ± 2.51E+01	5.00E+01 ± 6.38E−14	5.23E+01 ± 1.72E+00	1.07E+02 ± 1.11E+01	**2.56E+01 ± 2.27E+01**
f_{16}	5.80E−04 ± 2.39E−03	2.20E−02 ± 2.63E−02	2.49E−02 ± 2.81E−02	1.42E−02 ± 1.85E−02	**2.96E−04 ± 1.14E−04**
f_{17}	5.47E−01 ± 1.39E+00	9.46E−01 ± 1.71E+00	6.22E+01 ± 3.07E+01	4.70E+01 ± 1.02E+00	**4.11E−01 ± 1.65E+00**
f_{18}	**1.08E−03 ± 3.30E−03**	1.29E−03 ± 3.58E−03	5.45E+00 ± 1.60E+01	1.31E−02 ± 2.86E−02	7.81E+00 ± 2.81E+00
f_{19}	2.41E+01 ± 7.83E−01	1.98E+01 ± 3.91E−01	2.17E+01 ± 2.58E−01	2.34E+01 ± 2.50E−01	**1.87E+01 ± 8.42E−01**
f_{20}	1.59E+02 ± 1.92E+01	1.19E+02 ± 1.03E+01	1.82E+02 ± 2.54E+01	1.99E+02 ± 1.25E+02	**1.04E+01 ± 1.37E+01**
f_{21}	3.29E+03 ± 5.11E+02	5.74E+02 ± 2.50E+02	3.96E+01 ± 8.09E+01	1.42E+03 ± 4.17E+02	**4.61E−02 ± 7.45E−02**
f_{22}	2.33E+02 ± 6.38E+01	2.33E+01 ± 1.84E+01	3.24E+01 ± 1.80E+01	2.40E+01 ± 4.10E+00	**2.07E+01 ± 2.13E+01**
f_{23}	2.87E+01 ± 8.78E+00	1.04E+01 ± 1.12E+01	1.20E+01 ± 2.07E+01	**4.68E+00 ± 7.69E−01**	1.35E+01 ± 8.15E+00
f_{24}	2.16E+03 ± 4.29E+02	3.36E+02 ± 1.88E+02	5.93E+01 ± 8.32E+01	2.10E+02 ± 1.62E+02	**1.72E+01 ± 1.36E+01**
f_{25}	1.95E+03 ± 3.67E+02	3.11E+02 ± 1.80E+02	3.02E+01 ± 5.72E+01	1.94E+02 ± 1.56E+02	**1.29E+01 ± 1.56E+01**

In order to decide a new proposed method is considered better than another algorithm, by employing inside the framework of any experimental analysis. This task, which may not be trivial, has become necessary to confirm whether a new proposed method offers a significant improvement, or not, over the existing methods for a given problems [61]. The Null Hypothesis Significance Testing (NHST) is of utmost importance for comparing evolutionary algorithms as the performance of one algorithm over another can be scientifically proven, however, NHST is often misused, improperly applied and misinterpreted [32]. Therefore, this paper uses a new method, called a Chess Rating System for Evolutionary Algorithms (CRS4EAs) [32] for the comparison and ranking of evolutionary algorithms.

Chess rating system presented 3 factors: rating (R), rating deviation (RD), and rating volatility (σ) to compare between algorithms that retrieved from Gliko rating system [62]. The rating shows the real power of the MBF algorithm over other algorithms. The rating deviation illustrates the reliability of algorithm. Smaller rating deviation denotes better algorithm reliability and vice versa. The rating volatility illustrates the degree of expected fluctuation in algorithm's rating. Lower rating volatility informs that the algorithms rating is more reliable and expected and vice versa.

For newly proposed algorithm and advanced algorithms, this paper uses the numerical results from comparison with advanced algorithms section. Numerical results of 10, 30, 50-dimensional of CEC2013 and CEC2014 problems in Tables 6–8 uses the calculation of the rating and rating deviation and rating violation for advanced algorithms (i.e. CMA-ES, JADE, SaDE, and GL-25) against MBF. Each algorithm plays an equal number of games, which means the equal number of comparisons. 10^{-3} is used for ϵ value in CRS4EAs and amount of minRD has been set to 50. The rating and rating deviation and rating violation that achieved from these results are shown in Table 9. Graphical presentation of rating with 99.7 percent of confidence intervals is illustrated in Fig. 11. In advanced algorithms comparison Fig. 11 demonstrates that Mouth Brooding Fish algorithm is significantly better than GL-25, CMA-ES, and JADE in statistics, even though in some performances MBF is not significantly different from SaDE; as respects, the real power (i.e. the rating (R)) of the MBF algorithm is really promising against advanced algorithms.

