The Journal of Supercomputing (2022) 78:3998–4031
https://doi.org/10.1007/s11227-021-04015-9

A hybrid OBL-based firefly algorithm with symbiotic organisms search algorithm for solving continuous optimization problems

Mina Javanmard Goldanloo[1] · **Farhad Soleimanian Gharehchopogh**[1]

Accepted: 5 August 2021 / Published online: 18 August 2021

Abstract

The metaheuristic optimization algorithms are relatively new optimization algorithms introduced to solve optimization problems in recent years. For example, the firefly algorithm (FA) is one of the metaheuristic algorithms inspired by the fireflies' flashing behavior. However, its weakness in terms of exploration and early convergence has been pointed out. In this paper, two approaches were proposed to improve the FA. In the first proposed approach, a new improved opposition-based learning FA (IOFA) method was presented to accelerate the convergence and improve the FA's exploration capability. In the second proposed approach, a symbiotic organisms search (SOS) algorithm improved the exploration and exploitation of the first approach; two new parameters set these two goals, and the second approach was named IOFASOS. The purpose of the second method is that in the process of the SOS algorithm, the whole population is effective in the IOFA method to find solutions in the early stages of implementation, and with each iteration, fewer solutions are affected in the population. The experiments on 24 standard benchmark functions were conducted, and the first proposed approach showed a better performance in the small and medium dimensions and exhibited a relatively moderate performance in the higher dimensions. In contrast, the second proposed approach was better in increasing dimensions. In general, the empirical results showed that the two new approaches outperform other algorithms in most mathematical benchmarking functions. Thus, The IOFASOS model has more efficient solutions.

Keywords Symbiotic organisms search · Firefly algorithm · Continuous optimization problem · Optimization

✉ Farhad Soleimanian Gharehchopogh
bonab.farhad@gmail.com

[1] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran

1 Introduction

Optimization is a commonly encountered problem in all engineering disciplines; moreover, there is a wide range of optimization problems. The goal of solving optimization problems is to find the best possible solution or near-optimal solutions. The idea of near-optimal solutions has been developed over a short time with exact and approximate algorithms [1, 2]. Exact algorithms can accurately find the optimal solution; however, these algorithms cannot solve complex and Np-hard problems. Moreover, in the case of an increase in the dimensions of Np-hard and complex problems, their execution time increases exponentially concerning the problem. Therefore, approximate algorithms have been developed to overcome exact algorithms' problems; they can find the near-optimal solution at much less time to solve Np-hard and complex problems. The approximate algorithms are divided into heuristic and metaheuristic algorithms; heuristic and metaheuristic algorithms use approximate methods for solving optimization problems.

In most cases, heuristic algorithms seek the right solution in near-optimal insufficient computational time; however, heuristic algorithms do not guarantee an optimal solution. The main disadvantages of heuristic algorithms include producing a solution or a limited number of solutions, trapping in a local optimum, and early convergence. Metaheuristic algorithms have been developed to overcome the heuristic algorithms' weaknesses, such as trapping local optima and early convergence [3–10]. Each metaheuristic algorithm uses a specific method to escape the local optima or avoid trapping there. As a result, metaheuristic algorithms can solve optimization problems and find high-quality solutions for all optimization problems without trapping in local optimal [11–14].

Typically, the strategy of metaheuristic algorithms is to improve step by step as the search begins with a random structure from one or more points in the search space. An optimal layout operation is then used to find a new structure that further improves the objective function. Then, this improved structure is reconsidered as a new system structure, and this process continues to achieve the best improvement for the system.

In the last decade, metaheuristic optimization algorithms were more commonly used in engineering applications to solve Np-hard and complex optimization problems than other algorithms [11–14]. Because the metaheuristic optimization algorithms have the following advantages: (1) have relatively simple concepts and their implementation is easy; (2) do not need derivative data of the objective function; (3) can overcome local optima; (4) can be applied in a wide variety of Np-hard and complex optimization problems. Metaheuristic methods attempt to solve the various Np-hard and complex optimization problems using inspiration from nature and evolution. The nature-inspired metaheuristic algorithms solve the optimization problems by imitating biological or physical phenomena. These algorithms categorize into three main categories: evolution-based methods, physics-based methods, and congestion-based methods. All of the metaheuristic optimization algorithms, despite their nature, have several standard features [15, 16].

Yang (2008), inspired by the fireflies' flashing behavior, developed the FA [17]. FA considers as one of the successful and low-cost algorithms. Each individual in a group of fireflies moves toward a point where his best personal experience has occurred. The SOS algorithm [18] is also one of the most recent methods for solving optimization problems based on organisms' interactions in nature. This algorithm has three natural processes of Mutualism, Commensalism, and Parasitism that can be used as new processes to improve other algorithms.

The differential evolution (DE) algorithm is suitable because of mutation operators and hybridization with other metaheuristic algorithms to improve them. Therefore, in [19], Zhang et al. presented a new FA through hybridization of the DE algorithm operators for global optimization, which combines the FA and the DE algorithm's advantages to provide a robust hybrid algorithm for solving optimization problems. In [20], a hybrid algorithm has also been developed based on the FA and the DE algorithms for global optimization; the proposed method design covers the FA's weaknesses using the DE algorithm. Finally, a new adaptive hybrid evolutionary FA is proposed in [21] to optimize truss structures' size and shape under multiple frequency constraints. The proposed algorithm in this paper is a hybridization of the DE algorithm and FA. The results showed that the proposed method's convergence rate was significantly improved than the DE algorithm and FA and multiple literature approaches.

In [22], a hybrid model was introduced to improve the FA's performance by introducing automated learning to regulate firefly behavior and using a genetic algorithm (GA) to increase global search and generate new solutions. In another study, a hybridization of GA and FA has been proposed to optimize complex problems and search for a more accurate global solution. In this hybrid algorithm, the GA searches the global minimum solution space, and the FA improves the precision of the potential candidate's solution [23]. Rahmani and MirHassani proposed a hybrid optimization method called the hybrid FA and GA [24]. This method combines the discrete FA with the GA. So, in this paper, the authors present a detailed description of the problem and adapt the algorithm. In this study, a discrete FA is proposed to solve the mechanical problem of prepared facilities. Also, an improved method based on the GA is used to transform a test solution into a better solution.

A chaotic adaptive FA has been proposed for optimization problems of the mechanical design in [25], in which fewer computations are carried out with the help of firefly adaptive motions. Additionally, using comparative parameters, the search accuracy is higher, which results in better target values; therefore, it has been proved that improving the performance of the FA is possible with the help of chaotic functions. The most crucial drawback of this model is the scatter in the search space and the failure to discover the optimal points with low iterations. In [26], various advantages of using particle swarm optimization (PSO) and FA are proposed with three new operators in a hybrid optimizer, called FAPSO, based on these two algorithms. In the first step, the proposed method's population is divided into two selective sub-populations of the FA and the PSO as the initial algorithm to perform the optimization process. To exchange information among the two sub-populations, and then efficient use of the advantages of PSO and FAs, these sub-populations share their optimal solutions, while they have more than one predefined threshold, which encounters recession. In the second step, every single dimension of the search space is divided into many small sub-areas, based on which historical knowledge is recorded to help the best current solution to detect an operator. The most crucial drawback of the hybrid model is the overpopulation and the increasing time complexity.

In [27], the hybridization of the FA and the social spider algorithm (SSA) is presented by Gupta and Arora for the multi-purpose optimization function. The notion of hybridization is proposed in this model to tackle optimization issues that make optimal use of the concepts of exploration and exploitation throughout the search space. The proposed algorithm is compiled in this paper by combining the FA and the SSA's biological processes. The proposed algorithm in this paper was tested on standard benchmark problems and then compared with the FA and the SSA. The results showed that this paper's proposed algorithm outperforms the firefly and the SSAs on the benchmark functions.

In [28], a new optimization model, called CLA-FA, is suggested. This new model combines cellular learning automata (CLA) and the firefly algorithm (FA). In the proposed model, each dimension of search space is assigned to one cell of CLA, and in each cell, a swarm of fireflies is located, which have the optimization duty of that specific dimension. The experimental results showed that the proposed method could be practical to find the global optima and improve the global search and the exploration rate of the standard FA. The hybrid FA is also used in the image processing section. The most crucial drawback of this model is the lack of adjustment of FA parameters and failure to achieve the best optimal solution. A hybrid FA and fuzzy c-mean algorithm are proposed to fragment MRI brain images [29]. It was proven to be efficient in promoting the clustering function of the traditional fuzzy c-mean algorithm. The simulation results indicated the proposed algorithm's superiority to other fragmentation methods in quantitative and qualitative terms.

In [30], a hybridized WOA with GWO, WOAGWO, is proposed. Simulations are tested on three standard test functions called benchmark functions: 23 common functions, 25 CEC2005 functions, and 10 CEC2019 functions. Results showed that WOAGWO achieved optimum solution, which is better than WOA and GWO. Furthermore, the improved raven roosting optimization algorithm (IRRO) and the CSO algorithm are combined in a hybrid (IRRO–CSO) methodology [31]. The CSO algorithm is chosen for its efficiency in balancing local and global search, while the IRRO algorithm is chosen to solve premature convergence and its superior performance in larger search areas. Finally, the proposed hybrid IRRO–CSO algorithm is compared to various imitation-based swarm intelligence algorithms (CEC2017) using benchmark functions. Compared to the following algorithms: WOA, GWO, CSO, BAT, and PSO, the findings from implementing the hybrid IRRO–CSO algorithm in MATLAB revealed an improvement in the average best fitness.

Some algorithms with several limitations were discussed in the related work. These algorithms cannot provide optimal solutions due to local optimization. Demographic diversity and the lack of an optimal population were important limitations in the literature algorithms. Convergence is an essential factor that allows an algorithm to approach the optimal solution, but the described algorithms lacked optimal convergence.

This paper presents two approaches using the SOS algorithm processes and the opposite-based learning (OBL) method to improve exploration and exploitation, increase data-sharing, and avoid trapping in the FA's local optima.

The goal of the OBL strategy is to increase the efficiency of metaheuristic algorithms. OBL refers to candidate solutions generated by a random iteration scheme and their opposite solutions in different areas of the search space.

The main contributions of this paper are as follows:

- Improving exploration and exploitation operations using OBL
- Improving the firefly algorithm using OBL
- Hybridizing IOFA and SOS to discover optimal solutions
- Evaluation of the proposed models on 24 different mathematical functions.

The rest of the paper has been structured as follows: Sect. 2 describes the FA and the SOS algorithm, and in Sect. 3, two different approaches are proposed to improve the FA in solving optimization problems. Then, Sect. 4 shows experiments and experimental results. Finally, conclusions and suggestions for future work are presented in Sect. 5.

2 Materials and methods

2.1 Firefly algorithm

The FA was presented by Xin-She Yang in 2007 at the University of Cambridge based on the flashing behavior and patterns of fireflies in nature [17]. FA is a metaheuristic algorithm inspired by nature and used in almost all engineering and Np-hard problems optimization. It is a stochastic algorithm. That is, a random search is used to find a set of solutions. At its most basic level, the FA focuses on generating solutions within a search space and choosing the best survival solution. A random search avoids getting trapped in local optima. For metaheuristic algorithms, exploration means discovering various solutions within the search space, while exploitation means the search process focuses on the best adjacent solutions. Figure 1 presents the pseudo-code of the FA.

