EASR

Engineering and Applied Science Research

https://www.tci-thaijo.org/index.php/easr/index

Published by the Faculty of Engineering, Khon Kaen University, Thailand

RO-FOA: An ecosystem-inspired compact fruit fly optimization algorithm for Box-constrained optimization

Wirote Apinantanakon*[1)], Khamron Sunat[1)] and Joel Alan Kinmond[2)]

[1)]Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen 40002, Thailand
[2)]27 Patrick Street, Trenton, Ontario k8v 4B, Canada

Received 27 May 2019
Revised 6 August 2019
Accepted 9 August 2019

Abstract

The fruit fly optimization algorithm (FOA) was a recently proposed. FOA has a number of advantages over other nature-inspired algorithms such as its simple structure and ease of implementation. However, the FOA's search procedures present a problem. FOA has a low success rate search and a slow convergence when it has to deal with complex problems. This is because FOA generates a new position around its swarm location using a random uniform distribution. To eliminate this drawback, our paper presents an improved fruit fly algorithm called RO-FOA. The RO-FOA technique takes knowledge of a mutualistic relationship common in ecosystems and biological theory. Our strategy blends two popular algorithms, i.e., the random walk (RW) and the opposition-based learning (OBL) algorithms, to establish a two-characteristic swarm for searching procedures. RO-FOA's structure is very compact as the implementation uses only three fruit flies. Furthermore, the advantages of including a two-characteristic population and dynamic distribution adaptation in the evolving process can produce an algorithm with the necessary search efficiency to find an optimal solution. A comprehensive set of 34 benchmark functions, containing a wide range of dimensions were used to validate the capability of the proposed algorithm. The results show that RO-FOA outperformed the existing FOA, as well as seven comparatively well-known meta-heuristic algorithms. RO-FOA can efficiently train multi-layer perceptrons for 5-bit and 8-bit auto-encoder problems. These results demonstrate that the RO-FOA can enhance the diversity of population distributions, solution quality and the convergence rate of the algorithm.

Keywords: Optimization algorithm, Nature-inspired algorithm, Fruit fly optimization algorithm, Meta-heuristics, Ecosystem, Mutualistic relationship

1. Introduction

In order to solve complex optimization problems, researchers have started adapting knowledge from natural phenomena as tools for the development of several new algorithms. Concepts for the creation of new computational intelligence methods can be derived from natural mechanisms and principles. The main concepts of the aptly named 'nature-inspired algorithms' have been observed within successful biological systems. Accordingly, most nature-inspired algorithms are biologically inspired, or bio-inspired, and mimic specific behavior in nature. Examples of such popular nature-inspired algorithms include the particle swarm optimization algorithm (PSO) [1], which was inspired by the social behavior of flocking birds, or schooling fish, the ant colony optimization algorithm (ACO) [2], which mimics an ant colony's behavior in their search for food, the artificial bee colony algorithm (ABC) [3], motivated by the intelligent behavior of a honey bee swarm, the cuckoo search algorithm (CS) [4], inspired by the parasitic bio-interactions of a cuckoo species that lays their eggs in the nests of other host birds, and the bat-inspired algorithm (BA) [5], which was inspired by the echolocation behavior of bats, to name but a few. These widely used algorithms have proven highly efficient in solving problems in many scientific fields, such as engineering [6-7], task scheduling [8], mechanical design problems [9], data mining applications [10] and image processing [11]. Each nature-inspired algorithm has different capabilities when it comes to finding solutions, which depend on the individual abilities of living things in nature. Developing a successful and modern nature-inspired algorithm is a challenging task, even today, as there is no one particular nature-inspired algorithm capable of solving every scientific problem. Hence, continual development of new algorithms is required.

One of these new algorithms is the FOA, which was proposed by Pan [12]. The FOA, a swarm intelligence method-based stochastic optimization technique, mimics the foraging behavior of fruit flies. The FOA is user-friendly and because of its simplicity and shortness. FOA can be easily understood by most researchers in this field. This also means that it can more easily be implemented into program code, compared with other well-known algorithms such as the differential evolution (DE), genetic algorithms (GA) and

*Corresponding author. Tel.: +6681 977 2516
Email address: wirotta@gmail.com

particle swarm optimization (PSO). The FOA possesses the same processing abilities for finding solution optimizations as the other algorithms, yet it involves fewer parameter settings and its processes are much shorter than other swarm algorithms [12]. The FOA has achieved success in several applications including research into optimization problems [13-16], neural network parameter optimization [17-18], swarm techniques for mini-autonomous surface vehicles (ASVs) [19], identification of dynamic protein complexes [20], support vector regression for seasonal electricity consumption forecasting [21], a short-term power load forecasting model based on the generalized regression neural network with a decreasing step fruit fly optimization algorithm [22], and efficient truss optimization using the contrast-based fruit fly optimization algorithm [23]. However, updating the position of a swarm of fruit flies through iterative generations remains the most challenging obstacle in FOA search. To seek an optimal solution, the FOA determines the scope of the searching radius through a random uniform distribution to update a new position. The radius values can be in the range of [0, 1] and the range is fixed during iterations. The drawback of this approach is that the FOA has to deal with variations in optimization problems. For example, if the algorithm needs to find the optimal solution in a search space between -1000 and 1000, the FOA generates the location of a swarm of fruit flies (in the range of [0, 1]), which is always far from an optimal solution in the early iterations. The search radius is too small compared with the distance from the current swarm locations to a promising region. In contrast, in the final iterations, a very small radius is needed to adjust a search vector to approach an optimal solution, but the search radius is very large. This disadvantage, referred to as the *disadvantage ability of exploration and exploitation*, has encouraged many research efforts into variant techniques to improve the FOA [14-16].

Several researchers continue to propose various improved FOAs, such as the improved fruit fly optimization algorithm for solving optimization problems (LGMS-FOA) [24], the improved fruit fly optimization algorithm for continuous function optimization problems (IFFO) [15], and the novel multi-swarm fruit fly optimization algorithm (MFOA) [16]. These FOA-based algorithms focus on the 'radius' through gradual and continuous updating (through special parameters) in the search process. However, given the above-mentioned disadvantage of the improved FOAs, as search time increases within each algorithm, the radius value of the population often converges slowly until it is unable to change at all. When the algorithms are dealing with complex optimization problems, they become trapped at local optima, and are incapable of finding a final optimum solution.

In this paper, we imitated the natural phenomena of the relationships of organisms within an ecosystem, to improve the algorithm's searching efficiency. The proposed fruit fly optimization algorithm is a hybrid of a random walk and opposition-based learning algorithms. It is called fruit fly optimization algorithm (RO-FOA), mimics the relationship between organisms and their environment, which affects the survival of all organisms. The random walk is a fundamental random number generator used in several optimization algorithms [25-28]. The first attempt to implement the OBL concept in optimization was proposed by Rahnamayan et al. [29-30]. As the case study, OBL is used in the DE algorithm to improve its performance and it is called the opposition DE (ODE). In [30], a comprehensive trial was conducted to confirm ODE performance using a standard set of functions consisting of 58 global optimization problems, varying the impacts of dimension, spot differences, population size, different strategies of mutation, and the jump rate. The experimental results were checked, analyzed successfully and confirmed that ODE achieved better results than DE. Other than that, there are several recent literature reports that proposed opposition-based learning algorithms [31-34]. We put forth two new ideas regarding the OBL that are a variation of shrinking probability versus the number of iterations and the control of search transition from the exploration to the exploitation phase through this probability. The premise of the proposed method is based on the relationship of organisms within the same species, or, as named in various ecological studies, the *Intraspecific Relationship* [35]. The RO-FOA achieves better dispersion swarm locations than using the strategy of a random walk algorithm or opposition-based learning alone. Additionally, the main advantage of the RO-FOA algorithm is that it generates a more diversified population distribution than the original FOA and the existing 'improved' fruit fly optimization algorithms we tested. Furthermore, we found that the space complexity is very low as the implementation requires only three vectors.

We have organized the remainder of this paper as follows. The FOA is presented in Section 2. The concept of the relationship of an organism within an ecosystem, and the RO-FOA strategy are described in Section 3. Then, the proposed random walk and opposition-based learning - fruit fly optimization algorithm (RO-FOA) are explained in Section 4. Section 5 explains the evaluation of algorithms and settings. The results and discussion are summarized in Section 6. Section 7 shows the ability of RO-FOA in multi-layer perceptron training. Last, Section 8 concludes our proposed approach with suggestions for future research.

2. The fruit fly optimization algorithm (FOA)

The drosophila optimized algorithm or fruit fly optimization algorithm (FOA), developed in 2012 by Pan (also can called FFO [36]), determines global optimization based on the foraging behavior of fruit flies. Compared to other species, the fruit fly possesses a keener sense of smell and sight in search of their food. The drosophila olfactory organ can detect a food source as far as 40 kilometers away, which triggers a flight reaction toward the target location. Figure 1 illustrates the pattern of the fruit flies' search for food.

As with all swarm optimization algorithms, the first phase of the fruit flies' quest for food involves a random search, with no specific position or direction. In the second phase, the fruit fly with the best sense of smell, also referred to as *best fitness*, within the group is determined. In the last phase, the fruit fly with the *best fitness* updates a new position for each individual fruit fly. The steps of FOA are summarized in Algorithm 1.

2.1 Disadvantages of the FOA

The problems or disadvantages, which limit the capability of the FOA process in solving the various optimizations, are briefly described as follows:

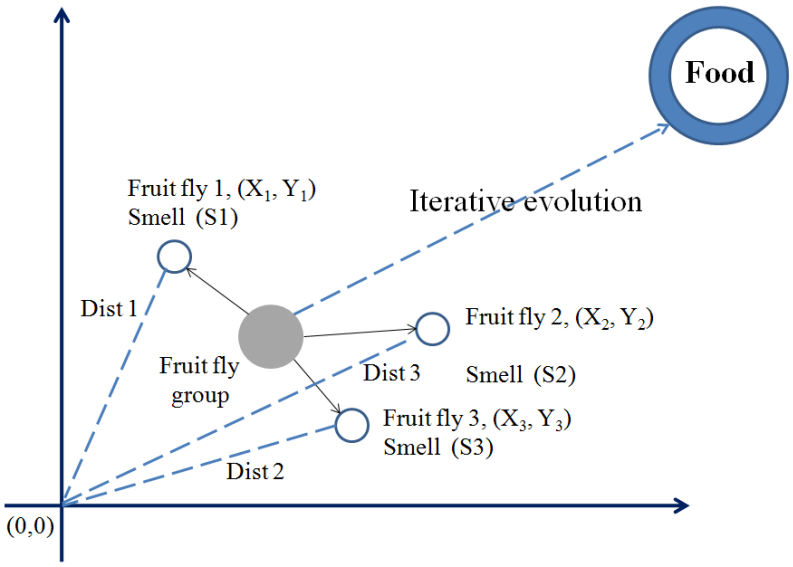


Figure 1 The directional process of fruit fly's search for food (Pan, 2012, p.70).

Algorithm 1: The FOA algorithm

Step 1. Initialization of FOA parameters, consisting of a random location (X_axis, Y_axis), population size (*sizepop*), and the maximum iteration (*maxgen*).

Step 2. Give the random position and fly direction of an individual fruit fly in their search for food.

$$X_i = X\_axis + Random()$$
$$Y_i = Y\_axis + Random()$$

Step 3. Calculate the distance (*Dist*) to the food's origin, as the exact position of the food's location is not known at this stage.

$$Dist_i = \sqrt{x_i^2 + y_i^2}.$$

Step 4. The smell concentration judgment value (S_i) is calculated.

$$S_i = \frac{1}{Dist_i}.$$

Step 5. The smell concentration judgment of the individual fruit fly, obtained from Step 4, is calculated by substituting S_i into the smell concentration judgment function (also called *fitness function*), to find the optimal smell.

$Smell_i$ = objective function (S_i).

Step 6. Determine the fruit fly with the optimal smell concentration judgment among the fruit fly group.

[*bestSmell*, *bestIndex*] = optimal (*Smell*).

Step 7. Keep the best (*x, y*) position and the minimal concentration value; and use this position as the base point of flying towards the next location (in step 2).

Smellbest = *bestSmell*,
X_axis = *X* (*bestIndex*),
Y_axis = *Y* (*bestIndex*).

Step 8. Enter into iterative optimization by repeating steps 2-7, and determine whether the smell concentration is better than the previous iterative smell concentration. If yes, go to step 7. The process will stop when the smell concentration no longer changes, or when the iterative number reaches the maximum iteration number.

(1) FOA cannot solve high-dimensional function optimization problems when the set of decision variables (X_i and Y_i) do not exist within Step 2.

(2) The smell value (S_i), in according to Step 4, cannot appropriately evaluate the "objective function (S_i)" when there are negative numbers in the domain because $S_i = 1/Dist_i > 0$ so that the function cannot determine S_i as negative.

(3) The fixed search radius, with random uniform distribution, *Random* ( ), within the processes, limited the convergence of FOA in the processes of exploration and exploitation. This point is the important disadvantage of FOA which has inspired several improved versions of FOA.