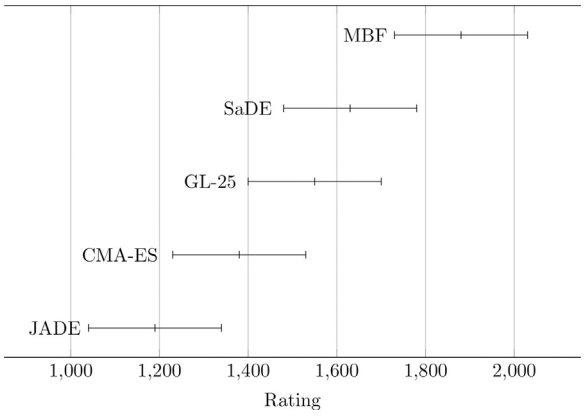


Fig. 11. 99:7% confidence intervals for the advanced algorithm's ratings from Table 9. The points along the black horizontal axes represent ratings.

6. Conclusion

In the present paper, the novel Mouth Brooding Fish (MBF) algorithm was presented to solve global optimization problems. This algorithm is based on the movements of the Mouth Brooding Fish in their life cycle and their children's struggle for survival. MBF can tackle a wide range of global optimization problems and has the potentiality to be employed to solve real-world problems because it is based on real-world phenomena. There are CEC2013 and CEC2014 groups of benchmark functions which are used in this paper in order to cover a wide range of various fields were employed to study the performance of the proposed method in comparison with the advanced algorithms (CMA-ES, JADE, SaDE, and GL-25). The tests were conducted on the algorithms to examine its convenience, complexity, and capability to solve general problems. In comparison with advanced algorithms, we use CEC2013

Table 9
Chess rating out performance of the advanced algorithms.

Algorithm	Rating (R)	Rating deviation (RD)	Rating volatility (σ)	Rating intervals (RI 99.7%)
MBF	1880	50	19	[1730,2030]
SaDE	1630	50	19	[1480,1780]
GL-25	1550	50	19	[1400,1700]
CMA-ES	1380	50	19	[1230,1530]
JADE	1190	50	19	[1040,1340]

and CEC2014 base functions for single objective optimization and MBF up against the advanced algorithms. The 10-dimensional problems outcomes showed the dominance of MBF against other advanced algorithms and for 30-dimensional problems, the proposed algorithm for both similar and non-similar, still stays at top of generating the best results; finally, for the 50-dimensional problems, MBF has produced very promising results against SaDE and by far has generated better results against CMA-ES, JADE, and GL-25. Although as can be seen, the proposed algorithm had more chance to generate better results when we increased the complexity (i.e. dimension) and also in more complicated functions which contain group III and group IV. At the end, MBF algorithm out-performance is considered and it is shown that the proposed algorithm has promising results in comparison with the advanced algorithms. As today's world problems become more complicated in every field, engineers and scientists are looking for stronger tools. Based on all the results and out-performance rating Mouth Brooding Fish (MBF) algorithm could assist the engineers and scientists to solve their global optimization problems.

Acknowledgment

The authors would like to thank the anonymous reviewer for useful and constructive comments.