The rate and mode of illumination and the time interval between the sent optical signals attract the two sexes to each other. The intensity of light, I, decreases with increasing distance (R) from the light source. Three essential features of the FA are [32]:

a. The firefly becomes brighter and more attractive when it moves randomly, and all fireflies are of the same sex.
b. The attractiveness of the fireflies is proportional to the brightness of the light and the distance from it. The light absorption coefficient γ calculates the reduction in light intensity. The value of the objective function also determines the luminance of the firefly.

```
Begin
01: Generate initial population of fireflies $X_i$ (i=1... N);
02: Initialize parameters: $\alpha$, $\gamma$, $\beta_{min}$, and t=0, and Fes = 0;
03: Brightness $l_i$ at $X_i$ is determined by f ($X_i$);
04: Define light absorption coefficient $\gamma$;
05: While (not meet the stop conditions)
06:   For i=1: N all N fireflies
07:     For j=1: N all N fireflies
08:       If $I_j > I_i$ Then
09:         Move firefly i towards j in all dimensions according to Equation (2);
10.       End If
11:       Attractiveness varies with distance;
12.       Evaluate the new solution and update its brightness; Fes=Fes+1;
13:     End For
14:   End For
15: Rank the fireflies and find the current best;
16: t=t+1;
17: End While
18: Post-process results.
End
```

Fig. 1 Pseudo-code of the FA

c. The distance between fireflies is obtained from Eq. (1), so that $X_{i,k}$ is the kth part of the spatial coordination and *i*th firefly.

$$r_{i,j} = \sqrt{(xi - xj)^2 + (yi - yj)^2} \tag{1}$$

The movement of the firefly and being attracted toward a brighter firefly is determined by Eq. (2):

$$X_i = X_i + B_0 e^{rij^2}\left(X_j - X_i\right) + a(rand - 1/2) \tag{2}$$

a is a randomizer variable, a rand is a random number obtained between [0, 1], and B indicates the light source's attractiveness. The attractiveness changes determine the parameter γ.

FA has two inner loops when going through the population n and one outer loop for iteration t. Therefore, the complexity at the extreme case is $O(n^2 t)$. As n is small (typically, n = 30), and t is large (for example, t = 5000), the computation cost is relatively inexpensive because the algorithm complexity is linear in terms of t.

One of the disadvantages of the firefly algorithm is its low detection rate. Because members of the population are always moving in the same direction in this algorithm, agents do not explore the entire search space. Also, search agents do not precisely follow the best person in the population due to the law of light intensity.

2.2 Symbiotic organisms search

Cheng and Prayog [18] proposed the SOS algorithm in 2014. The SOS algorithm is one of the newest methods for solving optimization problems based on organisms' interaction in nature. Organisms rarely live in isolation due to their reliance on other species for nutrition and even survival. This relationship is based on trust, which is known as a symbiotic relationship. The SOS algorithm begins with an initial population called the ecosystem. In the initial ecosystem, a group of organisms is generated randomly in a region. Each organism represents a proposed solution for the related problem. The SOS algorithm provides a new solution based on imitating the biological relationships between the two living organisms in an ecosystem. Figure 2 shows the SOS algorithm's pseudo-code, including three types: Mutualism, Commensalism, and Parasitism of biological relationships in nature [33–36]. The time complexity in the SOS algorithm is n^2.

2.2.1 Mutualism

At this point, both organisms benefit from the relationship they have with each other. Consider honeybees' relationship with flowers, for example, where the bees fly through the flowers to collect the nectar required for producing honey. It is also beneficial to the flowers because bees, scattering the pollen, facilitate their pollination [18]. In the SOS algorithm, Xi is an organism corresponding to the ith individual in the ecosystem. The next organism, Xj, is randomly selected and associated with the Xi in the ecosystem. Finally, Xi and Xj are updated in the Mutualism phase according to Eqs. (3) and (4):

$$X_{i\text{new}} = \mathrm{X_i} + \text{rand }(0,1) \times \left(X_{\text{best}} - \text{Mutual\_Vector } \times BF_1\right) \tag{3}$$

$$X_{j\text{new}} = \mathrm{X_j} + \text{rand }(0,1) \times \left(X_{\text{best}} - \text{Mutual\_Vector } \times BF_2\right) \tag{4}$$

$$\text{Mutual\_Vector} = \left(\mathrm{X_i} + \mathrm{X_j}\right)/2 \tag{5}$$

In Eqs. (3) and (4), rand (0,1) is a random vector of numbers. BF1 and BF2 are the profit factor of Xi and Xj, which show each organism's profit. In Eq. (3),

```
1. Initialization (initial ecosystem, set ecosystem size, and maximum iteration)
2. For counter=1 to maximum iteration
3.     For each organism in the ecosystem
4.         Mutualism Phase according to Equations (3,4)
5.         Commensalism Phase according to Equation (4)
6.         Parasitism Phase
7.         Update the best organism
8.     End For
9. End For
```

Fig. 2 The pseudo-code of the SOS algorithm

Mutual_Vector represents the relationship between the Xi and Xj. The Mutual_Vector * BF2 in Eqs. (33) and (4) is the attempt to increase the survival of living organisms. Given the Darwinian theory, i.e., the survival of the fittest, all organisms have to increase their compatibility level with their surroundings. Here, the Xbest represents the highest level of compatibility.

2.2.2 Commensalism

At this stage, one of the organisms benefits, and the other gains nothing from the relationship. For example, consider the relationship between sticky fish and sharks. In this relationship, the sticky fish sticks to the shark and feeds on the remaining food; the shark gains very little or no benefit [18]. As in the Mutualism phase, the organism Xj is randomly selected in nature and is associated with Xi. In this case, Xi strives to gain profit, but Xj gets no benefit or loss. Thus, Xi is updated by Eq. (6).

$$X_{i\mathrm{new}} = \mathrm{X_i} + \mathrm{rand}\,(-1, 1) \times \left(X_{\mathrm{best}} - \mathrm{X_j}\right) \tag{6}$$

where $\mathrm{X_{best}}$-$\mathrm{X_j}$ represents the benefit provided by $\mathrm{X_j}$ to increase the survival of $\mathrm{X_i}$.

2.2.3 Parasitism

At this stage, one organism benefits from the relationship, and the other endures a loss. Take as an example the Malaria blood parasite that the Malaria mosquito transmits to the human body. When the parasite proliferates in the human body, it may cause that person to die. In the SOS algorithm, the Xi, i.e., malaria mosquito, creates an artificial parasite called “Parasite-Vector.” The Parasite-Vector is created in the search space by replicating Xi. The Xj is selected randomly by the ecosystem and serves the parasite as its host. The Parasite-Vector attempts to take Xj’s place in the ecosystem. Both Xi and Xj are evaluated to measure their competency. When the Parasite-Vector destroys Xj and takes its place in the ecosystem, it reaches its highest competency. The highest competency for Xj is achieved when it resists the parasite, and the parasite cannot live any longer in the ecosystem.

2.3 Opposition-based learning

Opposition-based learning (OBL), the dual nature of the whole and the similarities of the opposites, is an algorithmic component inspired by the oriental philosophical issue [15, 37]. It includes generating some extra points using a hyper-rectangle and a central symmetry technique. Machine learning algorithms optimize the estimated solutions and search for an existing solution in large spaces. Typically, in obtaining the solution x for the given problem, they obtain an estimate of it as $\overline{x}$. It is not an accurate solution and can be based on experience or, in general, a random guess. In some cases, conditions are satisfied with estimating $\overline{x}$ Sometimes, try to reduce the difference between the estimated value and the optimal value known directly or indirectly. However, if the workflow process is known as a function approximation, there is a possibility of decreasing computational complexity, and it is possible to obtain an optimal solution [15].

In many cases, learning begins with a random point. For example, in artificial neural networks, the coefficients are initialized at random, in the genetic algorithm of the population parameters randomly; in the reinforcement learning algorithms, the learning agent is in a random and random selection algorithm. If the random conjecture of the optimal solution is not so far off, the quick results converge. Though it is pretty natural to initiate a random start from far-away points of the existing solutions, then at worst, the beginning in the "contrasting location" is the optimal solution, so the approximation, search, and optimization are considerably time-consuming, or in the worst-case mode. Logically, it must be searched simultaneously in all directions and the same direction. If x is looked for and is helpful in the direction of the shot, then the first step is to compute the opposite number $\overline{x}$.

If x∈R is an actual number defined on a given x∈[a, b] interval, then the opposite number $\overline{x}$ is defined as $\overline{x} = a + b - x$.

For example, if a = 0 and b = 1, the opposite number is equal to $\overline{x} = 1 - x$, and if a = − ∞ and b = + ∞, the opposite number is equal to $\overline{x} = -x$. This form of the opposite number is the usual symmetrical form of a number. In optimization problems, the OBL method uses contrasting numbers to search for the optimal point. Assume that $X = (x_1, x_2, \dots, x_n)$ is a search point in the next n space, and $x_i = \left[a_i, b_i\right]$ so that $i = 1,2, \dots, n$. The opposite number $X^* = (x_1^*, x_2^*, \dots, x_n^*)$ is calculated from Eq. (7).

$$x_i^* = a_i + b_i - x_i i = 1, 2, \dots, n \tag{7}$$

The principle of the OBL method in optimization based on X and X^* to search and find the optimal answers is that in each iteration, X^* is calculated from X and then f (X) and f (X^*) calculates as a fitness level from values of X and X^*, respectively. In iterations that $f(X) \geq f(X^*)$, X is considered as the answer vector. As a simple example of the OBL optimization process, the one-dimensional optimization state (n = 1) is shown in Fig. 3. In this figure, K is an iteration value.

As shown in Fig. 3, the growing number of iterations in the OBL method based on the search range should be reduced recursively. Then, as a sample in the first iteration, X is selected as candidate mode. The correct limit of the range reduces to $b_1' = {}^{b_1}/_2$. Also, if X^* is selected as a candidate mode, the left limit of the interval is

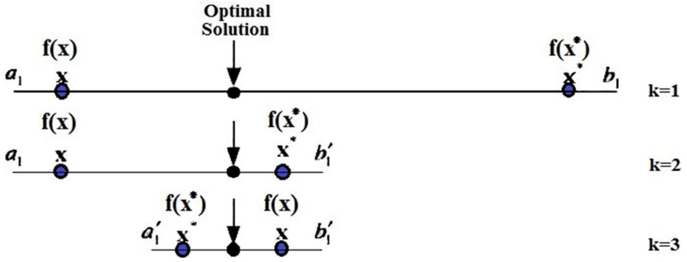

Fig. 3 Using the OBL method in optimizing a single one-dimensional problem

$a_1' = \frac{a_1}{2}$. This process eventually becomes convergent if X and X^* are close enough. In optimization problems, because solutions are continually changing, an excellent way to maintain the population is critical. Hence, by applying OBL, variety can be carried out and improves the metaheuristic algorithm's performance [38–40].