2.2 The improved version of FOA for solving high-dimensional function optimization

Several improved FOAs has been proposed to overcome the drawbacks of FOA [16, 36-37]. These improved FOAs try to improve the search efficiency by proposing a dynamic search radius. A brief summary of the improved FOAs are as follows:

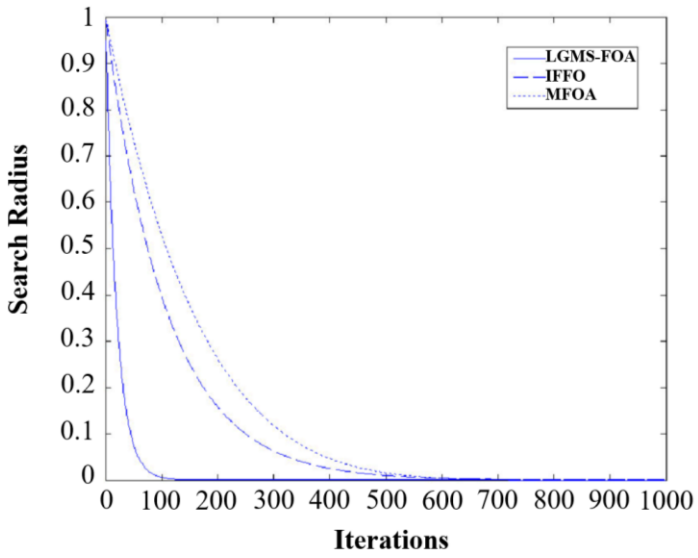


Figure 2 Variation of search radius versus iteration of LGMS-FOA, IFFO, and MFOA.

(1) The LGMS-FOA [24] presented some parameters to tune the radius by adding the parameter, *w*, when changing the radius. The LGMS-FOA method is: $x = X\_axis + w \times rand$ (domain of definition), $w = w_o \times \alpha^t$, where $\alpha = 0.95$, t = current iteration, $x \in X, x = (x_1, x_2, \ldots, x_n)$, $x \in \mathbb{R}^n$ and *X_axis* is the best position obtained during iterations.

(2) The IFFO [36] introduced a new control parameter to adaptively adjust the search radius. The IFFO presented the changing search radius dynamically during iterations such that $\lambda = \lambda_{max} \times \exp(\log(\lambda_{min} / \lambda_{max}) \times t / t_{max})$, where λ is the radius variation in each iteration, λ_{max} = (UB-LB) / 2, UB is the upper bound and LB is the lower bound, $\lambda_{min} = 10^{-5}$, t is the current iteration and t_{max} is the maximum iteration number.

(3) The MFOA [38] presented a multi-swarm fruit fly that employed sub-swarms in the search space with the behavior of simultaneously exploring the optimal solution. Moreover, MFOA provided a shrunken exploration radius as in equation $R(t) = (\text{UB - LB} / 2) \times (G_{max} - G / G_{max})^{\theta}$, where t is the current iteration, UB is the upper bound, LB is the lower bound, G is the number of sub-swarms, G_{max} is the maximum number of sub-swarms and $\theta = 2 \sim 6$.

(4) The MSFOA [39] presented a strategy to theoretically analyze the convergence of the FOA and showed that its convergence depends on the initial positions of the swarms. MSFOA used a Gaussian mutation operator rather than the uniform random number (as can be seen in the details of [39]). For a flying fruit fly, MSFOA used a linear generation mechanism, through the equation of $x_{i,j}^t = X_j^t + \omega \times rand(R_{min}, R_{max})$, where $\omega = \omega_0 \times \alpha^t$, $\omega_0 = 1, \alpha = 0.95$, t is current iteration, ω is the search coefficient, α is the initial weight and R_{min}, R_{max} are obtained from the domain boundary of the benchmarks function.

The graph in Figure 2 is generated from three improved FOAs approaches: LGMS-FOA, IFFO and MFOA. Parameters were set at the max-iteration number = 1000, max-radius = 1, min-radius = 0, X-axis is the number of iterations, and Y-axis is the search radius values. They contain an additional parameter dependent upon the algorithms. The search radius graph of MSFOA is similar to that of LGMS-FOA. The graph demonstrates that the radius will gradually decrease, depending on the iteration number.

2.3 Disadvantages of the variant FOAs

There are, however, some disadvantages in the searching procedures of variant FOAs such as LGMS-FOA, IFFO, MFOA and MSFOA. They are summarized as follows:

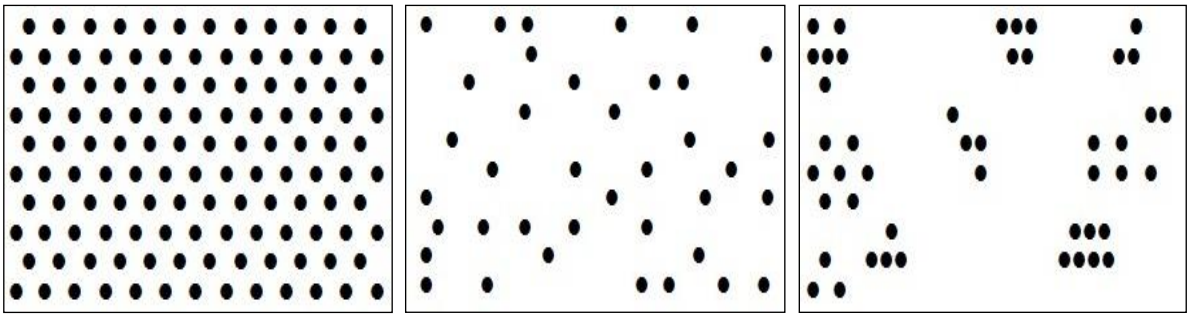
(1) The LGMS-FOA [24], IFFO [15], MFOA [38] and MSFOA [39] lacked the convergence speed to reach the optimal solution. The emphasis of these proposed algorithms was on the effect of gradually changing and adapting the radius range along with the iteration number. For example, the graphs in Figure 2 indicate that the search scope at the beginning was very large, and would gradually be reduced when the iterations increased. If the final solution is the minimum value, these algorithms suffered from slow convergence in their attempt to find a final solution.

(2) The LGMS-FOA, IFFO, MFOA and MSFOA algorithms contain only one type of population that is generated dependent on the technique of algorithms. They have no other method or any special parameter for enhancing the search process. As a result, it is possible for algorithms with only one type of population to become easily trapped at a local optimum. Owing to this disadvantage, several novel optimization algorithms have been developed with more than one population characteristic. This alteration has increased these newer algorithm's abilities in finding global solutions, such as in [5, 40-42].

This paper presents the RO-FOA, which consists of two types of populations for enhancing the diversity of the population and convergence speed. The proposed method was inspired by the characteristics of organisms in a natural ecosystem. The details of the supporting ideas for the RO-FOA creation are in the next section.

3. The relationship of an organism within an ecosystem and the RO-FOA strategy

According to the ecosystem concept [35], organisms within an ecosystem are often visible in different kinds of dispersions, which can vary from area to area. For example, consumers will have dispersion relationships with organisms that are food. There are two types of relationships for organisms. The relationship between organisms of the same species, which is referred to as an intraspecific relationship and the relationship between organisms of different species, which is referred to as an interspecific relationship.



(a) Uniform distribution (b) Random distribution (c) Clumped distribution

Figure 3 Dispersion simulations of same-species organisms in nature.

3.1 The intraspecific relationship

The intraspecific relationships, described in basic ecology [35] are aggregations of same-species organisms, which leads to individual competition, communication, and social interaction. These types of aggregation cause dispersion of the organisms, which in turn creates three types of population distributions.

(1) Uniform distributions are typically found in an environment where resources are limited. The individual organisms compete for the few available resources (spaces). The organism occupies the resource by maximizing the distance between its neighbors, as shown in Figure 3(a).

(2) Random distributions occurs in nature when the environment is unvaried, and the members do not group together. The common distribution of a random distribution differs from uniform distribution, in that the distance between its neighbors is unpredictable, as shown in Figure 3(b).

(3) Clumped distributions, found in natural settings, occur when the environment consists of scattered resources. In order to occupy the resource, the distance between neighbors will be minimized, as shown in Figure 3(c).

The distribution in each population of same-species organisms depends on the limitation of resources in each environment. In order for the organisms to survive within the environment, they must possess the ability to adjust themselves fit into the environment and its available resources in nature.

3.2 The interspecific relationship

The interspecific relationship is divided into nine categories, neutralism, protocooperation, mutualism, commensalisms, predation, parasitism, parasitoid, amensalism and competition. Different species relate to each other by transferring or passing on their energy or matter. This kind of relationship can happen temporarily or happen between two species that always depend on one another. The proposed RO-FOA provides the strategy of mimicking the behaviors of "mutualism", which can be described as follows.

Mutualism – Two species that live together and take benefits from each other. If, however, they are separated, both can still survive. Despite this ability to survive whilst separated, living together would provide more benefits than living alone. An example of this is lichen, which demonstrates the relationship between fungi and chlorophytes. Fungi receive nutrition from chlorophytes, which can generate its own food, while chlorophytes receive moistness from the fungi.

3.3 Concept of RO-FOA

Based on a literature review that studied several optimization algorithms and the disadvantages of the variant FOAs, we found that the difficulty in finding the final solution in complex optimization problems corresponds to the variation and complexity of the domain search space in each problem. The search spaces in optimization problems can be compared to those of the environmental variables within an ecosystem.

For an optimization process, if a population lacks diversity, an optimal solution will not be attained, just as organisms with only one food search pattern, will find it difficult to locate resources (food) necessary for their survival. Therefore, the organisms within an ecosystem survive by co-existing in order to share information, and have diverse populations with more than one pattern. The FOA lacks diversity of exploration. In order to update a new position (in seeking the optimal solution), the FOA determines the scope of the search radius using a random uniform distribution, without change or interruption of information from any special parameters. The variant FOAs (Section 2.3) attempt to rectify this problem through the application of dynamic distributions to improve their search abilities. However, we found that the methods used in the existing improved FOAs remain unsuitable for various optimization problems (see the results in Section 6.6, Table 2). To overcome such disadvantages we propose the RO-FOA, which consists of two types of fruit fly swarms and three distribution patterns of population. The details of which are described as follows:

(1) The first type is the fruit fly swarms generated by the random walk algorithm [42] with a dynamic search range.

(2) The second type is the fruit fly swarms generated by the opposition-based learning algorithm (OBL) [43] to find the opposite information from (1).

(3) Merging of the three distribution patterns (shown in Figure 3) produces a single-procedure RO-FOA algorithm. This process is achieved through the exploration and exploitation activities of (1) and (2) to enhance the search capability of the algorithm.

The algorithmic procedures used in finding the optimal solution are described in the following sub-sections.

3.4. The fruit fly swarms generated by the random walk algorithm

The random walk (RW) algorithm moves positions stochastically when searching for optimal solutions. The main benefits of the RW algorithm are: 1) movement within the search space is flexible, and, therefore, covers

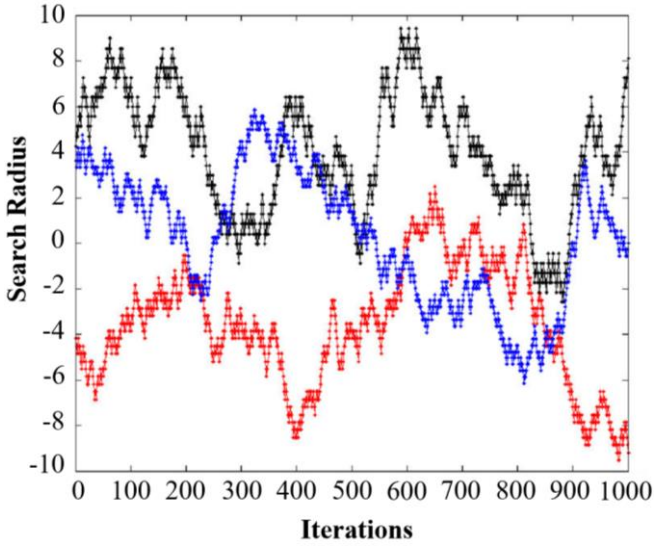


Figure 4 A demonstration of points generated from the random walk algorithm of three starting points

many diverse positions, and 2) the regional radius in the search area is not fixed. The characteristics of the population of RW species movements are described as:

$$\mathrm{x(t)} = [0, \mathrm{cumsum}(2\mathrm{r(t\_1)} - 1), \mathrm{cumsum}(2\mathrm{r(t\_2)} - 1), \ldots, cumsum(2r(t\_n) - 1)] \quad (1)$$

$$r(t) = \begin{cases} 1, & if\ rand > 0.5 \\ 0, & if\ rand \le 0.5 \end{cases},$$

where t denotes the iteration that came to a halt, $r(t)$ is a stochastic function, n is the number of max-iterations, *cumsum* is the cumulative sum, and *rand* is a random uniform point between [0,1]. As in the other swarm algorithms, the RW populations must continually update their positions, and the process is performed through the following equation:

$$\text{Randomwalk:} \quad \mathrm{X_i^t} = \frac{(\mathrm{X_i^t} - \mathrm{a_i}) \times (\mathrm{b_i} - \mathrm{c_i^t})}{\mathrm{d_i^t} - \mathrm{a_i}} + \mathrm{c_i}, \quad (2)$$

where a_i is the minimum boundary, $\mathrm{b_i}$ is the maximum boundary, c_i is the minimum of the *i-th* variable at *t-th* iteration and d_i^t indicates the maximum of the *i-th* variable at the *t-th* iteration. As shown in Figure 4, the position of X fluctuates dramatically within the search space. The three movement lines, red, blue, and black, are simulated from the first three columns of Eq. (2). Here, the X-axis represents the iteration numbers, and the Y-axis represents the search radius (with a boundary position between -10 and 10, iterations = 1000).