References

[1] D.B. Fogel, The advantages of evolutionary computation, in: Bio-Computing Emergent Computing, World Scientific Press, Singapore, 1997.
[2] S. He, Q.H. Wu, J.R. Saunders, Group search optimizer: an optimization algorithm inspired by animal searching behavior, IEEE Trans. Evol. Comput. 13 (5) (2009) 973–990, http://dx.doi.org/10.1109/TEVC.2009.2011992.
[3] D. Karaboga, B. Akay, A comparative study of Artificial Bee Colony algorithm, Appl. Math. Comput. 214 (2009) 108–132.
[4] D. Beasley, D. Bull, R. Martin, An Overview of Genetic Algorithms: Part 1, University of Cardiff, Cardiff, 1993.
[5] D. Beasley, D. Bull, R. Martin, An Overview of Genetic Algorithms: Part 2, University of Cardiff, Cardiff, 1993.
[6] D.B. Fogel, What is evolutionary computation? IEEE Spectr. (2000) 26–32.
[7] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading, 1989.
[8] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1996.
[9] C. Houck, J. Joines, M. Kay, A Genetic Algorithm for Function Optimization: A MATLAB Implementation, Tech. Rep. 95-09, 1995.
[10] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE International Conference of Neural Network, vol. 4, IEEE Press, Piscataway, NJ, 1995, pp. 1942–1948.
[11] Y. Valle, G.K. Venayagamoorthy, S. Mohagheghi, J.C. Hernandez, R.G. Harley, Particle swarm optimization: basic concepts, variants and applications in power systems, IEEE Trans. Evol. Comput. 12 (2) (2008) 171–195.
[12] J. Kennedy, R.C. Eberhart, Y.H. Shi, Swarm Intelligence, Morgan Kaufmann, San Mateo, CA, 2001.
[13] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: IEEE Conference, 1997, pp. 4104–4108.
[14] O.M. Nezami, A. Bahrampour, P. Jamshidlou, Dynamic diversity enhancement in particle swarm optimization (DDEPSO) algorithm for preventing from premature convergence, Proc. Comput. Sci. 24 (2013) 54–65.
[15] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems, in: IEEE Congress on Evolutionary Computation, Piscataway, NJ, 2004, pp. 1980–1987.
[16] K. Price, R. Storn, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[17] K. Price, R. Storn, Differential Evolution: A Practical Approach to Global Optimization, Springer Natural Computing Series, 2005.
[18] P. Bergey, C. Ragsdale, Modified differential evolution: a greedy random strategy for genetic recombination, Omega 33 (2) (2005) 255–265.
[19] S. Mohagheghi, A. Engelbrecht, M. Omran, Empirical analysis of self-adaptive differential evolution, Oper. Res. 183 (2007) 785–804.
[20] J. Clegg, J.F. Dawson, S.J. Porter, M.H. Barley, The use of a genetic algorithm to optimize the functional from of a multi-dimensional polynomial fit to experimental data, in: IEEE Congress on Evolutionary Computation, vol. 1, 2005, pp. 928–934.
[21] J.X. Zhou, W.D. Yang, Q. Li, Improved ant colony algorithm and simulation for continuous function optimization, Syst. Simul. 4 (2009) 1685–1688.
[22] X.J. Lei, Z.K. Shi, Application of particle swarm optimization method in function optimization and parameter analysis, Comput. Eng. Appl. 44 (2008) 53–54.
[23] P.A. Makris, C.G. Provatidis, D.T. Venetsanos, Structural optimization of thin-walled tubular trusses using a virtual strain energy density approach, Thin-Walled Struct. 44 (2006) 235–246.
[24] K.S. Lee, Z.W. Geem, A new structural optimization method based on the harmony search algorithm, Comput. Struct. 82 (2004) 781–798.
[25] R. Sedaghati, Benchmark case studies in structural design optimization using the force method, Solids Struct. 42 (2005) 5848–5871.
[26] R. Sedaghati, E. Esmailzadeh, Optimum design of structures with stress and displacement constrains using the force method, Mech. Sci. 45 (2003) 1369–1389.
[27] A. Kaveh, H. Rahami, Analysis, Design and optimization of structures using force method and genetic algorithm, Numer. Methods Eng. 65 (2006) 1570–1584.
[28] A. Kaveh, H. Rahami, Nonlinear analysis and optimal design of structures via force method and genetic algorithm, Numer. Methods Eng. 84 (2006) 770–778.
[29] D.J. Krusienski, W.K. Jenkins, Adaptive filtering via particle swarm optimization, in: 37th IEEE Asilomar Conference on Signals, vol. 1, 2003, pp. 571–575.
[30] Y.M. Li, M. Wang, L.J. Cui, D.M. Huang, A new classification arithmetic for multi-image classification in genetic programming, in: 6th International Conference on Machine Learning and Cybernetics, vol. 3, Hong Kong, 2007, pp. 1683–1687.
[31] H. Evans, M. Zhang, Particle swarm optimisation for object classification, in: 23rd International Conference Image and Vision Computing, New Zealand, 2008, pp. 1–6.
[32] N. Veček, M. Mernik, M. Črepinšek, A chess rating system for evolutionary algorithms: a new method for the comparison and ranking of evolutionary algorithms, Inf. Sci. 227 (2014) 656–679.
[33] K.Z. Tang, T.K.Y.J.Y. Sun, An improved genetic algorithm based on a novel selection strategy for nonlinear programming problems, Comput. Chem. Eng. 35 (2011) 615–621.
[34] H. Muehlenbein, D. Schlierkamp-Voosen, Modified differential evolution: a greedy random strategy for genetic recombination, Evol. Comput. 1 (1) (1993) 25–49.
[35] A. Amirjanov, The development of a changing range genetic algorithm, Comput. Methods Appl. Mech. Eng. 195 (2006) 2495–2508.
[36] C. Sivapathasekaran, S. Mukherjee, A. Ray, A. Gupta, R. Sen, Artificial neural network modeling and genetic algorithm based medium optimization for the improved production of marine biosurfactant, Bioresour. Technol. 101 (2010) 2884–2887.
[37] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[38] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004, pp. 1–305, ISBN:978-0-262-04219-2.
[39] Amarjeet, J.K. Chhabra, FP-ABC: fuzzy Pareto-dominance driven artificial bee colony algorithm for many-objective software module clustering, Comput. Lang. Syst. Struct. (2017), http://dx.doi.org/10.1016/j.cl.2017.08.001 (in press).
[40] V. Arora, R. Bhatia, M. Singh, Synthesizing test scenarios in UML activity diagram using a bio-inspired approach, Comput. Lang. Syst. Struct. 50 (2017) 1–19.
[41] U. Kirchmaier, S. Hawe, K. Diepold, A swarm intelligence inspired algorithm for contour detection in images, Appl. Soft Comput. 13 (2013) 3118–3129.
[42] B. Akay, A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding, Appl. Soft Comput. 13 (2013) 3066–3091.
[43] W.N. Chen, Y. Zhang, Y. Lin, N. Chen, Z.H. Zhan, H.S.H. Chung, Y. Li, Y.H. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.