2.4 Proposed method

Based on studies in Sect. (2), it can be claimed that the FA can be hybridized with other metaheuristic algorithms in terms of the biologic process to use its advantages in exploration and exploitation effectively. Therefore, an algorithm with vital biological processes was needed to improve the FA and use exploration and exploitation effectively. The SOS algorithm is advantageous in solving optimization problems. In [41], the OBL method was used to overcome early convergence and prevent FA from getting trapped in local optima.

One way to improve accuracy and escape from local optimum points is to change the initial population of fırefly and the number of iterations of the fırefly algorithm. With the OBL method, the distribution of fıreflıes in the problem space increases, and the chances of finding global optimizations increase. Accordingly, the SOS algorithm and OBL method processes can be used to achieve the two main goals of this paper: removing weaknesses and increasing exploration and exploitation of the FA. This section is divided into two subsections to illustrate the two proposed approaches: In Sect. (3.1), the first proposed approach presents a new IOFA to accelerate the convergence and improve exploration in the FA. In Sect. (3.2), the second proposed approach (IOFASOS) uses the SOS algorithm processes to improve the exploration and exploitation of the first proposed approach; these two goals are set in the second approach with two new T parameters and U.

2.5 Improved OBL firefly algorithm (IOFA)

The FA may start with an inappropriate population and may not perform a significant exploration until the end. Indeed, several studies have proven that the OBL method increases population diversity [42], improves the convergence rate [43, 44], improves accuracy and speed [45], improves efficiency and quality of the solution, and improves search in search space and production of more precise solutions [46]. The opposition-based method has a high potential to accelerate convergence and improve exploration in the FA. The randomness of the FA algorithm is eliminated using the OBL method, and only the optimal points are searched. Thus, the opposition-based method can be used during iteration to search the opposite space and generate a suitable initial population for the FA. The second proposed approach uses both opposition-based models to accelerate the convergence and exploration in the FA. In [41], there is also a version of the opposition-based method to improve the FA, which overcomes the FA's early convergence problem by replacing the worst firefly. An entirely different version of the opposition-based method can be used, which will be outlined later outline. First, generating an appropriate initial population for the FA using an opposition-based method will be discussed. Figure 4 shows a flowchart of the proposed method.

The FA begins by randomly generating an initial population and then attempts to move toward an optimal solution. This step is called the initialization of the population, which is initialized according to Eq. (8).

$$\text{FirstRandomFirefly}(FRF) = \left[\begin{array}{cccc|c} xrand_1^1 & xrand_2^1 & \dots & xrand_n^1 & f\left(xrand^1\right) \\ x\text{rand}_1^2 & x\text{rand}_2^2 & \dots & x\text{rand}_n^2 & f\left(x\text{rand}^2\right) \\ \vdots & \dots & \dots & \dots & \vdots \\ x\text{rand}_1^{Npop} & x\text{rand}_2^{Npop} & \dots & x\text{rand}_n^{Npop} & f\left(x\text{rand}^{Npop}\right) \end{array}\right] \tag{8}$$

$$x\text{rand} = \text{rand}(\text{Min},\ \text{Max}, D) \tag{9}$$

In Eq. (8), FRF is the primary firefly population, randomly generated according to the xrand function on a given whole interval. The xrand function is defined according to Eq. (8), where Min represents the minimum possible limit for a variable, Max represents the maximum possible limit for a variable, and D is the dimension of the optimization problem. Once the initial population is randomly generated, each solution's opposition must be determined according to the opposition-based method. Before explaining the generation of opposition-based populations, some definitions should be clarified.

Definition 1: If x is an actual number on the interval [a b], its mirror number is represented by $\breve{p}$ and defined as Eq. (10) [47].

$$\breve{x} = a + b - x \tag{10}$$

Definition 2: If p(x1,x2,…,xn) is a point in the n-dimensional space, where xi ∈ [ai, bi] is an actual number, then its mirror number is represented by $\breve{P}$ ($\breve{x}$1,$\breve{x}$2,…,$\breve{x}$n) and defined as Eq. (11) [47]:

$$\breve{x}_i = a_i + b_i - x_i i = 1, 2, ..., n \tag{11}$$

Therefore, Eqs. (10) and (11) describe how opposition-based solutions are generated. One can define an initial IOFA population according to Eq. (11).

$$\text{Opposition - based FRF (OFRF)} = \left[\begin{array}{cccc|c} xop_1^1 & xop_2^1 & \dots & xop_n^1 & f\left(xop^1\right) \\ xop_1^2 & xop_2^2 & \dots & xop_n^2 & f\left(xop^2\right) \\ \vdots & \dots & \dots & \dots & \vdots \\ xop_1^{Npop} & xop_2^{Npop} & \dots & xop_n^{Npop} & f\left(xop^{Npop}\right) \end{array}\right] \tag{12}$$

$$xop = \text{Min}(x)_i + \text{Max}(x)_i - \text{FRF}(x_i)i = 1, 2, \dots, D \tag{13}$$

In Eq. (12), OFRF represents the IOFA population generated according to the fireflies' initial random population, where each opposition-based individual is calculated using xop. The xop variable is also defined based on Eq. (13). In this way, FA has two populations of solutions to start with; one of the populations, i.e., the main one, is named FRF, filled by random functions, and the second population is called

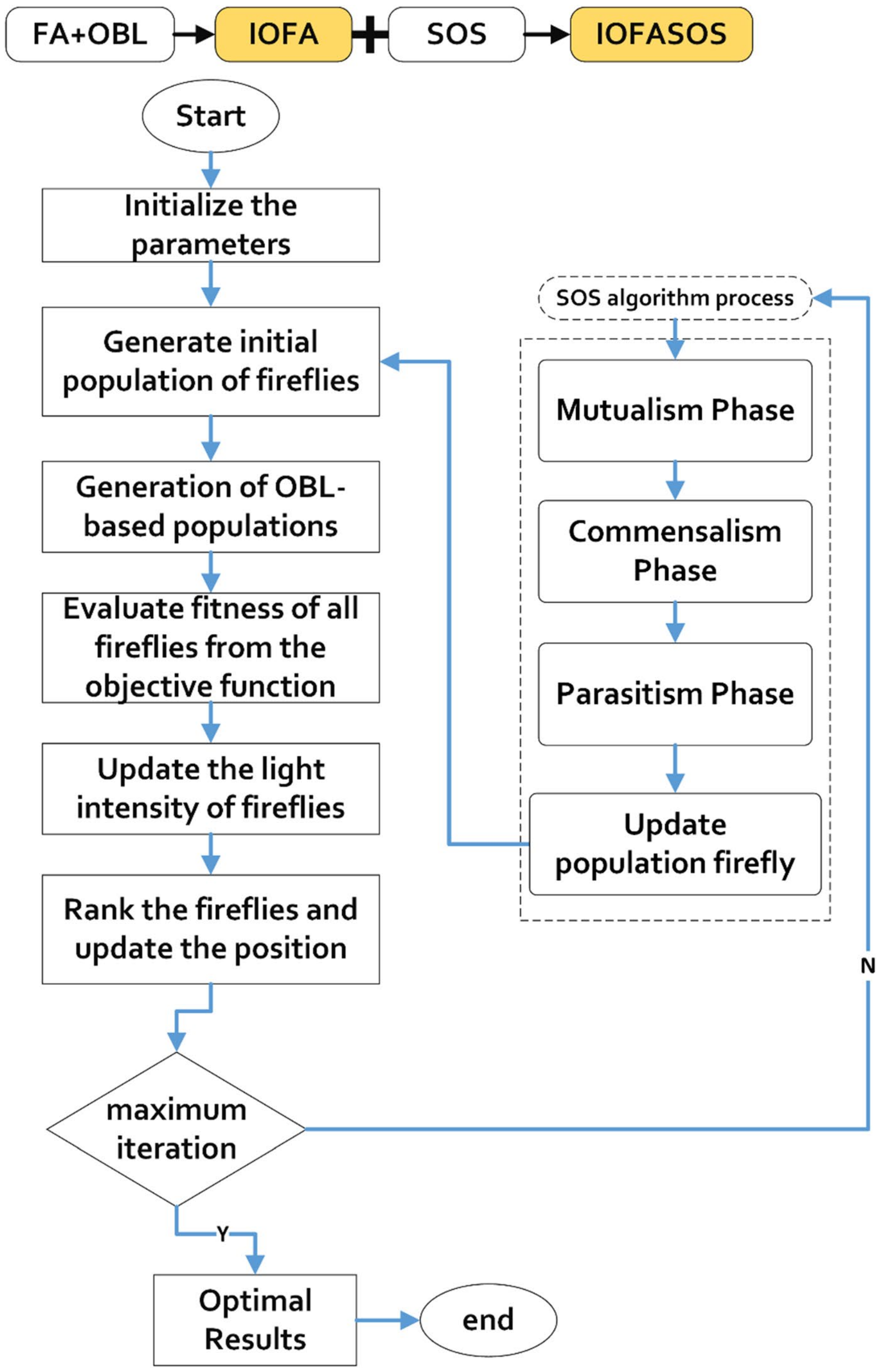
FA+OBL
IOFA
SOS
IOFASOS
Start
Initialize the parameters
Generate initial population of fireflies
Generation of OBL-based populations
Evaluate fitness of all fireflies from the objective function
Update the light intensity of fireflies
Rank the fireflies and update the position
maximum iteration
Y
N
Optimal Results
end
SOS algorithm process
Mutualism Phase
Commensalism Phase
Parasitism Phase
Update population firefly

Fig. 4 The flowchart of the proposed method

OFRF, which is generated by an opposition-based method. Given the objective function, if Fitness (xop) > = Fitness (x), then the solution available in the OFRF population can replace the solution achieved in the FRF population. Otherwise, the algorithm starts with the same solution available in the FRF population. An initial random population and its mirror population are evaluated simultaneously to use a more appropriate point as the starting point.

So far, the opposition-based method has been used to generate a suitable initial population and create convergence at the beginning of the FA; however, the FA may suffer inadequate exploration and exploitation during iteration. Hence, an opposition-based method was used to improve the FA's primary process in the first proposed approach. The FA has an essential process as shown in Eq. (2), where if a firefly is brighter than the other, the less bright one moves toward, the brighter firefly, and if there is a no brighter firefly, there not be any movement. In the second proposed method, if there is no brighter firefly, the firefly's mirror point compares to another firefly in terms of shine. The motion of the firefly and its attraction to the brighter opposite-based firefly are determined according to Eq. (14):

$$Xop_j = \mathrm{Min}\left(X_j\right) + \mathrm{Max}\left(X_j\right) - X_j \tag{14}$$

$$X_i = X_i + B_0 e^{rij^2}\left(Xop_j - X_i\right) + a(\mathrm{rand} - 1/2) \tag{15}$$

In Eq. (15), a is the randomizer variable; a rand is a random number between [0, 1]; B indicates the light source's attractiveness; and *rij* is the distance between fireflies. The changes in attractiveness determine the parameter γ. Xopi is a firefly generated based on the opposition-based method, according to Eq. (13). Thus, the improved opposition-based FA considering the changes pointed out in Fig. 5 is presented. According to the pseudo-code of Fig. 5, it can be stated that the time complexity in the proposed method is in the form of $(n^2 \times t) \times (n \times t) \times n^2$, where n is equal to the number of population and t is equal to iteration. The complexity of the FA and SOS algorithms is calculated based on OBL. The complexity of OBL is linear and does not increase complexity.