The transition from the exploration phase to the exploitation phase is controlled through the radius of the walk. The radii of the swarm's random walks are decreased adaptively with the following equations:

$$c^t = \frac{c^t}{I}, \quad (3)$$

$$d^t = \frac{d^t}{I}, \quad (4)$$

where c^t is the minimum of all variables at t-th iteration and d^t is the the maximum of all variables at t-th iteration. I is a special constant parameter that decreased the radius of the swarm's random walks with an interval ratio. For the RO-FOA processes, $I = 10^{\alpha} \times t/T$, α is a constant that is defined based on the current iteration ($\alpha = 2$ when $t > 0.1 \times T$, $\alpha = 3$ when $t > 0.5 \times T$, $\alpha = 4$ when $t > 0.75 \times T$, $\alpha = 5$ when $t > 0.9 \times T$, and $\alpha = 6$ when $t > 0.95 \times T$) where $t\ and\ T$ are the current iteration and the maximum number of iterations, respectively. A demonstration of Eq. (3) and (4) are presented in Figure 5.

Consequently, the behavior of the RW's populations was similar to that of the group of fruit flies in their search for food within the search space. The RW algorithm generates a diverse fruit fly position within the population through the imagination of RW_X, where X_i^t is a matrix of populations as expressed in Eq. (5).

$$\mathrm{RW\_X} = \mathrm{Randomwalk}\,(\mathrm{X_i^t}), \quad (5)$$

where RW_X is the RW of species swarms.

3.5. The fruit fly swarms generated by opposition-based learning

The basic premise of the OBL algorithm was inspired by the fundamental tasks in *machine intelligence* [43], learning, optimization, and search. Search and optimization techniques start with an initial solution (population), and obtain successive results necessary to achieve an optimal solution. The intrinsic nature of each algorithm is inspired by a specific biological behavior. However, for the current solution x, the OBL premise looks for a new solution in a given problem that is generally estimated as $\overline{\boldsymbol{X}}$. This estimation can be based empirically upon the opposed experience, or a random guess.

The concept of the opposite point is defined as:

Definition 1. Let $x \in \mathbb{R}$be a real number defined on a specific interval: $x \in [a, b]$. The opposite number $\bar{x}$ is defined in the following equation:

$$\bar{x} = a + b - x. \quad (6)$$

while, the opposite number in a multi-dimensional space is defined as:

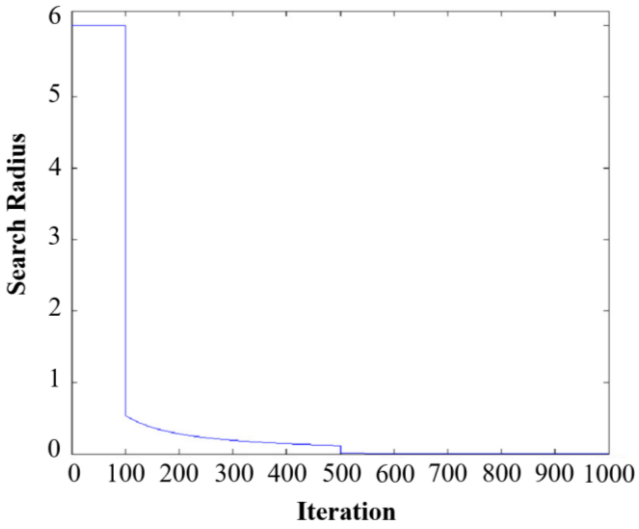


Figure 5 A demonstration of the radius of a swarm's random walks that are adaptively decreased. The X-axis is the number of iterations, set to 1 – 1000 and the Y-axis is the search radius, set to 0 through 6.

Definition 2. Let $P = (x_1, x_2, \ldots, x_D)$ be a point in *D*-dimensional coordinate system, with $x_1, x_2, \ldots, x_D \in \mathbb{R}$ and $x_i \in [a_i, b_i]$. The opposite point $\bar{x}_i$ is defined entirely by its coordinates as follows:

$$\bar{x}_{ij} = a_j + b_j - x_{ij}; j = 1, 2, ..., D \quad (7)$$

According to a review of opposition-based learning [44], there are many swarm intelligence forms based on stochastic optimization methods that are capable of achieving satisfactory performance when their algorithm is embedded with the opposition point generating scheme. In this paper, a new concept, OBL, is proposed for an OP species generation. The OP species is capable of finding a solution in a local region. Additionally, this concept further enhances the search with the faster convergence speeds required in finding the global optimal, in various optimization problems.

In order to achieve the strategy of RO-FOA, we applied OBL to generate the OP species population through each equation as the following procedure.

Let Ub be an upper bound to determine the max of the best position X_axis,

Lb be a lower bound to determine the min of the best position X_axis,

ub be an upper bound of the domain problem,

lb be a lower bound of the domain problem,

Sp (*t*) = *t*/T, a.k.a. a shrinking probability, where *t* is a point in an iteration, and T is the maximum iteration.

$$a_{bound} = \begin{cases} Ub, & Sp(t) < rand \\ ub, & otherwise \end{cases}, \quad (8)$$

$$b_{bound} = \begin{cases} Lb, & Sp(t) < rand \\ lb, & otherwise \end{cases}, \quad (9)$$

$$OP\_X_{ij} = (a_{bound} + b_{bound}) - X\_axis_{ij} \times rand; \; i = 1, 2, ..., sizepop, \; j = 1, 2, ..., D, \quad (10)$$

where OP_X_{ij} is an OP species population generated from OBL, a_{bound}, b_{bound} are the upper bound and lower bound of the algorithm. X_axis is the best position that can be obtained from the initial process of the algorithm, $rand()$ is a random number between [0,1). *sizepop* is the number of populations and *D* is the number of variables (dimension).

The new ideas regarding the proposed OBL are the variation of shrinking probability *Sp* (*t*) versus iterations that is illustrated in Figure 6 and the transition from the exploration to the exploitation phase that is controlled through this probability. The upper bound and lower bound are reduced through *Sp* (*t*). From Eq. (8) and (9), the upper and the lower bounds correspond to a_{bound} and b_{bound}. ub and lb are the max and min values found in the best fruit fly (X_axis), respectively. Ub and Lb are defined from the domain of the problem. Alternating between the two boundaries causes a dynamic change of the radius range. Consequently, the exploration ability could be improved.

4. The proposed random walk and opposition-based learning - fruit fly optimization algorithm

In order to overcome the disadvantages of the existing FOAs, this paper presents the improved RO-FOA, consisting of both the RW and OBL algorithms. The OBL is used to enhance the process of the FOA to achieve a global optimum solution through an accelerated convergence. While the addition of the RW method employs functions designed to enhance the ability of the FOA to create greater diversity than the standard or variant FOAs. The steps in Algorithm 2 outline the RO-FOA procedure and the generation of new parameters.

Regarding the distribution patterns, illustrated in Figure 3(a-c), the RO-FOA divides the entire population into two groups, similar to the concept of a mutualistic relationship, the populations generated from the OBL, and those generated from the RW. For example, when the initial parameter of the population size (*sizepop*) = 50, then each group size = 25 populations. These two groups, OBL and RW, differ from the multi-swarm technique [16] in that the multi-swarm method creates a random uniform distribution, and the number of groups (or sub-swarms) depends on the user definitions, whereas the RO-FOA has only one group and two distribution patterns. Additionally, the computational complexity of the problem does not increase, compared to that of the existing FOAs (as discussed in Section 6.1).

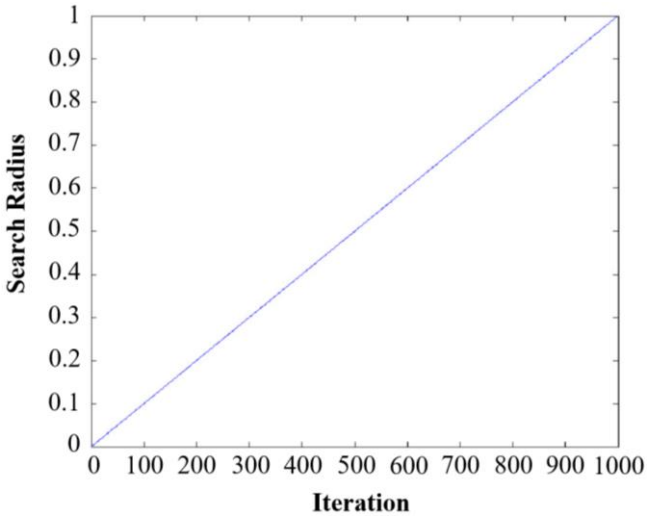


Figure 6 Variation of *Sp(t)* versus iterations.

Algorithm 2: The RO-FOA algorithm

//Initial procedure

Step 1. Determine the initial parameters, population size (*sizepop*), the maximum iteration (*maxgen*), number of variables (*D*), upper bound (*Ub*) and lower bound (*Lb*)

Step 2. Initialize fruit fly swarm positions randomly

$X_i = lb + (ub - lb) \times rand$, *i* = 1, 2, …, *sizepop*

Step 3. Find the best smell concentration (fitness value) by inputting each individual fruit fly's position into the objective function

$Smell_i$ = Objective Function (X_i), [*bestSmell*, *bestIndex*] = min (*Smell*),
Smellbest = *bestSmell*, *X_axis* = *X*(*bestIndex*)

//Main procedure

Step 4. Update *ub* and *lb* using information of *X_axis*

Step 5. $OP\_X_{ij} = (a_{bound} + b_{bound}) - X\_axis_{ij} \times rand$, *i* = 1, 2, …,*sizepop*/2 //Eq. (10)

Step 6. RW species populations are obtained from random walk algorithm

$RW\_X_i^t = Randomwalk\ (X_i^t)$, *i* = 1, 2, …, *sizepop*/2 //Eq. (5)

Step 7. Use the populations for each iteration to form *X* ,
X = *OP_X* ∪ *RW_X*

Step 8. Re-evaluate the smell concentration (fitness value) of the individual fruit fly
$Smell_i$ = Objective Function (X_i), *i* = 1, 2, …, *sizepop*

Step 9. Find the fruit fly with the least minimal smell concentration,

[*new_bestSmell*, *new_bestIndex*] = min (*Smell*),
new_Smellbest = *bestSmell*, *new_X_axis* = *X* (*bestIndex*)

Step 10. If the new smell concentration is better than the previous iterative smell concentration, then update *X_axis* with the current best fruit fly.

Step 11. Enter into iterative optimization.

Repeat steps 4-10. The process will stop when the smell concentration is no longer changed, or when the iteratiion number reaches the maximum value.

Output: the fruit fly with the best smell concentration.

The distribution discussed in Section 3.2 is generated from the following RO-FOA steps:

(1) Step 2 generates the pattern of populations of the uniform distribution (Figure 3 (a)). This strategy is used in the initial step before entering the main RO-FOA procedure.

(2) Step 5 generates the new populations of the uniform distribution. This step provides the new position of the population calculated through the best position *X_axis* obtained from step 3.

(3) Step 6 generates the new populations of the random distribution. This step provides the new position of population calculated through the best position *X_axis* obtained from step 3.

(4) Step 7 - after a period of time, steps 5 and 6 generate the populations of the clumped distribution.

5. The settings and evaluation of algorithms on benchmark functions

In this section, we evaluate the performance of our proposed method through the application of 34 standard benchmark functions (11 uni-modal and 23 multi-modal)

Table 1 The 34 standard benchmark functions.