[44] Z.H. Zhan, J. Zhang, Y. Li, H.S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. Evol. Comput. 39 (6) (2009) 1362–1381.
[45] A. Qin, X. Li, Differential evolution on the CEC-2013 single-objective continuous optimization tested, in: IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 2013, pp. 1099–1106.
[46] R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679–1696.
[47] K. Sørensen, Meta-heuristics: the metaphor exposed, Oper. Res. 22 (2015) 3–18.
[48] A. Qin, V. Huang, P. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[49] A. Draa, On the performances of the flower pollination algorithm – qualitative and quantitative analyses, Appl. Soft Comput. 34 (2015) 349–371.
[50] M. Črepinšek, S.-H. Liu, L. Mernik, M. Mernik, Is a comparison of results meaningful from the inexact replications of computational experiments? Soft Comput. 20 (2016) 223–235.
[51] M. Mernik, S.-H. Liu, D. Karaboga, M. Črepinšek, On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation, Inf. Sci. 291 (2015) 115–127.
[52] Y. Lee, J.J. Filliben, R.J. Micheals, P.J. Phillips, Sensitivity analysis for biometric systems: a methodology based on orthogonal experiment designs, Comput. Vis. Image Underst. 117 (2013) 532–550.
[53] S.-H. Liu, M. Mernik, D. Hrnčič, M. Črepinšek, A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model, Appl. Soft Comput. 13 (9) (2013) 3792–3805.
[54] J.Q. Yu, O.K. Li, A social spider algorithm for global optimization, Appl. Soft Comput. 30 (2015) 614–627.
[55] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.
[56] S. Das, P.N. Suganthan, Differential evolution: a survey of the sate-of-the-art, Evol. Comput. 15 (1) (2011) 4–31.
[57] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, Evol. Comput. 13 (5) (2009) 945–958.
[58] C. Garcia-Martinez, M. Lozano, F. Herrera, D. Molina, A. Sanchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, Eur. J. Oper. Res. 185 (2008) 1088–1113.
[59] J.J. Liang, B.-Y. Qu, P.N. Suganthan, A.G. Hernndez-Daz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization, Tech. Rep. 201212, Computational Intelligence Laboratory, Zhengzhou University/Nanyang Technological University, Zhengzhou China/Singapore, 2013.
[60] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Tech. Rep. 201311, Computational Intelligence Laboratory, Zhengzhou University/Nanyang Technological University, Zhengzhou China/Singapore, 2014.
[61] J. Derrac, S. Garcia, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (2011) 3–18.
[62] M.E. Glickman, Example of the Glicko-2 System, Boston University, 2012.