2.6 The Hybrid of IOFA and SOS (IOFASOS)

The first proposed approach presented an improved opposition-based FA to improve convergence rate and exploration capability. Nonetheless, the FA's primary process in Eq. (2) and its modified process in Eq. (2) may sometimes not maintain the balance between exploitation and exploration. Therefore, in this subsection, according to Fig. 5, three robust SOS algorithm processes were applied: Mutualism, Commensalism, and Parasitism, to cover the FA's potential weaknesses IOFA and to improve the balance between exploitation and exploration. A hybridization of IOFA and the SOS algorithm was created, and for balance between exploitation and exploration in the first proposed approach, there should be some set parameters. Thus, two new parameters, i.e., T and U, were used to control the number of fireflies altered by the SOS algorithm's natural process. In the following, these new parameters, how to

```
Begin
01: FRF= Generate an initial random population of fireflies according to equations (10,11);
02: OFRF= Evaluate the new population Opposition based method according to equations (14,15);
03: Update population firefly with OFRF
04: Initialize parameters: $\alpha$, $\gamma$, $\beta_{min}$, and t=0,
05: Brightness li at Xi determines by f ($X_i$);
06: Define light absorption coefficient $\gamma$;
07: While (not meet the stop conditions)
08:   For i=1: N all N fireflies
09:    For j=1: N all N fireflies
10:     If $I_j > I_i$, Then
11:       Move firefly i towards j in all dimensions according to Equation (2);
11:       Else if(Opposition($I_j$)> $I_i$)
11:       Replace firefly j with Opposition($I_j$)
11:       Move firefly i towards Opposition($I_j$) in all dimensions according to equation (16,17);
12.     End If
13:     Attractiveness varies with distance;
14.     Evaluate the new solution and update its brightness;
15:   End For
16:   End For
15: Rank the fireflies and find the current best;
16: t=t+1;
17: End While
18: Post-process results.
End
```

Fig. 5 The pseudo-code of the first proposed approach for IOFA

use them, and their impact on the first proposed approach have been outlined. The pseudo-code of the second proposed approach for improving the FA using the SOS algorithm to solve continuous optimization problems is presented in Fig. 6.

The advantages of the SOS algorithm over the FA algorithm include simplicity and the ability to control convergence speed. In the FA algorithm, attractiveness and distance affect convergence. However, they cannot easily control the convergence speed like the SOS algorithm. On the other hand, OBL in the FA algorithm is effective in increasing the convergence rate. Nevertheless, the main limitation of the FA algorithm is early convergence and getting caught up in local minima. Therefore, the best position of the worms should be changed in each iteration to prevent this from happening. To this end, it is possible to increase the diversity among members of the firefly population and reduce the likelihood of being caught in local minima by involving the operators of the SOS algorithm.

To conduct a balanced exploration and exploitation in IOFA, more exploration should be performed on the population at the beginning of the initial iterations. Then, by each iteration, the exploration should be reduced, and the exploitation should

```
Begin
01: FRF= Generate initial random population of fireflies according to equations (10,11);
02: OFRF= Evaluate the new population OBL according to equations (14,15);
03: Update population firefly with OFRF
04: Initialize parameters: α, γ, βmin, and t=0, and Fes = 0;
05: Brightness li at Xi is determined by f (Xi);
06: Define light absorption coefficient γ;
04: New parameter:
05: T=1;
06: U = [0,0.5]
07: While (not meet the stop conditions)
08:   For i=1: N all N fireflies
09:    For j=1: N all N fireflies
10:     If Ij > Ii Then
11:       Move firefly i towards j in all dimensions according to Equation (2);
11:       Else if(Opposition(Ij)> Ii)
11:       Replace firefly j with Opposition(Ij)
11:       Move firefly i towards Opposition(Ij) in all dimensions according to equations
(16,17);
12.     End If
13:     Attractiveness varies with distance;
14.     Evaluate the new solution and update its brightness; Fes=Fes+1;
15:   End For
16:   End For
18: New Section………………………………………………………
19: Update T Parameter according to Equation (8);
20: Update NE Parameter according to Equation (9);
21: If (NE>=4)
22: Ecosystem= Select NE of fireflies
23:     For each organism in the Ecosystem
24:           Mutualism Phase
25:           Commensalism Phase
26:          Parasitism Phase
27:          Update the best organism
28:      End For
29: End If
30: Update population firefly with Ecosystem population
15: Rank the fireflies and find the current best;
16: t=t+1;
17: End While
18: Post process results.
End
```

Fig. 6 The pseudo-code of the second proposed approach

increase. In this way, in the second proposed approach, the biological processes of the SOS algorithm should be applied to a more significant number of IOFA populations at first and a smaller number of IOFA populations in the end; that is, in the processes of the SOS algorithm, the entire firefly population in the IOFA is affected initially, and by each iteration, a smaller number of individuals in the population are affected. Thus, a parameter T was used in the second approach, the value of which was determined in iterations according to Eq. (16):

$$T = T * (T - U) \tag{16}$$

In Eq. (15), U is a number between [0, 0.5], which is determined initially; T's initial value is 1. Thus, in the second proposed approach, U's initial value is close to zero to reduce exploration and increase exploitation in each iteration. Once the value of T is calculated using the number of fireflies, according to Eq. (17),

$$\mathrm{NE} = \mathrm{Round}(T * N_{\mathrm{fireflies}}) \tag{17}$$

In Eq. (17), $N_{fireflies}$ is the number of fireflies in FA, and NE is the number of fireflies in IOFA, changed by the SOS algorithm's three processes. As a result, with each iteration, T's value is updated using Eq. (16). According to this equation, T's value decreases with each generation iteration, based on which the proposed method maintains the balance between exploration and exploitation. On the other hand, when the SOS algorithm processes are applied to the firefly, it somehow causes population diversity in the FA and changes its convergence rate. Note that there must be at least four solutions to the SOS algorithm's processes; therefore, we considered the condition value shown in Fig. 6. Hence, we presented two different approaches to improve the FA. Section 5 shows the results of both proposed approaches.

2.7 Experiment results

The two proposed approaches and other comparative algorithms were simulated using MATLAB on a system with a 5-core Intel processor, 2.30 GHz processing power, and 10 GB RAM. Twenty-four mathematical benchmark functions were presented for evaluation in Sect. (4.1). In addition, the two proposed approaches with other basic metaheuristic algorithms were implemented on benchmark functions to compare algorithms such as HS [48], ABC [49], FA [17], and SOS algorithm [18]. The results of comparisons of all the algorithms with the two proposed approaches are presented in the form of a diagram, and Sect. (4.2) presents the initial settings and parameters of the two proposed approaches, and comparative algorithms were initialized. Two experiments were conducted in subsection (4.3): In the first experiment, in each iteration, the best solution found for each function was graphically and separately illustrated by the algorithms mentioned in the literature. In the second experiment, the two proposed approaches and other algorithms were presented in the form of a table for comparison in terms of the best and the worst value of the objective function, the mean of the objective function for the total population, and other statistical parameters.

In this paper, four measurement criteria were used to evaluate the results. These measurements were the worst, best, mean fitness value, and standard deviation (SD). The worst, best, and mean fitness value of SD is mathematically defined as follows:

$$\text{Best} = \max_{i=1}^{t\text{Max}} BS_i \tag{18}$$

$$\text{Worst} = \min_{i=1}^{t\text{Max}} BS_i \tag{19}$$

$$\text{Mean} = \frac{1}{t\text{Max}} \sum_{i=1}^{t\text{Max}} BS_i \tag{20}$$

$$\text{SD} = \sqrt{\frac{\sum_{i=1}^{M} (BS_i - \mu)^2}{t\text{Max}}} \tag{20}$$

where t_{Max} and, moreover, BS are the maximum iteration and the best score obtained so far for each iteration, respectively.

2.8 Standard test functions

Standard mathematic functions, called benchmark functions, are used to evaluate all optimization and metaheuristic algorithms. The standard benchmark functions are introduced in [50] with a comprehensive description, evaluating the two proposed approaches. To evaluate the performance and comparison, we use 24 benchmark functions; Table 1 provides a general description of each function and its numerical range.

2.9 Initial parameters

It was necessary to set and initialize some of their fundamental parameters at the beginning, including the population, the dimension of each solution, the minimum value of each dimension, the maximum value of each dimension, and the number of generation iteration of each algorithm. In all the simulations and comparisons of the metaheuristic algorithms, some of the algorithms' quantitative parameters were considered equal to an appropriate comparison. The number of objective function evaluations is one of the criteria for which all algorithms must have the same value to study each algorithm's actual performance. The number of evaluations for all algorithms was considered 2000 in the simulation section; moreover, the comparative algorithms' other quantitative and qualitative parameters are given in Table 2.

The previous studies used the same algorithms proposed in previous years to demonstrate the performance of a new model. Therefore, HS, ABC, FA, and SOS algorithms were used to compare and evaluate this paper.

2.10 Evaluation of the proposed methods

In this subsection, the two proposed approaches have been evaluated in two ways: In the first experiment, the two proposed approaches and other comparative algorithms were implemented with the same objective function evaluations. The results have been shown graphically. The graphic illustration shows only the best solution found so far based on objective function evaluations (3000 here). The second experiment is related to the algorithms' total populations' statistical criteria shown in Table 3. The best and the worst values of the objective function, the mean of the objective function for the total population, and other statistical parameters have been shown in tables. The two proposed approaches were then implemented on different benchmark functions, including 2D, 4D, 10D, and 30D functions. The reason for using different dimensions is to demonstrate how the two proposed approaches perform in different dimensions.