Function	Type	Function name	Dimension	Optimum
f1	unimodal	Sphere model	30	0
f2	unimodal	Axis parallel hyperellipsoid	30	0
f3	unimodal	Schwefel's problem 1.2	30	0
f4	multimodal	Rosenbrock's valley	30	0
f5	multimodal	Rastrigin's function	30	0
f6	multimodal	Griewank's function	30	0
f7	unimodal	Sum of different power	30	0
f8	multimodal	Ackley's path function	30	0
f9	unimodal	Beale function	2	0
f10	multimodal	Colville function	4	0
f11	unimodal	Easom function	2	-1
f12	multimodal	Hartmann function 1	3	-3.86278
f13	multimodal	Hartmann function 2	6	-3.32237
f14	multimodal	Six Hump Camel back function	2	-1.03162
f15	multimodal	Levy function	30	0
f16	unimodal	Matyas function	2	0
f17	multimodal	Perm function	2	0
f18	multimodal	Michalewicz function	10	-9.66015
f19	multimodal	Zakharov function	10	0
f20	multimodal	Branins' function	2	0.3979
f21	unimodal	Schwefel's problem 2.22	30	0
f22	unimodal	Schwefel's problem 2.21	30	0
f23	unimodal	Step function	30	0
f24	multimodal	Quartic function	30	0
f25	multimodal	Kowalik's function	4	0.0003075
f26	multimodal	Shekel's Family (1)	4	-10.2
f27	multimodal	Shekel's Family (2)	4	-10.4
f28	multimodal	Shekel's Family (3)	4	-10.5
f29	multimodal	Tripod function	2	0
f30	unimodal	De Jong's function 4 (no noise)	2	0
f31	multimodal	Alpine function	30	0
f32	multimodal	Schaffer's function 6	2	0
f33	multimodal	Pathological function	30	0
F34	multimodal	Inverted cosine wave function (Masters)	4	-*n*+1

taken from [45-48]. Each function contains different dimension levels (see Table 2). The definition of each benchmark function and their global optima are listed in Appendix A. Each algorithm was coded in MATLAB. Experiments were conducted using MATLAB 7.10.0 (2010a) on a personal computer, with a 3.2 GHz CPU, 8 GB RAM running a Microsoft Windows 7 operating system. For MSFOA, 15 benchmark functions used in [39] are also used in this paper. Therefore the experimental results relating that 15 functions are taken from Tables 3 and 4 of [39] and are used to compare with that of the proposed RO-FOA.

5.1 Parameters and settings

In the parameter settings, the maximum iteration number (*maxgen*) of each algorithm was fixed at 1000, and the population sizes (*sizepop*) at 50. The mean value (Mean) and standard deviations (*Std*) over 30 independent runs are presented for each algorithm.

(1) Comparison between the RO-FOA and variant FOAs

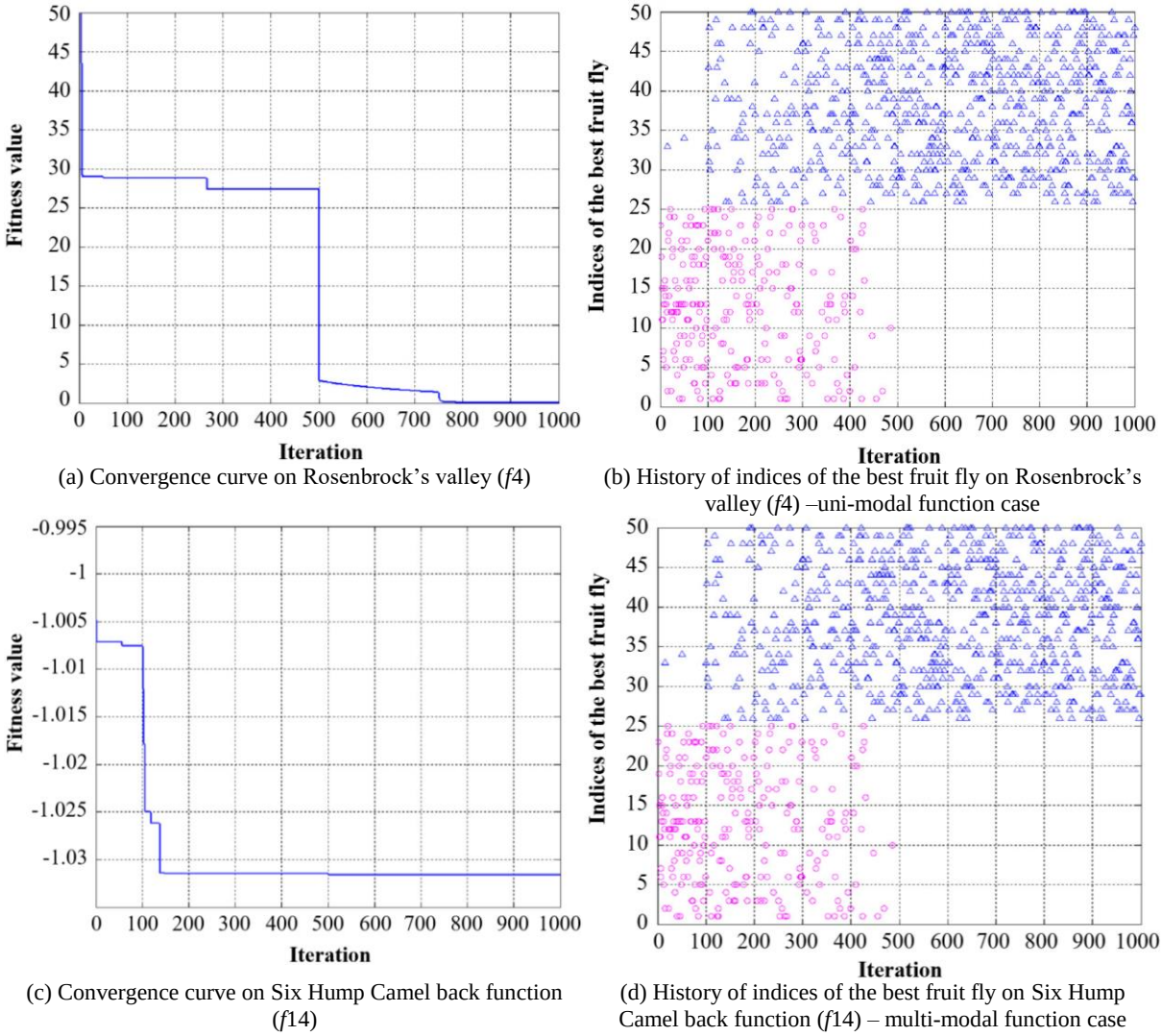
We compared the performance of the RO-FOA and other variant FOAs with the same benchmark functions (Table 2). The individual parameters of the LGMS-FOA [37] are *sizepop* = 50, n = 0.005, $w0$ = 1, and α = 0.95. The IFFO parameters [15] are $\gamma_{max} = UB - LB/2$, and $\gamma_{\min} = 10^{-5}$. Within the parameter settings of the proposed MFOA [16], the multi swarm number, M, is set at = 5, and θ is set at values from 2 to 6.

(2) Comparison between the RO-FOA and the meta-heuristic algorithms

The parameters of the meta-heuristic algorithms include the particle swarm optimization (PSO), where $c1$ = 2, $c2$ = 1.5, w_{max} = 1.3, w_{min} = 0.3, and v_{max} is limited to 20% of the domain [49]. Differential evolution (DE) was done with a scaling factor (F) = 0.5, and the crossover probability constant (CR) = 0.9 [50]. Harmony search (HS) had a harmony memory size (HMS) = *sizepop*. The harmony memory consideration rate (HMCR) = 0.90, and the pitch adjusting rate (PAR) = 0.35 [51]. The gravitational search algorithm (GSA) [52], and the grey wolf optimizer (GWO) [53], flower pollination algorithm (FPA) [54], and a modified flower pollination algorithm (MFPA) [55] were also examined. These algorithms were coded in MATLAB and obtained from https://www.mathworks.com/ matlabcentral/fileexchange/. The default parameters of each algorithm are shown in Appendix B.

6. Benchmarking results and discussion

In brief, this paper presents the RO-FOA, which consists of both the RW and OBL algorithms. The experimental results show that the RO-FOA improves the performance over the original FOA in finding an optimal solution within the optimizing function. Furthermore, the proposed method increases search capability and diversity during exploration, through the use of the RW to expand the population's position, while the OBL accelerates the speed of the



(a) Convergence curve on Rosenbrock's valley (*f*4)

(b) History of indices of the best fruit fly on Rosenbrock's valley (*f*4) –uni-modal function case

(c) Convergence curve on Six Hump Camel back function (*f*14)

(d) History of indices of the best fruit fly on Six Hump Camel back function (*f*14) – multi-modal function case

Figure 7 Graphic of the convergence curves and indices of the best fruit fly history that were produced during two runs of RO-FOA. The triangle was produced by the RW and the circle was produced by the OP.

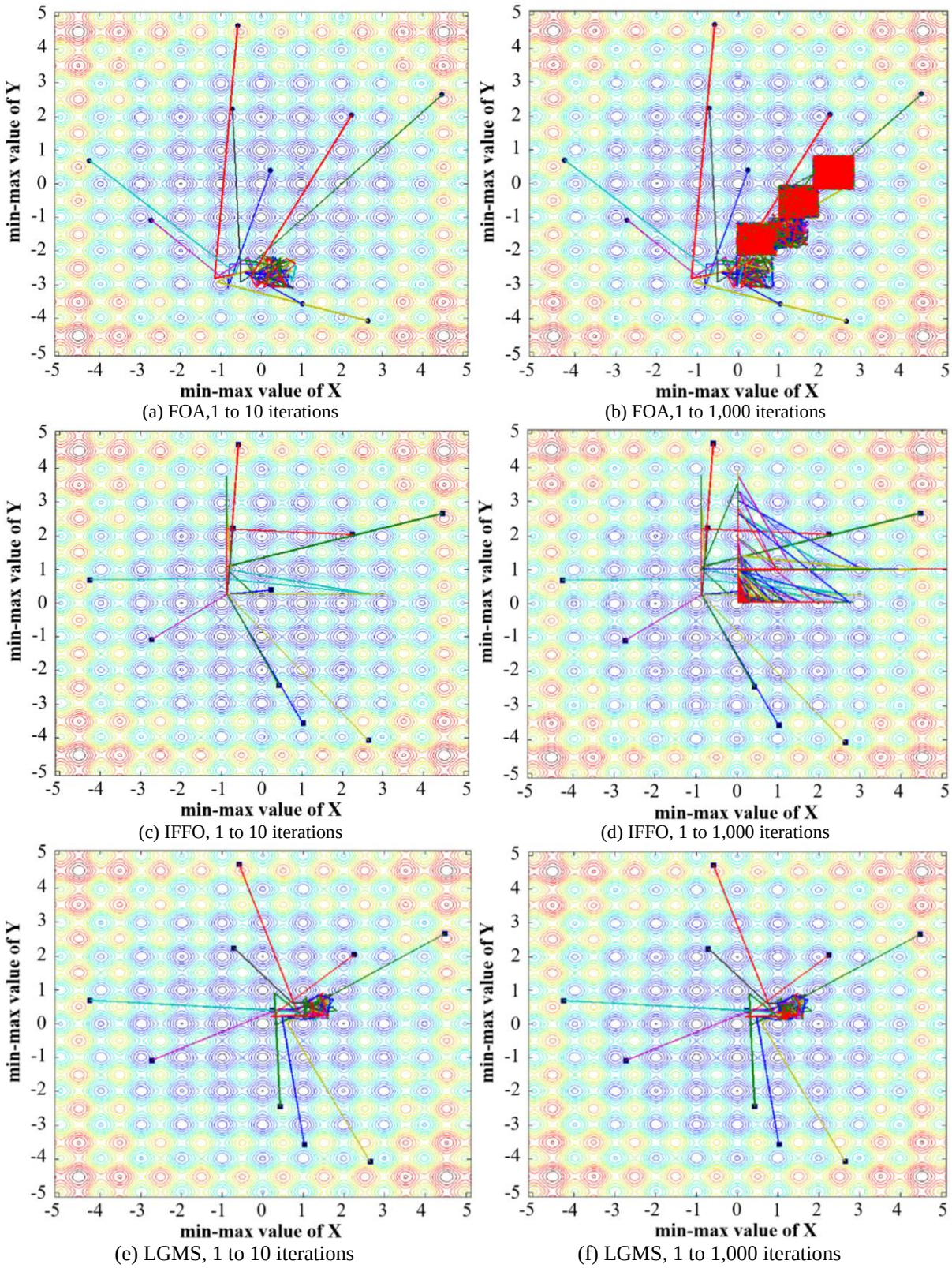
exploration process and convergence in locating the promising region.

6.1 Computational complexities of the RO-FOA and original FOA

The number of function calls (NFCs) determines computational complexity, and it is the most commonly used metric in recent literature [30, 56]. The computational complexity of the original FOA was calculated through the sum of the NFCs of each individual fruit fly, in each of the two phases. The initial phase calculates the cost of a candidate solution to establish the best initial position. The second phase computes the updated position of each individual fruit fly. In this paper, the computational parameters involve a set of population sizes in the initial and the updated phases, in which each phase = 50, and the max iteration = 1000. In the original FOA, the resulting NFCs = 50 + (50 × 1000) = 50050. Within these same computational settings, the proposed RO-FOA generated a population with the best initial position = 50, similar to the initial phase of the original FOA. The updated phase of the RO-FOA then combined the OBL and RW characteristics of each individual fruit fly (Step 5 and 6), resulting in a population = 50. The complexity of the RO-FOA is therefore calculated as (50) + (25+25 × 1000), and the NFCs = 50050. In the proposed of RO-FOA, the fruit fly's behaviors (Steps 7), inspired from previous research [5, 40-42], proves more capable of finding the optimal solution than with a single characteristic. This technique can further enhance the diversity of the population of the algorithms' exploration capabilities, and has the potential to accelerate convergence to quickly obtain the optimal solution. In summary, we may conclude that the complexity of the RO-FOA is not significantly different from that of the original FOA, as the RO-FOA employed two populations (OBL and RW). Yet the population size did not increase. In the next section, we discuss the capabilities of the RO-FOA through the evaluation of several commonly used benchmark functions, and compare them with other meta-heuristic algorithms.

6.2 The benefit of having both RW and OBL together

In Figure 7, each graph was generated by RO-FOA, including both RW and OBL. The left column shows the convergence curve while the right column shows the historical indices of best fruit flies produced from the Rosenbrock's valley and Six Hump Camel back functions,



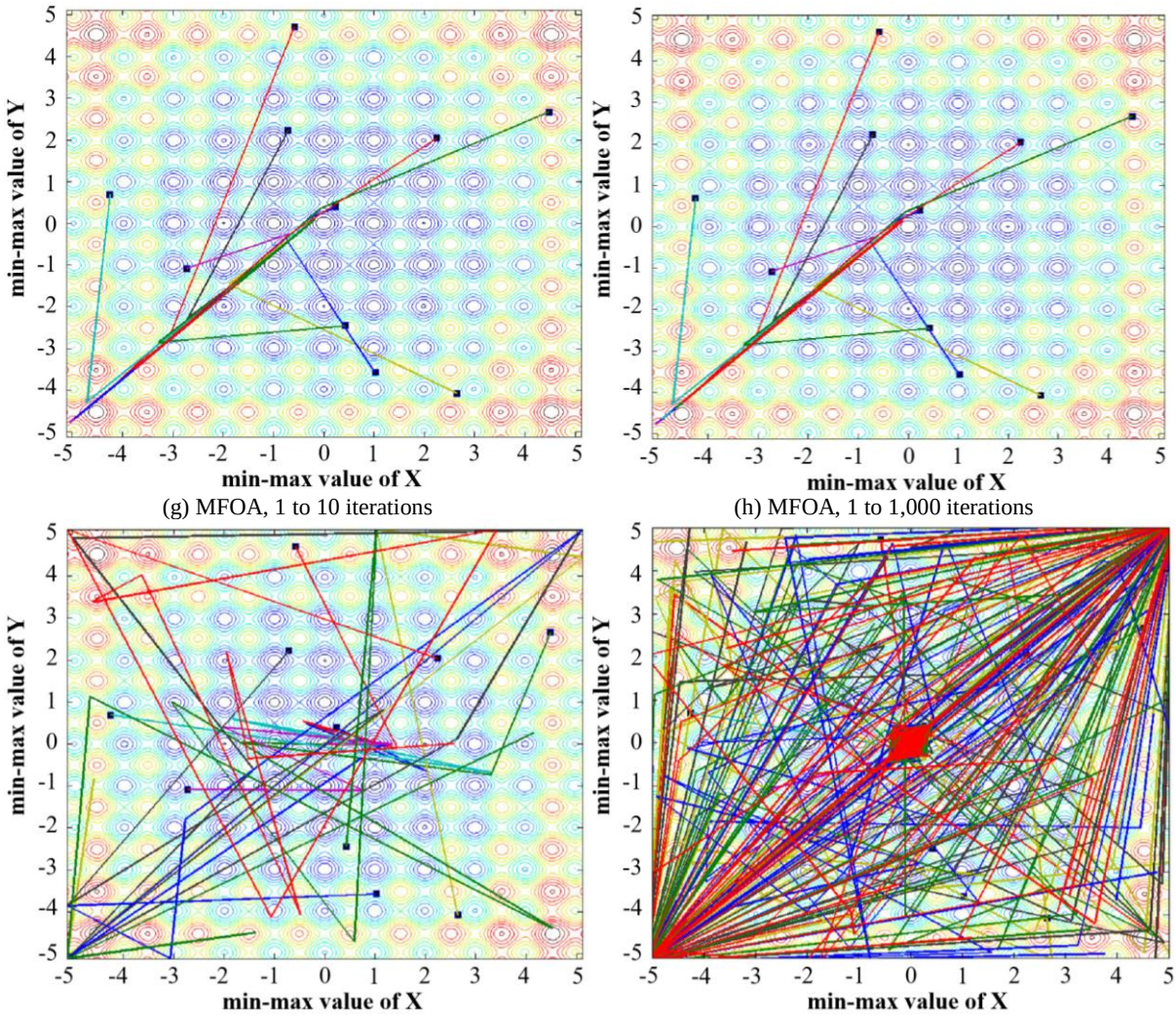
(a) FOA,1 to 10 iterations (b) FOA,1 to 1,000 iterations

(c) IFFO, 1 to 10 iterations (d) IFFO, 1 to 1,000 iterations

(e) LGMS, 1 to 10 iterations (f) LGMS, 1 to 1,000 iterations

Figure 8 The *f*5 best search routes produced by the RO-FOA and variant FOA algorithms

respectively. As can be seen in Figures 7 (a) and (c), the blue lines show that the convergence curve tends to continue reducing and converging quickly to the solution. In Fig. 7 (b) and (d), the pink dots are the best positions obtained from OP species and the blue triangles are the best positions obtained from RW species. In Fig. 7 (b), the Rosenbrock's valley function is a uni-modal function. Successful exploration is done by the OP species and successful exploitation is done by the RW species, in the early and e latter phases, respectively. In Fig. 7 (c), the Six Hump Camel back function is a multi-modal function. In the early phase, successful exploration is done by cooperation of the



(g) MFOA, 1 to 10 iterations (h) MFOA, 1 to 1,000 iterations

(i) RO-FOA,1 to 10 iterations (j) RO-FOA, 1 to 1,000 iterations

Figure 8 (continued) The *f*5 best search routes produced by the RO-FOA and variant FOA algorithms.

two species. When the algorithms evolve to the exploitation phase, successful exploitation is done by RW species.

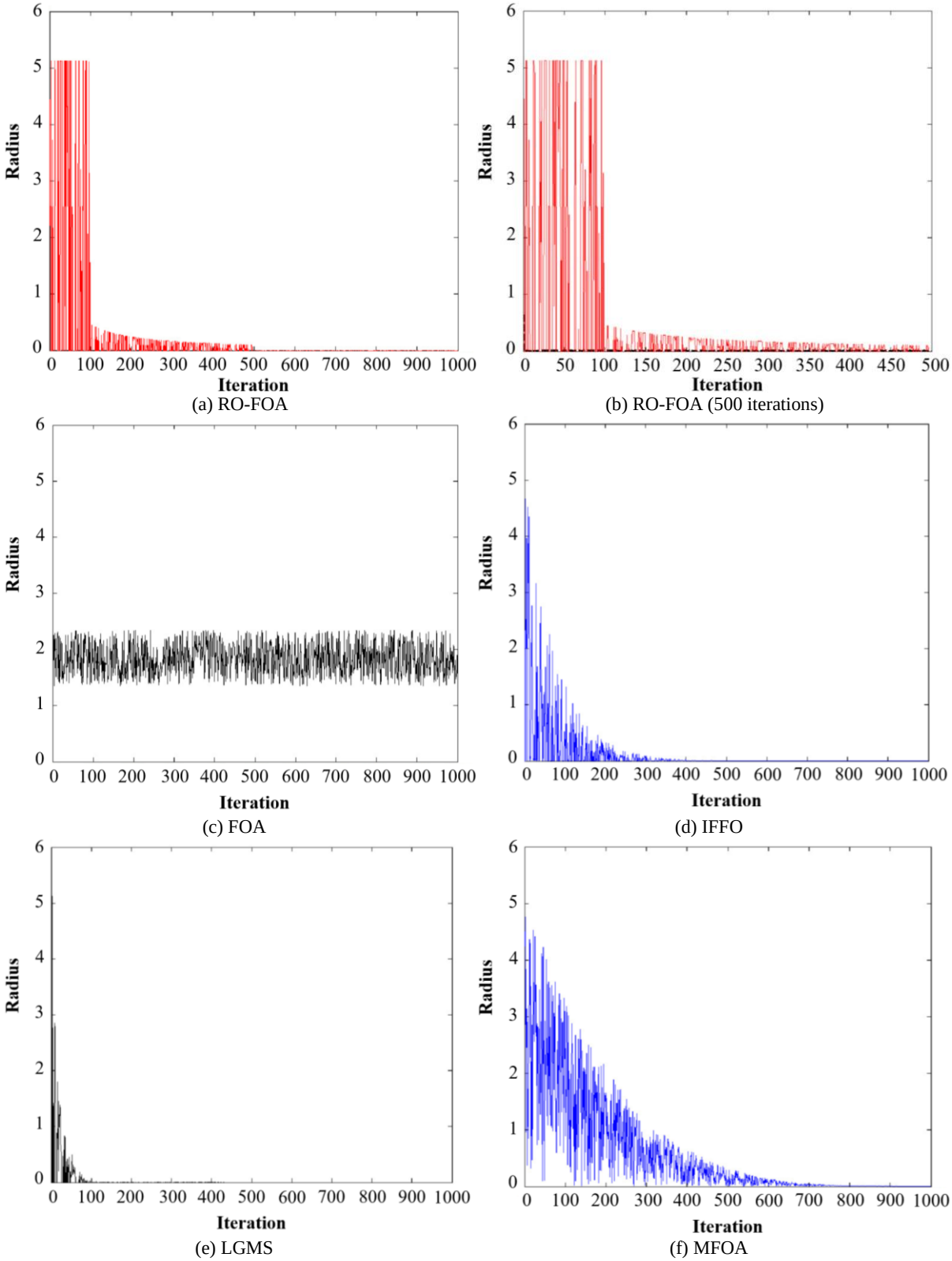
6.3 Enhancement of population diversity

To show enhancement of the diversity of exploration of the RO-FOA, the best search routes of the RO-FOA and those of the variant FOA algorithms are presented in Figure 8. Each graph displays the results rendered from the multi-modal Rastrigin function (*f*5). The *f*5 details are elaborated in Table 1 and Appendix A. The figures illustrate the populations' starting positions through the end of the search processes in the original FOA, IFFO, LGMS, MFOA, and RO-FOA algorithms. The simulation parameters were set at population=10, the iteration numbers ran from 1 to 10 in the left column, and 1 to 1,000 in the right column, where the X- and Y-axes represent the two dimensional position of the population within the range of the search space between -5.12 and 5.12. Each algorithm population began with the same value, employing a random uniform distribution. The optimal solution of *f*5= (0, 0).

The RO-FOA algorithm significantly enhances the diversity of exploration by spreading and moving its population search, compared with the other variant FOAs. The figures of the proposed RO-FOA (Figures 8 (i) and 8 (j)) show the highest population diversity and the most accurate solution. Within the first ten iterations, shown in the left column of Figure 8, the population diversities of the original FOA, IFFO, LGMS, and MFOA gradually change or adapt, depending on the iteration number. They tend to align with one another or move in a slightly different direction. In contrast, the RO-FOA (Figure 8 (i)) produces a dramatic fluctuation of the population, and refracts, thereby improving its position. When expanding the RO-FOA's iterations to 1,000 (Figure 8 (j)), the population continues to demonstrate a more positive refracted direction, and is able to obtain the optimal solution.

6.4 Comparison of the adjusted radius within the RO-FOA and the variant FOA algorithms

As discussed above, a primary disadvantage of the variant FOAs is the way in which they adjust or change the radius in both exploration and exploitation. The RO-FOA strategy adjusts the radius within the course of its iterations, through two techniques, which produce a substantial advantage. The first technique adjusts the interval of the RW. The second calculates probabilities through opposition based learning, in adapting the radius within the course of iterations. As a demonstration, the 2-D Rastrigin function



(a) RO-FOA (b) RO-FOA (500 iterations)

(c) FOA (d) IFFO

(e) LGMS (f) MFOA

Figure 9 The adjusted radii of the RO-FOA, FOA, IFFO, LGMS, and MFOA over the course iteration.

(*f*5) was selected, and solved with 50 populations, over 1,000 iterations, as depicted in Figure 9 (with the exception of Figure 9 (b), which is enlarged to examine the RO-FOA through 500 iterations). The X-axis represents the number of iterations, and the Y-axis represents the boundary values. In Figure 9 (a), the red line demonstrates the wider radius of the RO-FOA in the first 100 iterations and an increase in the number of shorter radii of the final process in subsequent iterations. This behavior is a result of the RW algorithm, Eq. (2), which provides a more advantageous exploration and exploitation, and higher solution accuracy. Figure 9(b), examines only 500 iterations and more closely examines

Table 2 Solution quality comparisons of the RO-FOA and variant FOAs within 34 benchmark functions. For the MSFOA, its results were taken from [39]. Lower is better. N/A means no available data. Minimum Means and Stds are in bold. The *h* values signify the results of the rank-sum test.