The results of the implementation of the two proposed approaches and other basic algorithms on ten 2D benchmark functions are shown in Figs. 7, 8, and 9,

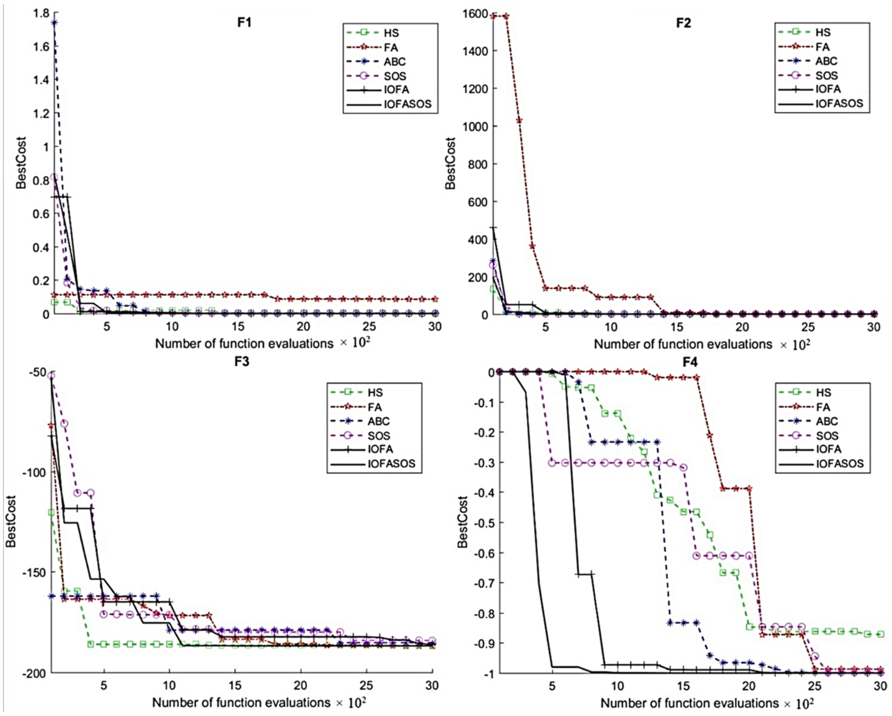

Fig. 7 The convergence rate of the two proposed approaches on functions 1–4 with two dimensions

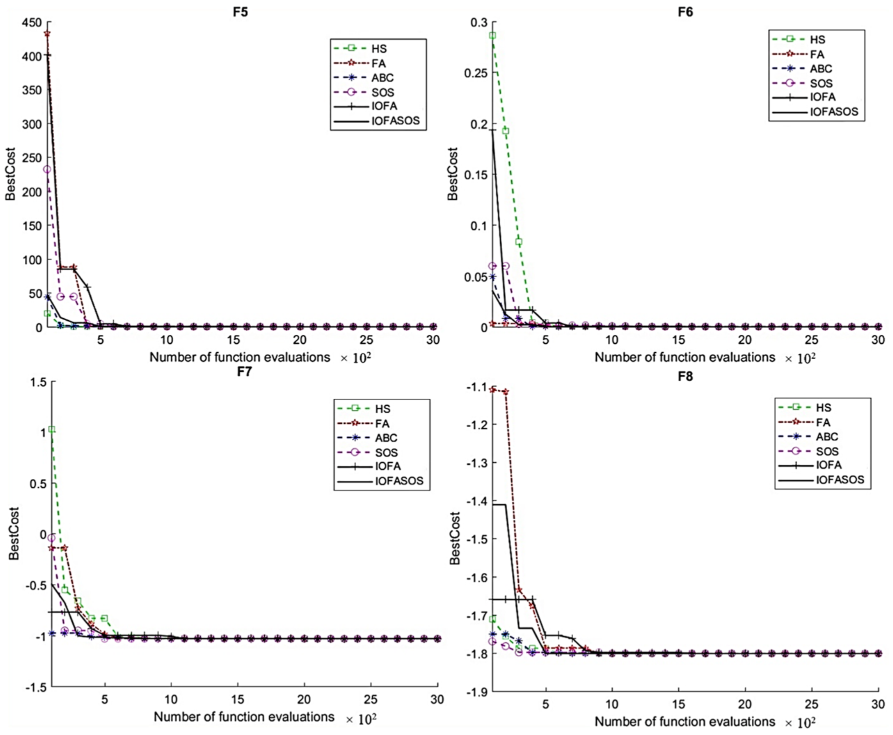

Fig. 8 The convergence rate of the two proposed approaches on functions 5–8 with two dimensions

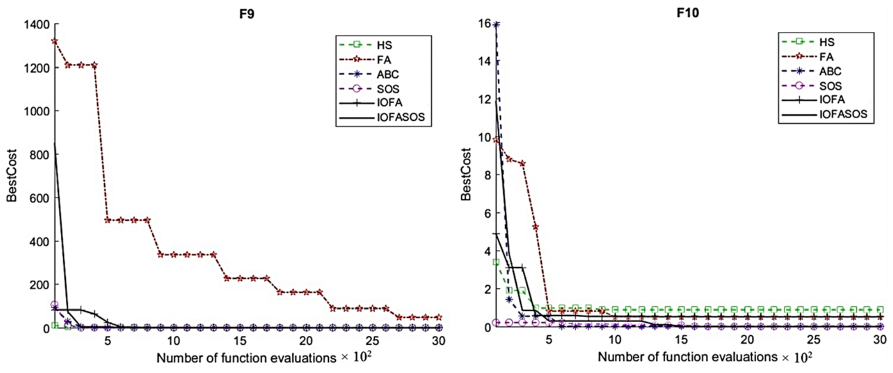

Fig. 9 The convergence rate of the two proposed approaches on functions 9–10 with two dimensions

respectively. Based on these results, both proposed approaches have a relatively better performance in terms of the convergence of two-dimensional benchmark functions compared to other metaheuristic algorithms. In general, the two proposed approaches, like all the metaheuristic algorithms, successfully solve small-dimensional problems.

Figure (7) shows that the convergence speed of the IOFASOS model on the F1, F2, F3, and F4 functions is higher than other algorithms. In the IOFA model, the search for optimization functions begins at the optimal points of the search space using OBL. On the other hand, the IOFASOS model uses SOS to intensify exploration operations and solutions to find the best spots in the search space.

The implementation of the two proposed approaches and other basic algorithms on four 4D benchmark functions is shown in Fig. 9 for comparison. Based on these results, it can be concluded that both proposed approaches outperformed other metaheuristic algorithms in terms of the convergence rate of 4D benchmark functions. However, Fig. 10 generally shows that implementing the second proposed approach on 4D benchmark functions is more satisfactory.

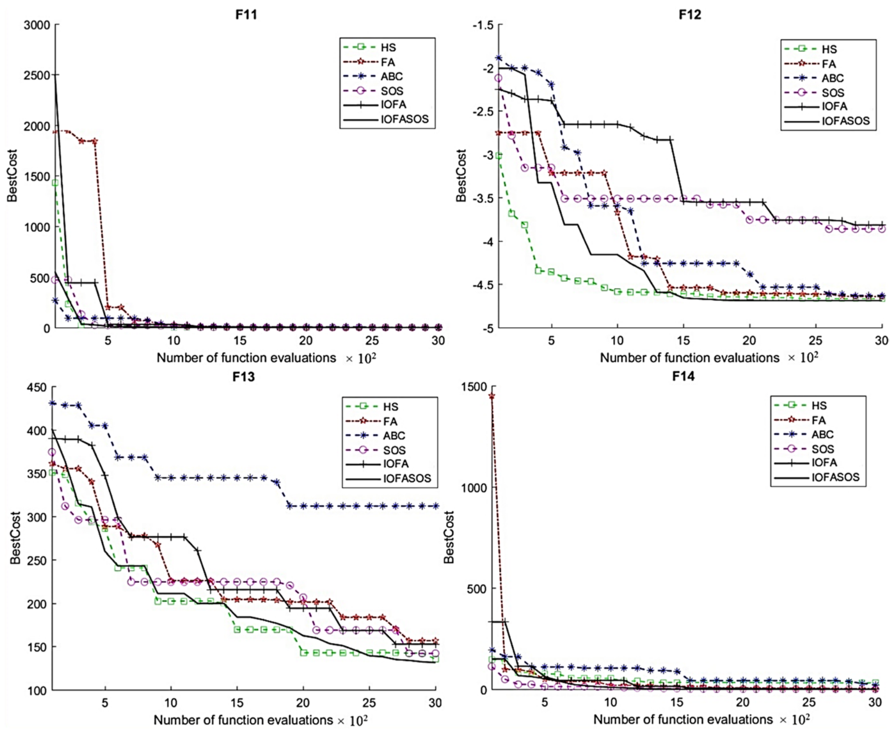

Fig. 10 The convergence rate of the two proposed approaches on 4D benchmark functions 11–14

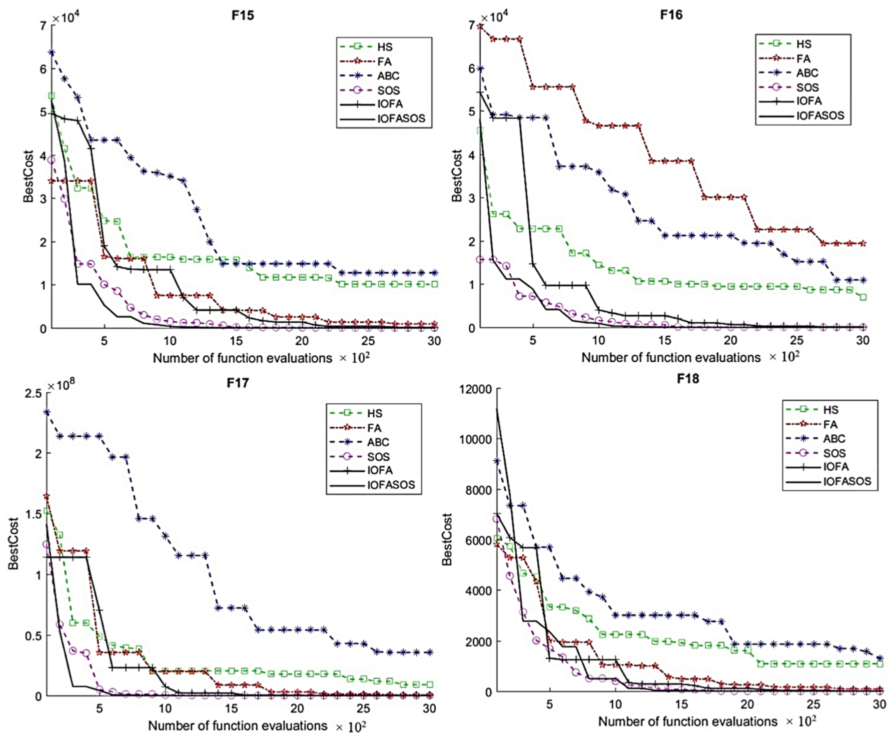

Fig. 11 The convergence rate of the two proposed approaches on 10D functions 15–18

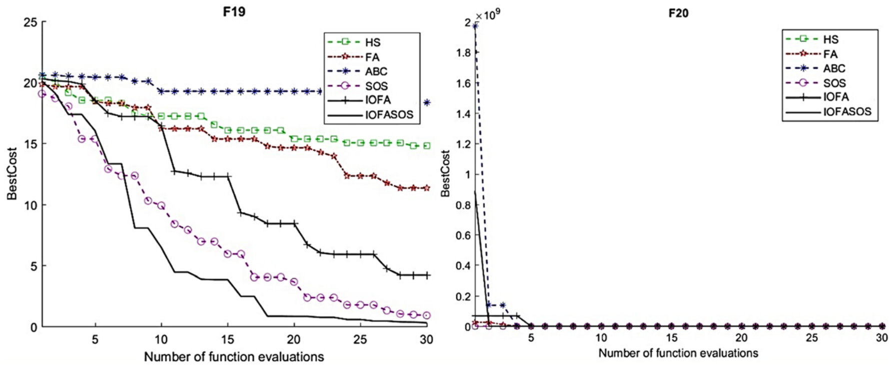

Fig. 12 The convergence rate of the two proposed approaches on 10D functions 19–20