Function	Criteria	FFO	LGMS_FOA	IFFO	MFOA	MSFOA	RO-FOA	+/−/≈
	Mean	1.51e+02	5.77e+00	2.45e-04	4.48e-02	1.70e+00	**0**	
*f*1	Std	3.88e+01	6.07e-01	1.96e-02	1.04e-01		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	1.42e-08	3.22e-02	9.50e-09	2.76e-02	8.98e-01	**0**	
*f*2	Std	2.73e-08	5.30e-02	1.84e-03	4.10e-02		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	4.08e+02	2.99e+01	1.16e-01	9.89e-12	3.09e+00	**0**	
*f*3	Std	2.55e+02	3.01e+01	7.44e+00	1.77e-11		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	9.32e+02	6.32e+02	2.81e+01	4.47e+01	2.89e+01	**1.17e-06**	
*f*4	Std	4.62e+02	7.04e+02	1.54e+00	3.37e+01		**1.16e-06**	
	h	1	1	1	1	1		5/0/0
	Mean	9.63e+01	2.39e+01	2.00e-02	1.47e+01	8.82e-01	**0**	
*f*5	Std	1.88e+01	8.41e+00	6.04e-01	6.88e+00		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	1.06e+00	9.77e-03	9.73e-02	2.47e-04	9.66e-01	**0**	
*f*6	Std	1.25e-02	9.83e-03	2.74e-01	1.35e-03		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	1.65e-03	1.70e-38	2.83e-159	8.54e-56	1.60e−09	**0**	
*f*7	Std	8.21e-03	8.52e-38	6.18e-57	4.68e-55		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	1.01e+01	1.48e+00	6.08e-02	1.29e+00	5.92e-01	**8.88e-16**	
*f*8	Std	5.01e-01	8.51e-01	4.44e-01	1.36e+00		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	5.34e+00	3.92e-03	6.57e-02	2.86e-01		**1.30e-13**	
*f*9	Std	4.96e+00	5.38e-03	4.70e-02	1.43e+00	N/A	**1.47e-13**	
	h	1	1	1	1			4/0/0
	Mean	9.32e+03	7.09e-01	1.76e+00	4.66e+01		**3.33e-07**	
*f*10	Std	1.50e+04	1.16e+00	1.12e+01	6.05e+00	N/A	**5.51e-07**	
	h	1	1	1	1			4/0/0
	Mean	-0.19	-0.99	-0.99	-8.34e-09		**-1**	
*f*11	Std	3.34e-01	3.33e-03	2.42e-01	2.13e-08	N/A	**0**	
	h	1	1	1	1			4/0/0
	Mean	-2.04444	-3.53052	-3.82067	-0.06797		**-3.86278**	
*f*12	Std	6.78e-01	3.14e-01	3.28e-02	0	N/A	**4.52e-16**	
	h	1	1	1	1			4/0/0
	Mean	-1.82364	-2.39944	-2.92143	-1.33252		**-3.03862**	
*f*13	Std	2.51e-01	2.28e-01	6.81e-02	6.78e-16	N/A	**1.17e-02**	
	h	1	1	1	1			4/0/0
	Mean	20.9856	-0.94249	-1.0280	-0.82828		**-1.0316**	
*f*14	Std	7.11e+01	7.84e-02	1.04e-01	1.87e-01	N/A	**6.78e-16**	
	h	1	1	1	1			4/0/0
	Mean	8.94e+00	2.82e+01	2.47e+00	3.31e+00		**2.22e-12**	
*f*15	Std	7.34e+00	1.27e+01	1.59e-01	7.13e-02	N/A	**1.83e-12**	
	h	1	1	1	1			4/0/0
	Mean	3.02e+00	1.32e-03	3.20e-09	1.90e-04		**0**	
*f*16	Std	4.09e+00	1.51e-03	1.39e-04	2.57e-04	N/A	**0**	
	h	1	1	1	1			4/0/0
	Mean	7.93e+04	1.24e-01	9.59e+04	4.65e+04		**7.30e-04**	
*f*17	Std	2.32e+04	2.18e-01	7.03e+02	2.42e+02	N/A	**6.86e-04**	
	h	1	1	1	1			4/0/0
	Mean	-2.61898	-2.85910	-4.94701	-1.10669		**-8.87573**	
*f*18	Std	5.79e-01	4.88e-01	3.71e-01	3.79e-01	N/A	**4.16e-01**	
	h	1	1	1	1			4/0/0
	Mean	2.37e+02	8.55e+07	1.88e+11	1.13e+02		**8.03e-13**	
*f*19	Std	5.30e+01	2.43e+08	2.37e-03	3.50e+02	N/A	**1.87e-12**	
	h	1	1	1	1			4/0/0
	Mean	11.7302	0.4520	0.3991	37.6936		**0.3979**	
*f*20	Std	1.29e+01	7.30e-02	8.84e-02	4.32e+00	N/A	**2.78e-13**	
	h	1	1	1	1			4/0/0
	Mean	7.48e+09	7.20e+01	1.23e-08	8.03e-10	2.46e-05	**0**	
*f*21	Std	4.08e+10	3.96e+01	1.85e-01	5.42e-10		**0**	
	h	1	1	1	1			4/0/0
	Mean	4.74e+00	1.61e+01	8.10e-09	7.45e-08	4.17e-01	**0**	
*f*22	Std	2.04e-01	7.33e+00	4.38e-01	9.59e-08		**0**	
	h	1	1	1	1			4/0/0
	Mean	2.21e+02	1.09e+01	6.47e+01	3.33e-02	0	**0**	
*f*23	Std	3.91e+01	4.73e+00	1.35e+02	1.83e-01		**0**	
	h	1	1	1	1			4/0/1
	Mean	1.27e+02	1.11e-01	1.54e-03	4.20e-03	7.60e−08	**2.61e-09**	
*f*24	Std	5.95e+01	2.71e-02	1.29e-02	1.55e-03		**2.08e-09**	
	h	1	1	1	1	1		5/0/0
	Mean	13.12	0.0054480	0.0008350	0.0012568		**0.0003079**	
*f*25	Std	2.68e+01	8.37e-03	4.60e-03	3.62e-03	N/A	**5.40e-07**	
	h	1	1	1	1			4/0/0
	Mean	-0.2	-1.1	-6.8	-8.6		**-10.2**	
*f*26	Std	2.23e-01	6.78e-01	8.87e-01	2.37e+00	N/A	**3.61e-15**	
	h	1	1	1	1			4/0/0

Table 2 (continued) Solution quality comparisons of the RO-FOA and variant FOAs within 34 benchmark functions. For the MSFOA, it results were taken from [39]. Lower is better. N/A means no available data. Minimum Means and Stds are in bold. The *h* values signify the results of the rank-sum test.

Function	Criteria	FFO	LGMS_FOA	IFFO	MFOA	MSFOA	RO-FOA	+/–/≈
	Mean	-0.2	-1.4	-6.2	-9.5		**-10.4**	
*f*27	Std	1.55e-01	8.58e-01	7.87e-01	2.01e+00	N/A	**0**	
	h	1	1	1	1			4/0/0
	Mean	-0.3	-2.0	-5.8	-9.9		**-10.5**	
*f*28	Std	1.39e-01	1.62e+00	6.61e-01	1.65e+00	N/A	**3.71e-10**	
	h	1	1	1	1			4/0/0
	Mean	7.91e+00	8.00e-01	4.42e+00	6.33e-01		**6.80e-06**	
*f*29	Std	1.12e+01	8.47e-01	4.70e-01	6.69e-01	N/A	**3.12e-06**	
	h	1	1	1	1			4/0/0
	Mean	1.59e-18	1.25e-99	0	2.83e-81		**0**	
*f*30	Std	3.92e-18	6.05e-99	3.11e-13	8.12e-81	N/A	**0**	
	h	1	1	1	1			4/0/0
	Mean	5.70e+01	6.83e+00	1.19e-09	5.16e-03	5.36e-02	**0**	
*f*31	Std	1.88e+01	2.04e+00	8.49e-02	1.42e-02		**0**	
	h	1	1	1	1	1		5/0/0
	Mean	4.25e-01	2.82e-02	1.57e-02	1.02e-02		**0**	
*f*32	Std	1.21e-01	2.80e-02	2.43e-02	1.99e-02	N/A	**0**	
	h	1	1	1	1			4/0/0
	Mean	1.51e-02	3.60e-06	5.74e-09	-4.81e-17	4.46e+00	**0**	
*f*33	Std	1.69e-02	1.05e-05	7.40e-04	1.06e-15		**0**	
	h	1	1	1	1	1		4/0/0
	Mean	-20.70	-23.13	-23.89	-13.72	-28.92	**-28.93**	
*f*34	Std	6.61e-01	5.03e-01	1.12e-01	4.49e-01		**2.33e-01**	
	h	1	1	1	1			4/0/0
Total								146/0/1

Table 3 Solution quality comparisons of the RO-FOA with seven meta-heuristic algorithms on 34 benchmark functions. Lower is better. Minimum Means and Stds are in bold. The right most column is the results from the rank-sum test. The *h* values signify the results of the rank-sum test.

Function	Criteria	PSO	HS	GSA	DE	GWO	FPA	MFPA	RO-FOA	+/–/≈
	Mean	1.86e-12	1.42e+01	2.00e-17	6.67e-17	5.87e-73	2.47e+01	1.20e-04	**0**	
*f*1	Std	8.64e-12	2.95e+00	6.44e-18	8.82e-17	1.30e-72	3.38e+00	4.53e-04	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	5.94e-77	3.68e-05	7.19e-122	0	0	1.66e-02	3.17e-110	**0**	
*f*2	Std	2.38e-76	4.81e-05	3.90e-121	0	0	3.92e-02	1.65e-109	**0**	
	h	1	1	1	0	0	1	1		**5/0/2**
	Mean	1.83e-03	1.06e+04	1.81e+01	1.14e-03	1.47e-29	4.02e+01	8.70e+00	**0**	
*f*3	Std	3.69e-03	1.65e+03	8.62e+00	4.82e-03	7.21e-29	1.34e+01	1.04e+01	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	2.03e+01	4.91e+02	2.61e+01	2.51e+01	2.67e+01	5.48e+03	7.39e+01	**1.17e-06**	
*f*4	Std	1.89e+00	1.03e+02	1.82e+01	1.28e+00	7.83e-01	1.56e+03	6.28e+01	**1.16e-06**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	9.42e+00	1.08e+01	3.02e+00	1.02e+01	0	1.84e+02	5.76e+01	**0**	
*f*5	Std	4.57e+00	2.70e+00	1.72e+00	7.55e+00	0	1.78e+01	1.22e+01	**0**	
	h	1	1	1	1	0	1	1		**6/0/1**
	Mean	5.07e-02	5.39e+01	4.45e+00	2.79e-03	3.67e-03	6.81e-01	2.20e-02	**0**	
*f*6	Std	1.31e-01	1.14e+01	2.11e+00	5.63e-03	7.44e-03	5.37e-02	3.19e-02	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	0	1.33e-72	0	0	0	8.94e-41	0	**0**	
*f*7	Std	0	7.27e-72	0	0	0	3.25e-40	0	**0**	
	h	0	1	0	0	0	1	0		**2/0/5**
	Mean	1.42e+00	1.35e+01	3.51e-09	3.10e-02	1.36e-14	4.78e+00	3.95e+00	**8.88e-16**	
*f*8	Std	1.07e+00	8.32e-01	4.92e-10	1.70e-01	2.75e-15	2.85e-01	2.14e+00	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	5.08e-02	1.30e-02	7.27e-21	0	2.46e-08	8.33e-01	0	**1.30e-13**	
*f*9	Std	1.93e-01	2.72e-02	7.24e-21	0	2.32e-08	3.52e-01	0	**1.47e-13**	
	h	1	1	1	-1	1	1	-1		**5/2/0**
	Mean	3.16e-05	4.42e+01	1.18e+00	2.68e-05	6.02e-01	2.47e+01	4.00e-24	**3.33e-07**	
*f*10	Std	3.60e-05	7.04e+01	1.58e+00	1.47e-04	1.34e+00	2.09e+01	1.20e-23	**5.51e-07**	
	h	1	1	1	1	1	1	-1		**6/1/0**
	Mean	-1	-0.18	-0.95	-1	-1	-0.01	-1	**-1**	
*f*11	Std	6.60e-02	3.64e-01	1.97e-01	0	6.68e-08	5.68e-03	0	**0**	
	h	1	1	1	0	1	1	0		**5/0/2**
	Mean	-3.86202	-3.86202	-3.86278	-3.86278	-3.86176	-1.96715	-3.86278	**-3.86278**	
*f*12	Std	1.90e-04	7.90e-04	3.16e-15	3.12e-15	2.60e-03	9.23e-01	3.16e-15	**4.52e-16**	
	h	1	1	1	1	1	1	0		**6/1/0**
	Mean	-3.01636	-3.01636	-3.01246	-2.99125	-3.01995	-1.37914	-3.02812	**-3.04862**	
*f*13	Std	3.10e-02	1.65e-02	1.36e-15	2.33e-02	3.36e-02	9.95e-02	2.64e-02	**1.17e-02**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	-1.0316	-1.0315	-1.0316	-1.0316	-1.0316	-0.8917	-1.0316	**-1.0316**	
*f*14	Std	4.52e-16	1.37e-04	4.52e-16	4.52e-16	2.23e-09	1.44e-01	4.52e-16	**6.78e-16**	
	h	0	1	0	0	1	1	0		**3/0/4**
	Mean	4.42e-02	4.42e-02	1.35e-31	1.79e-01	7.91e-08	2.27e-01	1.35e-31	**2.22e-12**	
*f*15	Std	6.53e-01	8.95e-02	6.68e-47	4.59e-01	7.61e-08	1.68e-01	6.68e-47	**1.83e-12**	
	h	1	1	0	1	1	1	0		**5/0/2**

Table 3 (continued) Solution quality comparisons of the RO-FOA with seven meta-heuristic algorithms on 34 benchmark functions. Lower is better. Minimum Means and Stds are in bold. The right most column is the results from the rank-sum test. The *h* values signify the results of the rank-sum test.