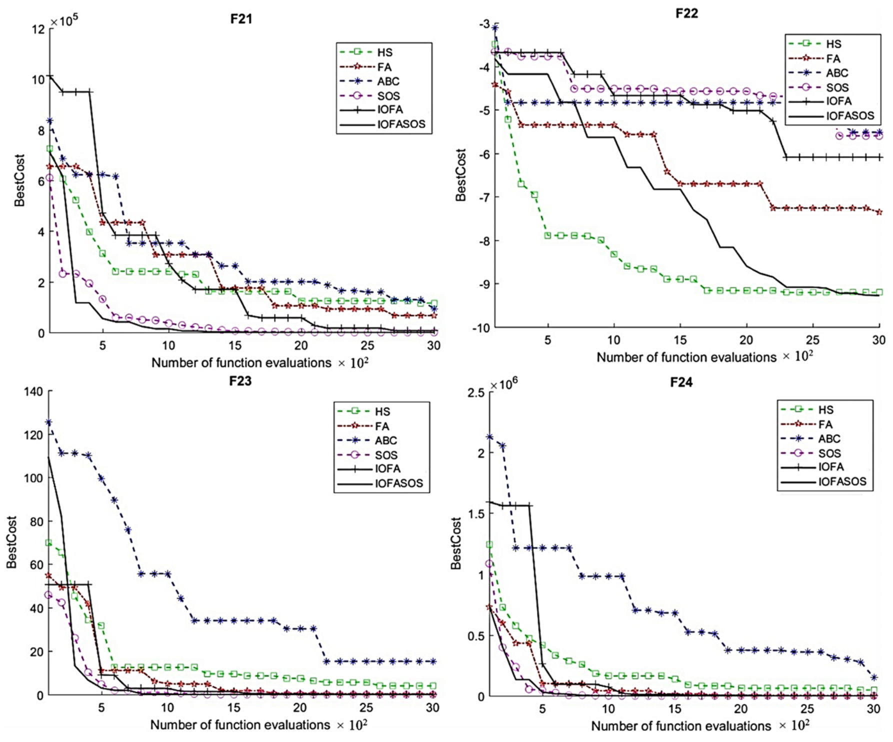

Fig. 13 The convergence rate of the two proposed approaches on 30D functions 21–24

The implementation of the two proposed approaches and other basic algorithms on the ten 10D benchmark functions is shown in Figs. 11 and 12, respectively. Based on the results presented in these figures, it can be concluded that both of the proposed approaches relatively outperformed other metaheuristic algorithms in terms of the convergence rate of the benchmark functions. The two proposed approaches succeeded in converging high-dimensional optimization functions well to the objective. Thus, the second proposed approach shows better results compared to the first one.

The proposed algorithms performed better on the F13 to F16 functions because these functions have a smaller range, and especially the IOFASOS algorithm was able to find the best points. Moreover, it brought the value of the functions to the nearest optimal point.

The results of implementing the two proposed approaches and other basic algorithms in the ten 30D benchmark functions are shown in Fig. 13. Based on these results, it can be deduced that by increasing the dimension to 30, some algorithms such as HS and ABC show weaker performance; however, the two proposed approaches still maintain their higher dimensions. However, the second proposed approach shows better results in comparison with other metaheuristic algorithms in higher dimensions.

This test further illustrates the effect of algorithms on the population as a whole. The results of this test are compared in Table 3. In this experiment, the two proposed approaches are compared with the SOS algorithm, ABC, FA, and HS algorithm in low, medium, and high dimensions.

Table 3 shows that the IOFASOS method has a high ability to find the optimal points. The SOS algorithm has the optimal answer for F5, F8, F19, and F21 functions. Moreover, the HS and ABC algorithms in the F8 function have optimal solutions. Therefore, the IOFASOS method has been able to find the best solutions by using SOS and OBL.

For a better evaluation in this experiment, the value of the objective function evaluation is considered to be 3000 so that all algorithms have the same evaluation number and can be compared together. Considering the results of the implementation of the two proposed approaches, SOS algorithm, ABC, HS, and FA in low, medium, and high dimensions, as shown in Table 3, it can be argued that the two proposed approaches are significant in terms of statistical criteria such as the worst, mean, and standard deviation. These criteria indicated that the two proposed approaches change all the population's solutions and converge them to the objective. Furthermore, the second proposed approach has better outcomes than the first approach and other algorithms regarding most benchmark functions, which indicates that the second approach based on the opposition method and generation of new and opposite fireflies has affected the entire population and improved the performance of the FA.

To further evaluate the two proposed approaches, they were compared with the algorithms Qi et al. presented in [51] in 2017 as two new types of butterfly algorithms, i.e., ABO1 and ABO2. Qi et al. implemented ABO1, ABO2, and basic metaheuristic algorithms such as ABC [49], GA [52], PSO [53], and cooperative PSO (CPSO) algorithms [54] on 22 benchmark functions and reported the results in Table 3 of the 22 benchmarks functions used in this reference which were the same ones reported in Table 1. Thus, the authors in [51] considered the initial population size for all algorithms to be 20 and the maximum objective function evaluation to be 100,000. Moreover, C1 and C2 learning rates were both set at 2.05 for both CPSO and PSO algorithms. Furthermore, the compression factor X was set to 0.729, and the division factor for CPSO was considered equal to the dimension. In addition, a one-point crossover operator used GA, where the crossover probability was set to

Table 1 Standard benchmark functions (Type: M: multimodal, N: non-separable, U: unimodal, S: separable) for evaluating the two proposed approaches

Number	Function	Type	Formulation	Range	D	Min
1	Beale	UN	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	[− 4.5, 4.5]	2	0
2	Bohachevsky3	MN	$f(x) = x_1^2 - 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$	[− 100, 100]	2	0
3	Shubert	MN	$f(x) = (\sum_{i=1}^{5} i\cos(i+1)x_1 + i)(\sum_{i=1}^{5} i\cos(i+1)x_2 + i)$	[− 10, 10]	2	− 186.73
4	Easom	UN	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	[− 100, 100]	2	− 1
5	Bohachevsky2	MN	$f(x) = x_1^2 - 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$	[− 100, 100]	2	0
6	Matyas	UN	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	[− 10, 10]	2	0
7	Six Hump Camel Back	MN	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 + 4x_2^2 + 4x_2^4$	[− 5, 5]	2	− 1.03163
8	Michalewicz2	MS	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(ix_i^2/\pi))^{20}$	[0, π]	2	− 1.8013
9	Bohachevsky1	MS	$f(x) = x_1^2 + 2\,x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	[− 100, 100]	2	0
10	Booth	MS	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	[− 10, 10]	2	0
11	Colville	UN	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1(x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)^2 + (x_4 - 1)^2$	[− 10, 10]	4	0
12	Michalewicz5	MS	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(ix_i^2/\pi))^{20}$	[0, π]	4	− 4.6877
13	Rastrigin	MS	$f(x) = \sum_{i=1}^{D} (x_i^2 - \cos(2\pi x_i) + 10)^2$	[− 5.12, 5.12]	4	0
14	Zakharov	UN	$f(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5ix_i)^2 + (\sum_{i=1}^{D} 0.5ix_i)^4$	[− 5, 10]	4	0
15	Step	US	$f(x) = (\sum_{i=1}^{D} x_i + 0.5)^2$	[− 5.12, 5.12]	10	0
16	Sphere	US	$f(x) = \sum_{i=1}^{D} x_i^2$	[− 100, 100]	10	0
17	Rosenbrock	MN	$f(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	[− 30, 30]	10	0
18	SumSquares	US	$f(x) = \sum_{i=1}^{D} ix_i^2$	[− 10, 10]	10	0
19	Ackley	MN	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{D-1}\cos(2\pi x_i)\right) + 20 + e$	[− 32, 32]	10	0
20	Schwefel 2.22	UN	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[− 10, 10]	10	0

Table 1 (continued)

Number	Function	Type	Formulation	Range	D	Min
21	Schwefel 1.2	UN	$f(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	[− 100, 100]	30	0
22	Michalewicz10	MS	$f(x) = -\sum_{i=1}^{D} \sin\left(x_i\right)(\sin(ix_i^2/\pi))^{20}$	[0, π]	30	− 9.6602
23	Quartic	US	$f(x) = \sum_{i=1}^{D} ix_i^4 + \text{Rand}$	[− 1.28, 1.28]	30	0
24	Dixon-Price	UN	$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{D} i(2x_i^2 - x_i - 1)^2$	[− 10, 10]	30	0

Table 2 Initialization of the parameters of the two proposed approaches and other algorithms

Algorithm	Values of parameters
HS [48]	bw = 0.2, HMCR = 0.5, PAR = 0.3, population size is 30
ABC [49]	limit = 5D, population size is 30, Non-looker = 30
FA [17]	α = .2,β0 = 1, γ = 1 and population size is 30
SOS [18]	population size is 30
IOFA	α = .2, β0 = 1, γ = 1 and population size is 30
IOFASOS	T = 1, U = 0.01, α = .2,β0 = 1, γ = 1 and population size is 30

be 0.95, and the probability mutation was set to be 0.1. In this test, the parameters of the algorithm of the two proposed approaches were set as the population = 20 and the maximum objective function evaluation = 100,000, so a proper comparison can be obtained. The results are reported in Table 4 where it was shown that the improvement of IOFA on F14 function compared to GA, CPSO, PSO, ABC, ABO2, and ABO1 was 100%. Furthermore, the results of the IOFA and FASOS methods in the F16 function were equal to zero, which indicates that the proposed algorithm has a high power of exploration and exploitation.

Table 4 compares the two proposed approaches with other robust algorithms, i.e., PSO, cooperative PSO, ABC, and GA implemented on the 30D benchmark optimization functions. Thus, to demonstrate the two proposed approaches' performance and evaluate them more precisely, the algorithms are evaluated in Table 4 on 20D functions. The results showed that the second approach had outperformed the other basic metaheuristic algorithms, modified particle, while the first approach in higher-dimension problems, namely in dimensions 20 and 30. Moreover, the second approach continued to improve the results with increasing dimensions of the problem significantly.

2.11 Conclusion and future works

The FA is a metaheuristic optimization algorithm inspired by the flashing behavior of fireflies in nature. Studies on the FA have shown the strengths and weaknesses of this algorithm; therefore, researchers have presented various hybrid, adaptive, modified, and improved versions of this algorithm in recent years to solve various optimization problems. In this paper, two approaches to improve the FA were proposed using the SOS algorithm's natural and opposition-based methods. In the first proposed approach, a new method of IOFA was proposed to accelerate the convergence rate and improve the FA's exploration capability. In the second approach (IOFA-SOS), the processes of the SOS algorithm were used to improve the exploration and exploitation of the first proposed approach; these two objectives were set in the second approach with two new parameters T and U. After introducing the two proposed approaches, 24 mathematical benchmark functions were introduced to evaluate the two proposed approaches. The two proposed approaches were compared with HS

Table 3 Evaluation of the two proposed approaches based on statistical criteria

Function	Criteria	HS	ABC	FA	SOS	IOFA	IOFASOS
F1	Best	0.00507	0.148033	0.000889	4.18E-06	2.83E-05	**0**
	Worst	0.00507	0.842532	2984.256	1.943409	0.000337	**1.7E-07**
	Mean	0.00507	0.714581	143.1926	0.059463	0.000152	**7E-08**
	SDT	0	0.191328	578.3979	0.298793	0.000103	**5E-08**
F2	Best	0.089186	0.001019	0.000208	3.33E-06	0.006893	**2.17E-06**
	Worst	0.089186	0.018143	6590.61	0.240957	0.111347	**2.05E-05**
	Mean	0.089186	0.008703	535.8661	0.037869	0.047515	**8.67E-06**
	SDT	0	0.005684	1494.911	0.076333	0.034578	**6.21E-06**
F3	Best	− 186.537	**− 186.731**	− 186.639	− 186.517	− 186.691	**− 186.731**
	Worst	− 186.537	− 186.675	35.02838	− 31.744	− 184.61	**− 186.731**
	Mean	− 186.537	− 186.709	− 70.9949	− 115.628	− 185.79	**− 186.731**
	SDT	0	0.017867	55.15291	43.46192	0.718473	**1.01E-05**
F4	Best	− 0.15118	− 1	− 0.99999	− 0.9825	− 0.99992	**− 1**
	Worst	− 8E-05	− 0.99977	0	0	− 0.99279	**− 1**
	Mean	− 0.00512	− 0.99989	− 0.67772	− 0.47532	− 0.99803	**− 1**
	SDT	0.027588	0.000072	0.459003	0.406982	0.002034	**6E-07**
F5	Best	0.000564	669.7472	1.11E-06	**0**	0.0002	1E-07
	Worst	0.056821	805.3719	11,305.19	0.001569	0.021491	**9.66E-06**
	Mean	0.045569	734.7568	1394.357	4.55E-05	0.005857	**4.84E-06**
	SDT	0.022887	57.70908	2962.69	0.000223	0.006834	**3.36E-06**
F6	Best	0.004105	9.7E-07	1.39E-05	2E-08	8.1E-07	**0**
	Worst	0.005685	1.32E-05	74.59726	0.001041	5.16E-06	**1E-08**
	Mean	0.005633	5.34E-06	6.842466	3.32E-05	2.24E-06	**0**
	SDT	0.000289	3.62E-06	17.59155	0.00015	1.63E-06	**0**
F7	Best	− 1.03147	**− 1.03163**	**− 1.03163**	− 1.03162	− 1.0316	**− 1.03163**
	Worst	− 1.03117	− 1.03156	2193.26	− 0.79473	− 1.03096	**− 1.03163**
	Mean	− 1.03144	− 1.0316	72.17185	− 1.0184	− 1.03138	**− 1.03163**
	SDT	9.09E-05	1.89E-05	400.6097	0.037055	0.000209	**2E-08**
F8	Best	**− 1.8013**	**− 1.8013**	**− 1.8013**	**− 1.8013**	− 1.80129	**− 1.8013**
	Worst	**− 1.8013**	− 1.80128	− 1.10998	− 1	− 1.80116	**− 1.8013**
	Mean	**− 1.8013**	**− 1.8013**	− 1.76072	− 1.75814	− 1.80125	**− 1.8013**
	SDT	0	5.79E-06	0.13078	0.159834	3.85E-05	**1E-08**
F9	Best	0.401504	43.59588	1.16E-05	1.35E-05	0.003885	**4.73E-06**
	Worst	0.401504	80.82938	9096.136	0.489445	0.092874	**0.000129**
	Mean	0.401504	63.70456	1210.487	0.039766	0.037871	**5.16E-05**
	SDT	0	9.064231	2424.931	0.126574	0.026984	**4.33E-05**
F10	Best	0.075508	0.32598	0.507413	0.001521	**2.56E-05**	0.507236
	Worst	0.103728	5.016178	16,342.68	24.07529	**0.001023**	0.507237
	Mean	0.095262	2.987656	553.545	1.565191	**0.000417**	0.507236
	SDT	0.013153	1.179136	2982.258	3.595199	0.000355	**3.8E-07**

Table 3 (continued)

Function	Criteria	HS	ABC	FA	SOS	IOFA	IOFASOS
F11	Best	0.570799	0.265546	0.279281	0.55475	0.434908	**0.000243**
	Worst	0.570799	1.4114	757,063.5	93.32534	1.038928	**0.001617**
	Mean	0.570799	1.101689	111,372.5	11.69078	0.681984	**0.000962**
	SDT	0	0.253782	218,250	15.01096	0.287364	**0.000479**
F12	Best	− 4.63935	**− 4.65849**	− 4.6015	− 3.88753	− 4.48238	− 4.36921
	Worst	− 4.63598	**− 4.5211**	− 3.45382	− 2.30715	− 4.44766	− 4.36088
	Mean	− 4.63675	**− 4.60095**	− 4.2049	− 3.07792	− 4.47034	− 4.36452
	SDT	0.00082	0.043084	0.339208	0.376912	0.010729	**0.002447**
F13	Best	149.0286	174.4285	290.5744	205.7738	138.4821	**20.80887**
	Worst	182.1557	221.7692	480.7881	293.6647	181.5516	**22.30346**
	Mean	172.0386	202.1199	383.8295	255.0945	165.7612	**21.6926**
	SDT	8.633782	14.92533	33.91997	20.65836	15.55422	**0.467921**
F14	Best	217.1136	49.68085	239.9202	37.16161	15.48633	**3.525626**
	Worst	415.7911	91.37739	120,022.5	329.6845	29.61443	**4.117182**
	Mean	345.4038	72.65809	4379.748	125.5099	23.25916	**3.865484**
	SDT	47.20682	10.55848	21,841.76	70.51708	3.875904	**0.175269**
F15	Best	8983.545	12,419.15	10,010.1	5.665056	152.0947	**5.553813**
	Worst	13,747.88	16,420.38	38,627.29	24.73191	335.0202	**6.251262**
	Mean	11,948.09	14,030.96	19,120.93	11.37422	195.1326	**5.970805**
	SDT	1279.322	949.8395	5904.941	4.049603	77.45958	**0.203503**
F16	Best	19.24362	109.4713	29.15242	0.006411	0.15841	**0.004287**
	Worst	30.32951	119.9052	87.55063	0.038708	0.212024	**0.004752**
	Mean	26.48818	114.7659	52.55396	0.018141	0.150876	**0.004482**
	SDT	3.009924	2.460754	15.01536	0.00749	0.049035	**0.000139**
F17	Best	7,721,901	211,289	37,594,918	55.25486	5159.715	**31.96762**
	Worst	18,263,543	639,747.3	2.79E+08	221.2664	10,137.69	**32.6698**
	Mean	15,011,081	428,935.6	1.1E+08	98.78383	5073.296	**32.28204**
	SDT	2,986,951	103,522.1	56,663,347	30.22111	3002.787	**0.242727**
F18	Best	1006.421	4830.389	1310.953	0.844671	8.255387	**0.404581**
	Worst	1640.17	5530.142	4878.09	4.053942	14.21591	**0.795941**
	Mean	1435.47	5177.963	2379.672	1.935624	9.849041	**0.594415**
	SDT	151.128	177.972	877.7271	0.787253	2.965925	**0.110761**
F19	Best	12.12033	12.85397	13.93749	**0.941729**	2.750572	1.883125
	Worst	13.84492	14.94433	18.92977	2.424198	3.079144	**2.133204**
	Mean	13.05092	14.16576	15.92589	**1.37308**	2.869807	2.011667
	SDT	0.434662	0.443112	1.270447	0.323626	0.189307	**0.072159**
F20	Best	24.75832	10.46132	76.8287	0.893273	3.967263	**0.451827**
	Worst	37.54409	13.76729	1.96E+11	1.90544	6.913953	**0.579039**
	Mean	33.79012	12.22458	6.75E+09	1.27354	5.340428	**0.531694**
	SDT	3.418202	0.810814	3.58E+10	0.251587	1.0833	**0.037934**

Table 3 (continued)

Function	Criteria	HS	ABC	FA	SOS	IOFA	IOFASOS
F21	Best	108,715.9	264,591.9	130,853.3	**50.07041**	1200.804	50.37555
	Worst	162,254.6	350,679.8	643,373.9	229.2392	2213.981	**56.02894**
	Mean	141,547.6	298,980.5	250,664.6	92.89756	1368.37	**53.55342**
	SDT	15,278.81	18,099.61	94,454.22	43.64614	576.7521	**1.48253**
F22	Best	− 9.0546	**− 8.29412**	− 4.74712	− 5.59955	− 7.47441	− 6.9183
	Worst	− 8.80381	− 6.30053	− 3.18278	− 3.34703	− 6.70922	**− 6.88772**
	Mean	− 8.87048	− 6.83199	− 3.78143	− 4.45626	**− 7.0215**	− 6.89786
	SDT	0.059694	0.421232	0.429872	0.547037	0.291193	**0.008695**
F23	Best	3.550624	0.12324	8.169501	0.030369	0.029195	**0.00653**
	Worst	9.169915	0.275828	169.8931	0.279566	0.098741	**0.022363**
	Mean	7.417959	0.210969	47.22267	0.126182	0.0735	**0.015706**
	SDT	1.606593	0.04632	38.52895	0.062332	0.018824	**0.004832**
F24	Best	68,424.4	830.5563	156,371	2.522114	9.730604	**0.91413**
	Worst	136,895.8	1480.822	1,734,747	7.310317	14.60718	**0.944961**
	Mean	110,354.1	1149.422	673,565.5	4.177716	10.59777	**0.931911**
	SDT	19,745.25	163.3065	401,352.2	1.194761	3.173252	**0.007579**

and ABC, FA, and SOS algorithms in all experiments. Experiments were carried out on functions with 2 to 30 dimensions, in two parts, i.e., in terms of convergence rate and statistical criteria. The results of experiments in different dimensions showed that the first proposed approach performs better in low and medium dimensions and shows a relatively moderate performance in higher dimensions, while the second proposed approach was able to show a better performance better. Moreover, the second approach provided better results in terms of statistical criteria such as the best and the worst value of the objective function, the mean of the total population's objective function, and other statistical parameters compared with other algorithms and the first proposed approach. Future work will evaluate the proposed algorithm using more complex problems, and hybrid algorithms will improve the operators.