Function	Criteria	PSO	HS	GSA	DE	GWO	FPA	MFPA	RO-FOA	+/–/≈
	Mean	1.26e-65	9.36e-03	6.32e-89	2.29e-23	2.20e-28	4.54e-03	1.33e-55	**0**	
f16	Std	6.77e-65	1.42e-02	1.71e-88	0	0	4.76e-03	7.14e-55	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	8.87e+04	8.54e+04	9.16e+04	4.43e-02	5.93e-01	9.33e+04	9.47e-04	**7.30e-04**	
*f*17	Std	2.80e+03	5.13e+02	4.57e+03	1.22e-01	6.89e-01	4.91e+03	8.77e-04	**6.86e-04**	
	h	1	1	1	1	1	1	0		**6/0/1**
	Mean	-7.93635	-8.00890	-8.62540	-5.13600	-7.65460	-8.54994	-8.57597	**-8.87573**	
*f*18	Std	7.86e-01	4.34e-01	5.07e-01	1.38e+00	8.92e-01	2.61e+00	5.03e-01	**4.16e-01**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	1.10e+01	3.64e+02	5.18e+01	1.23e+00	2.14e-27	8.33e+00	2.64e-10	**8.03e-13**	
*f*19	Std	2.37e+01	5.40e+01	1.09e+01	1.03e+00	3.33e-27	1.09e+00	6.68e-10	**1.87e-12**	
	h	1	1	1	1	-1	1	1		**6/1/0**
	Mean	0.3979	0.3980	0.3979	0.3979	0.3979	8.3250	0.3978	**0.3979**	
*f*20	Std	0	9.95e-05	0	3.71e-02	5.86e-05	1.09e+00	0	**2.78e-13**	
	h	0	1	0	0	0	1	0		**2/0**/5
	Mean	7.07e-02	1.85e+01	2.18e-08	5.78e-08	4.08e-41	2.14e+01	8.28e-02	**0**	
*f*21	Std	1.83e-01	2.40e+00	3.17e-09	5.14e-08	4.50e-41	2.24e+00	3.87e-01	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	6.01e-01	5.87e+01	3.74e-02	1.26e+01	2.79e-17	1.70e+00	2.07e+01	**0**	
*f*22	Std	4.29e-01	5.95e+00	2.05e-01	6.17e+00	6.01e-17	9.47e-02	5.49e+00	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	6.57e+00	5.15e+03	0	1.33e-01	0	2.37e+01	1.47e+00	**0**	
*f*23	Std	9.84e+00	8.44e+02	0	3.46e-01	0	4.68e+00	6.20e+00	**0**	
	h	1	1	0	1	0	1	1		**5/0/2**
	Mean	1.32e-02	3.31e+00	2.07e-02	4.18e-02	5.33e-04	5.84e+02	9.62e-02	**2.61e-05**	
*f*24	Std	6.75e-03	1.03e+00	8.90e-03	9.42e-03	2.96e-04	1.84e+02	3.79e-02	**2.08e-05**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	0.000499	0.003529	0.002330	0.000374	0.002405	0.035233	0.0003074	**0.0003079**	
*f*25	Std	4.29e-04	5.24e-03	1.16e-03	2.33e-04	6.09e-03	3.36e-02	1.83e-19	**5.40e-07**	
	h	1	1	1	1	1	1	0		**6/0/1**
	Mean	-4.6	-5.7	-7.0	-9.8	-9.6	-1.5	-10.1	**-10.2**	
*f*26	Std	2.75e+00	3.51e+00	3.63e+00	1.28e+00	1.55e+00	8.62e-01	6.45e-15	**3.61e-15**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	-7.0	-6.2	-10.4	-10.4	-10.4	-1.3	-10.4	**-10.4**	
*f*27	Std	3.69e+00	3.60e+00	7.38e-16	0	1.87e-04	7.76e-01	5.71e-16	**0**	
		1	1	0	0	1	1	1		**5/0/2**
	Mean	-5.0	-5.5	-5.0	-10.3	-10.3	-1.38	-10.5	**-10.5**	
*f*28	Std	3.15e+00	2.86e+00	1.40e+00	1.18e+00	1.14e+00	7.42e-01	3.83e-15	**3.71e-10**	
	h	1	1	1	1	1	1	0		**6/0/1**
	Mean	5.00e-01	2.21e+00	5.33e-01	1.0e-181	7.38e-01	4.82e+01	3.26e-41	**6.80e-06**	
*f*29	Std	6.30e-01	1.38e+00	7.30e-01	0	7.39e-01	1.85e-01	5.96e-41	**3.12e-06**	
	h	1	1	1	-1	1	1	-1		**5/2/0**
	Mean	1.3e-161	4.66e-12	2.80e-51	0	0	4.85e-06	9.0e-199	**0**	
*f*30	Std	0	1.14e-11	1.33e-50	0	0	2.06e-05	0	**0**	
	h	1	1	1	0	0	1	0		**4/0/3**
	Mean	4.24e-03	1.03e+01	2.11e-09	3.56e-04	6.41e-05	1.62e+01	2.44e-02	**0**	
*f*31	Std	1.16e-02	2.06e+00	2.99e-10	9.39e-04	1.62e-04	2.93e+00	9.23e-02	**0**	
		1	1	1	1	1	1	1		**7/0/0**
	Mean	0	1.37e-01	8.19e-03	0	0	4.76e-02	7.83e-03	**0**	
*f*32	Std	0	1.12e-01	8.74e-03	0	0	3.56e-02	4.29e-02	**0**	
		0	1	1	0	0	1	1		**4/0/3**
	Mean	-2.6e-16	1.6e-02	2.0e-03	-1.1e-15	1.2e-05	1.6e-03	4.3e-05	**0**	
*f*33	Std	2.92e-16	2.03e-02	4.17e-03	8.52e-16	1.89e-05	1.19e-03	9.38e-05	**0**	
	h	1	1	1	1	1	1	1		**7/0/0**
	Mean	-3.19	-3.07	-3.00	-3.50	-3.41	-2.42	-2.75	**-3.53**	
*f*34	Std	3.00e-01	2.60e-01	1.51e-01	1.81e-15	2.56e-01	2.41e-01	1.42e-01	**2.33e-01**	
	h	1	1	1	1	1	1	1		**7/0/0**
Total										**197/5/26**

the adjusted radii created through the opposition algorithm (black dots), Eq. (7), where the radius is shorter than that from the RW algorithm. Based on these two radii, the RO-FOA produces a superior search solution to the variant FOAs that possess only a single technique to adjust the radius. Figures 9 (c-f) illustrate the adjusted radii through the course of any iteration produced by the variant FOAs. In Figure 9 (d-f), the radii illustrate continuously decreasing boundaries, each of which is generated by only one mechanism.

6.5 The solution quality comparisons of the RO-FOA and the variant FOA algorithms

The solution quality of each algorithm's search (and success) in locating the global optimum in each of the 34 benchmark functions is presented in Table 1. The final objective values produced from each algorithm are represented by their mean value (Mean) and standard deviation (Std). The *h* values signify the results of the rank-sum test in which an *h* value equal to 1 or -1, represented by the symbols (+) and (-), respectively, are given in the last column of Tables 2 and 3. The RO-FOA may therefore be viewed as being significantly better or worse, than the competing algorithms. The best results in each function are highlighted in bold. However, a comparison of two algorithms in which the *h* value is equal to zero (≈), would indicate that the algorithms are not statistically different.

From Table 2, the RO-FOA outperformed each of the five algorithms, as evidenced by its lowest average final Means and Stds. For example, in function *f*1, the mean final objective values attained by the FFO (1.51E+02),

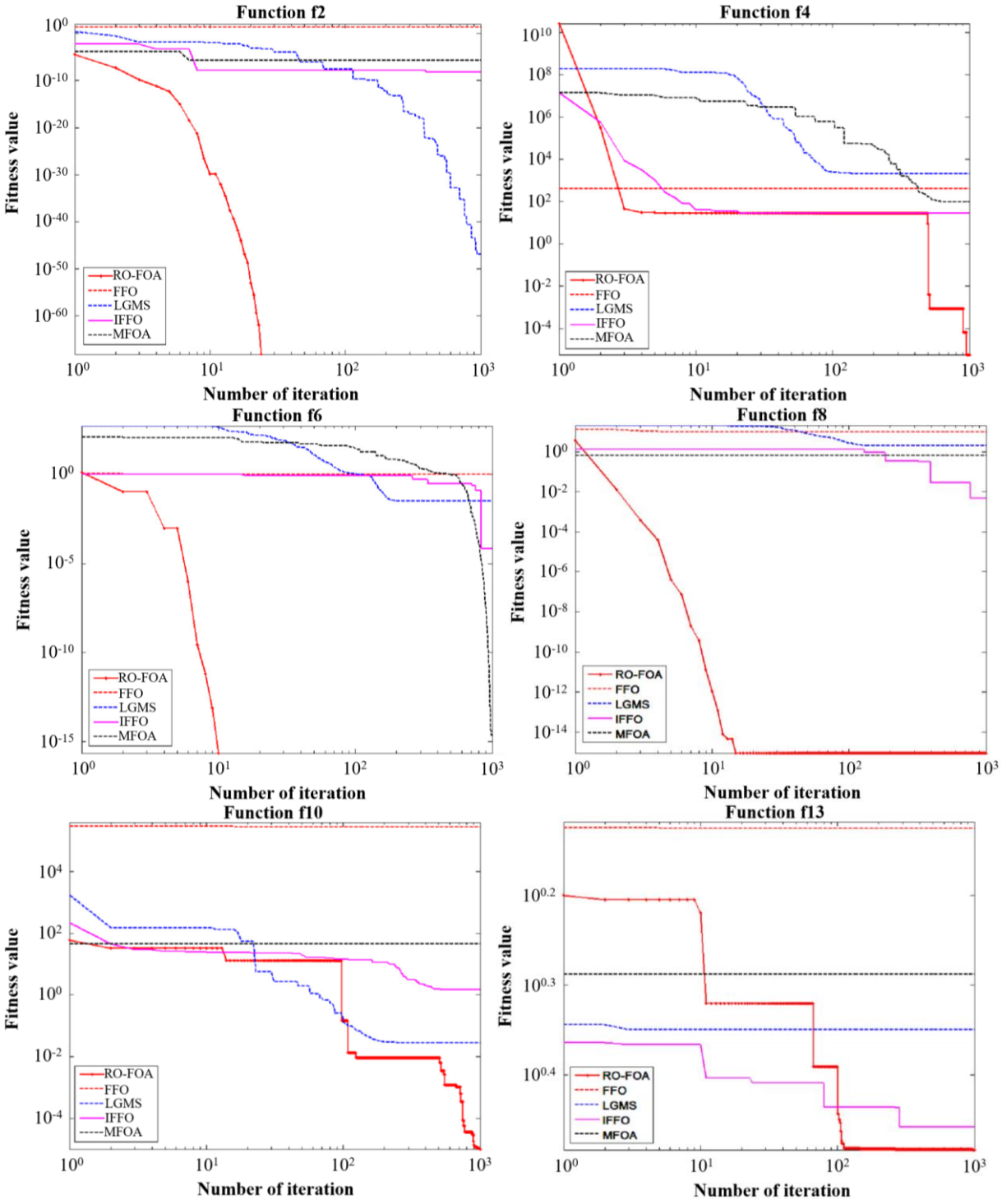


Figure 10 Convergence curve graphs that compare FFO, LGMS, IFFO, MFOA and RO-FOA

LGMS_FOA (5.77E+00), IFFO (2.45E-04), MFOA (4.48E-02), and MSFOA (1.70E+00) were out performed by the proposed RO-FOA, which obtained the best solution, with the optimum value of $f1 = 0$. The rank-sum test statistically confirmed this claim.

6.6 The solution quality comparisons of the RO-FOA and meta-heuristic algorithms

In order to better evaluate the performance of the proposed RO-FOA, we compared the final fitness values produced by seven widely used meta-heuristics algorithms, PSO, HS, GSA, DE, GWO, FPA, and MFPA (shown in Table 3). The 'winning' algorithms in any case are highlighted in bold. While the RO-FOA's final fitness value was not superior in all functions, the summary values of the rank-sum tests did find the RO-FOA superior in performance to all of the competing algorithms. The RO-FOA won 197 out of 238 cases. The rank-sum test statistically supports our claim.