Table 4 Evaluation of the proposed algorithm and its comparison with other algorithms reported in 30D

IOFA	FASOS	ABO1 [51]	ABO2 [51]	ABC [49]	PSO [53]	CPSO [54]	GA [52]	Criteria	Function	
0.000000e+000	**0.000000e+000**	0.000000e+000	4.547474e− 014	4.069932e-010	1.243182e-009	1.442855e-002	2.914067e+002	Mean	30D	F13
0.000000e+000	**0.000000e+000**	0.000000e+000	1.781358e-013	1.243182e-009	1.920520e+001	8.228709e-003	5.459585e+001	Std		
0.000000e+000	**0.000000e+000**	0.000000e+000	0.000000e+000	1.136870e-013	2.832990e+001	3.709010e-003	1.944010e+002	Best		
0.000000e+000	**0.000000e+000**	0.000000e+000	9.663380e-013	5.659500e-009	1.167770e+002	3.569160e-002	4.253200e+002	Worst		
2.7382129954e-7	2.1223412583e-5	2.256266e+001	1.987342e+000	2.194447e+002	1.338578e+002	2.106311e+002	4.302965e+002	Mean	30D	F14
2.0916194329e-6	1.4227595869e-4	8.799411e+000	1.775732e+000	3.597426e+001	1.219247e+002	8.063483e+001	9.529330e+001	Std		
8.1894569802e-7	7.0847347803e-5	6.474196e+000	5.430510e-001	1.639170e+002	7.990490e-002	6.958440e+001	2.648770e+002	Best		
5.1467636210e-7	3.6385004812e-5	4.179764e+001	8.002880e+000	2.865630e+002	5.000310e+002	4.040080e+002	6.260390e+002	Worst		
0.000000e+000	**0.000000e+000**	6.698830e-017	7.155941e-017	8.052865e-016	2.219698e-004	9.718608e-006	2.168373e+000	Mean	30D	F16
0.000000e+000	**0.000000e+000**	3.101193e-017	3.884028e-017	1.277349e-016	1.391320e-004	5.504014e-006	8.592691e-001	Std		
0.000000e+000	**0.000000e+000**	3.740350e-017	4.545520e-017	6.033220e-016	4.728400e-005	2.984920e-006	9.105550e-001	Best		
0.000000e+000	**0.000000e+000**	2.198343e-016	2.141320e-016	1.204110e-015	6.932630e-004	2.586070e-005	4.622420e+000	Worst		
3.0630335903e-73	5.6997198197e-46	7.617745e-017	9.584096e-017	7.947345e-016	4.251270e-002	5.624401e-004	7.265310e+001	Mean	30D	F18
1.2651354133e-72	1.2679655903e-45	4.043972e-017	6.050023e-017	1.230775e-016	5.274829e-002	2.982597e-004	3.432228e+001	Std		
5.2880811339e-73	7.8704295787e-46	4.028703e-017	3.623430e-017	4.038980e-016	7.852210e-003	2.260130e-004	3.165710e+001	Best		
3.0630335803e-73	5.6997191786e-46	2.137263e-016	2.431550e-016	9.856360e-016	2.805710e-001	1.481230e-003	1.842400e+002	Worst		

Table 4 (continued)

IOFA	FASOS	ABO1 [51]	ABO2 [51]	ABC [49]	PSO [53]	CPSO [54]	GA [52]	Criteria	Function	
4.4408920985e-15	**4.4408920985e-15**	6.335673e-015	6.809368e-015	6.412648e-014	2.030707e+000	1.547524e-002	2.015904e+001	Mean	30D	F19
4.4408920985e-15	4.4408920985e-15	**1.802705e-015**	1.703396e-015	9.207796e-015	5.647839e-001	3.958397e-003	2.491498e-001	Std		
4.4408920985e-15	**4.4408920985e-15**	**4.440892e-015**	**4.440890e-015**	4.352070e-014	9.468440e-001	9.894610e-003	1.955740e+001	Best		
4.4408920985e-15	**4.4408920985e-15**	7.993606e-015	7.993610e-015	8.260060e-014	3.099410e+000	3.233120e-002	2.065720e+001	Worst		

References

1. Gharehchopogh FS, Maleki I, Dizaji ZA (2021) Chaotic vortex search algorithm: metaheuristic algorithm for feature selection. Evol Intel 1:1–32
2. Gharehchopogh FS, Abdollahzadeh B (2021) An efficient harris hawk optimization algorithm for solving the travelling salesman problem. Cluster Comput. https://doi.org/10.1007/s10586-021-03304-5
3. Gharehchopogh FS, Gholizadeh H (2019) A comprehensive survey: whale optimization algorithm and its applications. Swarm Evol Comput 48:1–24
4. Shayanfar H, Gharehchopogh FS (2018) Farmland fertility: a new metaheuristic algorithm for solving continuous optimization problems. Appl Soft Comput 71(1):728–746
5. Gharehchopogh FS, Shayanfar H, Gholizadeh H (2020) A comprehensive survey on symbiotic organisms search algorithms. Artif Intell Rev 53(3):2265–2312
6. Zamani H, Nadimi-Shahraki MH, Gandomi AH (2019) CCSA: conscious neighborhood-based crow search algorithm for solving global optimization problems. Appl Soft Comput 85:105583
7. Abdollahzadeh B, Gharehchopogh FS, Mirjalili S (2021) African vultures optimization algorithm: a new nature-inspired metaheuristic algorithm for global optimization problems. Comput Indust Eng 1:107408
8. Cheng Z et al (2021) Hybrid firefly algorithm with grouping attraction for constrained optimization problem. Knowledge-Based Syst 220:106937. https://doi.org/10.1016/j.knosys.2021.106937
9. Nand R, Sharma BN, Chaudhary K (2021) Stepping ahead firefly algorithm and hybridization with evolution strategy for global optimization problems. Appl Soft Comput 109:107517
10. Nadimi-Shahraki MH, Taghian S, Mirjalili S (2021) An improved grey wolf optimizer for solving engineering problems. Expert Syst with Appl 166:113917
11. Zaman HRR, Gharehchopogh FS (2021) An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems. Eng Comput. https://doi.org/10.1007/s00366-021-01431-6
12. Ghafori S, Gharehchopogh FS (2021) Advances in spotted hyena optimizer: a comprehensive survey. Archiv Comput Methods Eng. https://doi.org/10.1007/s11831-021-09624-4
13. Zamani H, Nadimi-Shahraki MH, Gandomi AH (2021) QANA: Quantum-based avian navigation optimizer algorithm. Eng Appl Artif Intell 104:104314
14. Abdollahzadeh B, Soleimanian Gharehchopogh F, Mirjalili S (2021) Artificial gorilla troops optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems. Int J Intell Syst. https://doi.org/10.1002/int.22535
15. Abedi M, Gharehchopogh FS (2020) An improved opposition based learning firefly algorithm with dragonfly algorithm for solving continuous optimization problems. Intell Data Analy 24(2):309–338
16. Banaie-Dezfouli M, Nadimi-Shahraki MH, Beheshti Z (2021) R-GWO: Representative-based grey wolf optimizer for solving engineering problems. Appl Soft Comput 106:107328
17. Yang X-S (2010) Firefly algorithm, Levy flights and global optimization. Research and development in intelligent systems XXVI. Springer, pp 209–218
18. Cheng M-Y, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112
19. Zhang L et al (2016) A novel hybrid firefly algorithm for global optimization. PLoS ONE 11(9):0163230
20. Sarbazfard S, Jafarian A (2017) A hybrid algorithm based on firefly algorithm and differential evolution for global optimization. J Adv Comput Res 8(2):21–38
21. Lieu QX, Do DT, Lee J (2018) An adaptive hybrid evolutionary firefly algorithm for shape and size optimization of truss structures with frequency constraints. Comput Struct 195:99–112
22. Farahani SM et al (2012) Some hybrid models to improve firefly algorithm performance. Int J Artificial Intell 8(12):97–117
23. Farook S, Raju PS (2013) Evolutionary hybrid genetic-firefly algorithm for global optimization. IJCEM Int J Comput Eng Manag 16(3):37–45
24. Rahmani A, MirHassani S (2014) A hybrid firefly-genetic algorithm for the capacitated facility location problem. Inf Sci 283:70–78
25. Baykasoğlu A, Ozsoydan FB (2015) Adaptive firefly algorithm with chaos for mechanical design optimization problems. Appl Soft Comput 36(1):152–164

26. Xia X et al (2018) A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm. J comput sci 26(1):488–500
27. Gupta S, Arora S (2016) A hybrid firefly algorithm and social spider algorithm for multimodal function. Intelligent Systems Technologies and Applications. Springer, pp 17–30
28. Hassanzadeh T. and Meybodi MR (2012) A new hybrid algorithm based on Firefly Algorithm and cellular learning automata. in 20th Iranian Conference on Electrical Engineering (ICEE2012).
29. Alsmadi MK (2014) A hybrid firefly algorithm with fuzzy-C mean algorithm for MRI brain segmentation. Am J Appl Sci 11(9):1676–1691
30. Mohammed H, Rashid T (2020) A novel hybrid GWO with WOA for global numerical optimization and solving pressure vessel design. Neural Comput Appl 32(18):14701–14718
31. Torabi S, Safi-Esfahani F (2019) A hybrid algorithm based on chicken swarm and improved raven roosting optimization. Soft Comput 23(20):10129–10171
32. Maleki I, Ebrahimi L, Gharehchopogh FS (2014) A hybrid approach of firefly and genetic algorithms in software cost estimation. MAGNT Res Report 2(6):372–388
33. Mohmmadzadeh H, Gharehchopogh FS (2021) An efficient binary chaotic symbiotic organisms search algorithm approaches for feature selection problems. The J Supercomput 1:1–43
34. Mohammadzadeh H, Gharehchopogh FS (2021) Feature selection with binary symbiotic organisms search algorithm for email spam detection. Int J Inf Technol Decis Mak 20(01):469–515
35. Nama S, Saha AK, Sharma S (2021) Performance up-gradation of symbiotic organisms search by backtracking search algorithm. J Ambient Intell Humanized Comput. https://doi.org/10.1007/s12652-021-03183-z
36. Sharma S et al (2021) MPBOA-A novel hybrid butterfly optimization algorithm with symbiosis organisms search for global optimization and image segmentation. Multimedia Tools and Appl 80(8):12035–12076
37. Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06), IEEE, Vol. 1, pp. 695–701
38. Tubishat M et al (2020) Improved salp swarm algorithm based on opposition based learning and novel local search algorithm for feature selection. Expert Syst with Appl 145(1):113122
39. Dhargupta S et al (2020) Selective opposition based grey wolf optimization. Expert Syst Appl 151(1):113389
40. Jain P, Jain P, Saxena A (2020) Opposition theory enabled intelligent whale optimization algorithm. Intelligent Computing Techniques for Smart Energy Systems. Springer, pp 485–493
41. Yu S et al (2015) Enhancing firefly algorithm using generalized opposition-based learning. Computing 97(7):741–754
42. Zhou Y, Wang R, Luo Q (2016) Elite opposition-based flower pollination algorithm. Neurocomputing 188:294–310
43. Sharma TK, Pant M (2017) Opposition based learning ingrained shuffled frog-leaping algorithm. J Comput Sci 21(1):307–315
44. Sarkhel R et al (2018) An improved harmony search algorithm embedded with a novel piecewise opposition based learning algorithm. Eng Appl Artif Intell 67(1):317–330
45. Bulbul SMA et al (2016) Opposition-based krill herd algorithm applied to economic load dispatch problem. Ain Shams Eng J 9(3):423–440
46. Elaziz MA, Oliva D, Xiong S (2017) An improved opposition-based sine cosine algorithm for global optimization. Expert Syst Appl 90:484–500
47. Xu Q et al (2014) A review of opposition-based learning from 2005 to 2012. Eng Appl Artif Intell 29(1):1–12
48. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. SIMULATION 76(2):60–68
49. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.
50. Jamil M, Yang X-S (2013) A literature survey of benchmark functions for global optimisation problems. Int J Math Modell Num Optimisat 4(2):150–194
51. Qi X, Zhu Y, Zhang H (2017) A new meta-heuristic butterfly-inspired algorithm. J Comput Sci 23(1):226–239

52. Sumathi S, Hamsapriya T, Surekha P (2008) Evolutionary intelligence: an introduction to theory and applications with Matlab. Springer, Newyork
53. Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multi-dimensional complex space. IEEE Trans Evol Comput 6(1):58–73
54. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239