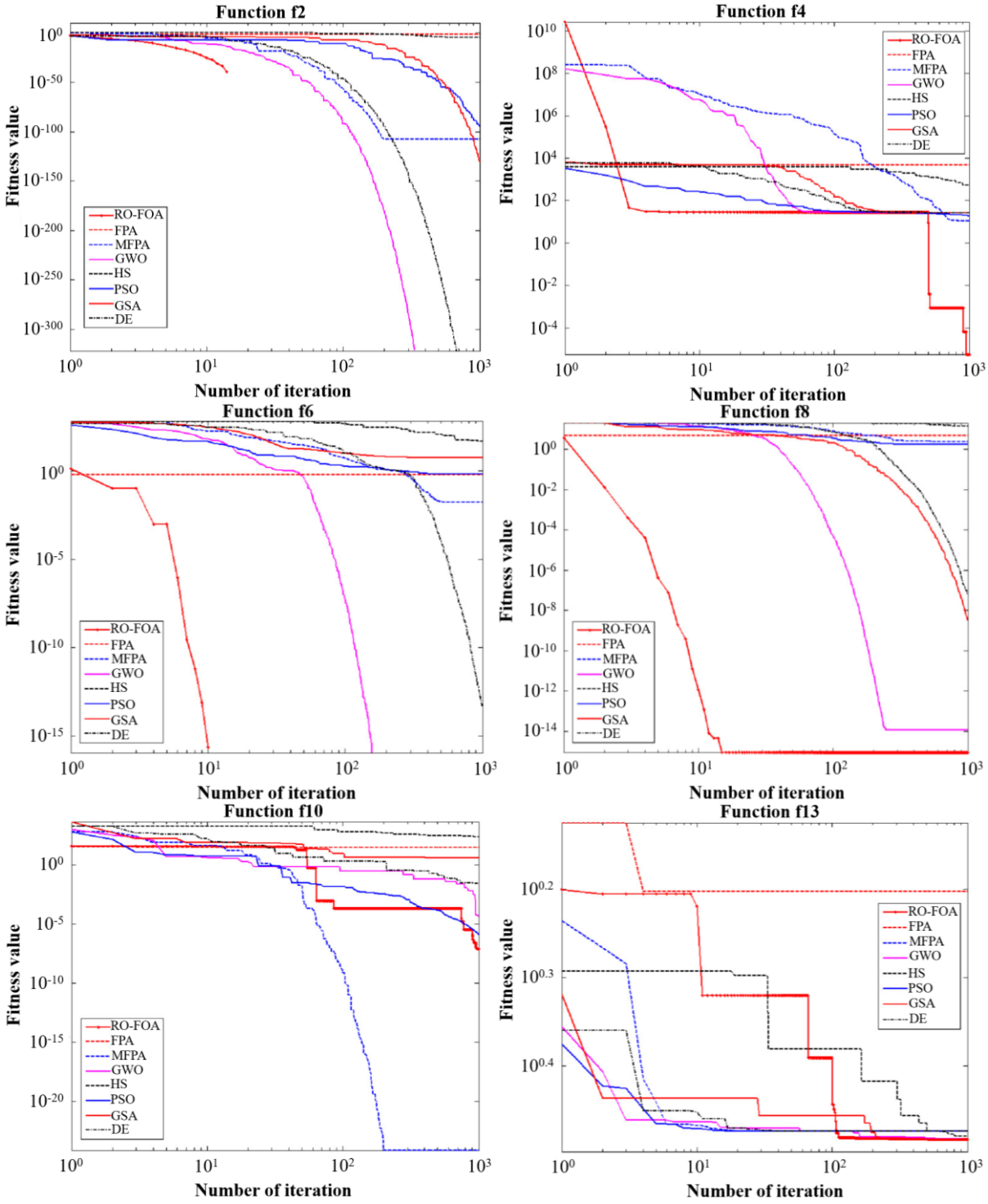


Figure 11 Convergence curve graphs that compare DE, GSA, PSO, HS, GWO, FPA, MFPA and RO-FOA.

6.7 Examples of the convergence graphs of the RO-FOA results and the comparative algorithms.

The advantage of incorporation of the opposition (OBL) and the random walk algorithms within the RO-FOA is that the new algorithm can quickly find the promising region of the search space, depending on the best solution index in any iteration. Unlike the variant FOAs, the boundary or radius in the search phase depends upon the convergence of the time interval when the radius must be changed, which is very time consuming. The RO-FOA renders a faster time in locating the best solution, compared to all of the competing algorithms (Figures 10 and 11). Graph histories are displayed in two sections. The first section, shown in Figure 10, depicts a convergence graph that compares the RO-FOA and variant FOAs. The second section, shown in Figure 11, depicts a convergence graph that compares the RO-FOA with seven meta-heuristic algorithms. The graph is presented in a log-log scale plot, where the X-axis is the number of iterations, and the Y-axis is the average fitness values obtained at that the corresponding iterations from the algorithms.

Table 4 Experimental results for the 5-bit and 8-bit auto-encoder datasets

Algorithm	5-bit auto-encoder			8-bit auto-encoder		
	MSE (AVE ± STD)	Classification Rate (%)	Mean of Weights' Magnitude	MSE (AVE ± STD)	Classification Rate (%)	Mean of Weights' Magnitude
RO-FOA	5.80e-01 ± 1.06e-02	**47.18**	2.772	6.22e-01 ± 7.59e-02	**24.11**	2.204
MFPA	7.02e-01 ± 1.46e-01	40.20	7.120	1.76e+00 ± 1.04e-01	21.70	7.465
FPA	**3.65e-01 ± 3.72e-02**	25.41	1.149	**2.56e-01 ± 2.27e-02**	20.58	1.131
MVO	6.44e-01 ± 1.39e-01	41.35	4.793	6.39e-01± 1.66e-01	20.15	4.784
GWO	5.98e-01 ± 9.14e-02	40.72	3.002	6.29e-01 ± 1.09e-01	18.48	2.693
PSO-GSA	7.68e-01 ± 1.49e-01	39.68	6.091	7.89e-01 ± 1.38e-01	17.96	6.432
FFO	1.57e+00 ± 1.89e-01	28.43	4.773	1.61e+00 ± 1.51e-01	16.87	4.894
LGMS	1.62e+00 ± 8.61e-02	24.06	4.813	1.68e+00 ± 1.22e-01	18.75	4.912
IFFO	6.62e-01 ± 1.32e-01	39.62	13.112	8.08e-01 ± 140e-01	22.34	11.738
MSFOA	1.84e+00 ± 2.65e-01	27.50	4.5611	2.05e+00 ± 1.79e-01	21.66	5.283

As can be seen in Figure 10, the RO-FOA, represented by a red line, has a faster convergence and is capable of finding a solution faster than the variant FOAs, in all of the example functions (*f*2, *f*4, *f*6, *f*8, *f*10 *and f*13*)*.The RO-FOA further demonstrates the ability to converge faster and find a better solution than the seven well-known meta-heuristic algorithms, DE, GSA, PSO, HS, GWO, FPA, and MFPA, as can be seen in Figure 11.

We conclude from the above results, that the proposed RO-FOA outperforms all variant FOAs and competitive meta-heuristic algorithms in finding solutions to unconstrained function optimization problems.

7. How effective is the RO-FOA in training auto-encoder multi-layer perceptrons?

The multi-layer perceptron [57] is one of most successful tools for expert systems or intelligent systems applications. This section employs the proposed RO-FOA for training Multi-Layer Perceptron (MLP) for the first time. The structure of the MLPs has been fixed, while all the weights and biases are simultaneously determined by the training algorithms to reduce the MLP's overall error. The RO-FOA training methodology begins with collection, normalization and feeding of the data set. Each layer of the network is ready for training if the network is structured for a specific application, including the required neuron number settings.

7.1 Settings

The sigmoid function is selected as the activation function of a MLP. The MLP structure is defined by the number of layers and number of neurons in the layer. The greater number of hidden layers and nodes, the more complex the network. The number of incoming and outgoing neurons in the MLP network depends on the problem. When using RO-FOA to optimize the weight and bias in a network with one hidden layer, a vector encoding method is utilized. Therefore, the dimension of each fruit fly, *D*, is computed using Eq. (11):

$$D = (I \times H) + (H \times C) + H_{bias} + C_{bias} \quad (11)$$

where I, H, and C refer to the number of input, hidden, and output neurons of a MLP, respectively. Also, H_{bias} and C_{bias} are the number of biases in the hidden and output layers, respectively.

Two datasets including 5-bit auto-encoder and 8-bit auto-encoder datasets are utilized to benchmark the performance of the proposed RO-FOA. The first dataset contains 32 patterns of 5-bit binary input/output data, the second dataset contains 256 patterns of 8-bit binary input/output data. As it is the auto-encoder problem, the input pattern and the output pattern are the same. For verification, the results are compared with Multi-Verse Optimizer (MVO) [58], MFPA, FPA, GWO, PSO-GSA, FOA, LGMS, IFFO, and MSFOA algorithms. For these two problems, the number of hidden nodes (H) is set to three. Therefore, the dimension of a fruit fly for 5-bit auto-encoder is 38, and the dimension of a fruit fly for 5-bit auto-encoder is 50.

For each algorithm, all individual solutions are evaluated for their qualities. This assessment is made by transmitting the weight vector, bias and input/output data to FNNs. The mean squared error (MSE) criterion is calculated based on the ability of the neural network. Using the training data set through continuous iteration, the fruit fly producing the lowest MSE is ultimately successful, which can be considered as the best weights and biases of neural networks. The MSE criterion is given in Eq. (12), where $\hat{y}$ and y are the actual and the estimated values based on proposed model, P is the number of samples in the training dataset and C is the number of output neurons.

$$MSE = \frac{1}{P}\sum_{i=1}^{P}\sum_{k=1}^{C}(y_{ik} - \hat{y}_{ik})^2 \quad (12)$$

The auto-encoder is a kind of classification problem. Therefore, the performance of the trained MLP can be evaluated as the classification rate, the higher is better. The classification rate is computed using Eq. (13):

$$\text{classification rate} = \frac{\sum_{i}^{P} predicted(i)==actual(i)}{P} \times 100 \quad (13)$$

where *predicted* is the predict value obtained from MLP and *actual* is the target value of each dataset. In the parameter settings, the maximum iteration number (*maxgen*) of each algorithm is fixed at 250, and the population size (*sizepop*) at 200. The weights and biases boundary is [-10, 10]. The mean value (*Mean*) and standard deviations (*Std*) over 30 independent runs are presented for each algorithm.

7.2. Training results

From Table 4, MLP trained by FPA produced the lowest MSE, and the MLP trained by RO-FOA was next best. Since, the MLP is a highly nonlinear models, its classification ability cannot be interpreted directly from MSE. The MLP trained by RO-FOA can produced the highest classification rates for both cases. That is because the magnitude of MLP's weights and biases, Eq. (11), for the auto-encoder problem should be small, but not be too small. FPA produced MLP with smaller magnitudes of weights and biases than RO-

FOA. We extended the experiment to cover eight standard datasets including five classification and three function-approximation datasets as in [59]. The conclusions were not changed. RO-FOA can find more proper weights and biases for MLP than the competitive algorithms in all kinds of problems tested. These results demonstrate a high level of accuracy in classification and approximation of the proposed RO-FOA trainer.

$$\text{Mean of Weights' Magnitude} = \frac{\sum_i^D |w_i|}{D} \quad (14)$$

where w is the weight vector of each MLP, D is the number of weights including the biases.

8. Conclusions

This paper presents random walk and opposition-based learning - fruit fly optimization algorithm, referred to as the RO-FOA algorithm. It borrows the knowledge from biological theory and mimics the dispersion patterns found in the organisms' survival tactics.

For the implementation, the RO-FOA algorithm uses both random walk and opposition-based learning algorithms. The two new ideas are put in the proposed OBL. They are (i) variation of shrinking probability versus iterations, and (ii) control of search transition from the exploration to the exploitation phase through the shrinking probability.

The main characteristics of RO-FOA in finding the optimal solution can be summarized as follows:

- The RO-FOA uses the RW algorithm as the main procedure.
- The RO-FOA uses the OBL algorithm to further enhance the main procedure.
- The RO-FOA generates three population patterns, uniform distribution, random distribution, and clumped distribution.
- Together with the three population distributions, the RO-FOA uses the individual exploration of RW and OBL to enhance the diversity of populations in a wide area while individually exploiting RW and OBL to enhance the search solution in a narrow area.
- Two types of population and dynamic distribution behaviors can help prevent the algorithm from becoming trapped at local optima whereas only one population behavior variant in the FOA can lead to that algorithm becoming trapped easily in local optima. The RO-FOA demonstrated its performance in solving unconstrained function optimization problems, through an enhanced search method, which adds diversity to its searching ability to find the best position.
- By a simple programming technique, for any value of *sizepop*, only three fruit flies are required.

We evaluated the RO-FOA's performance using 34 well-known standard benchmark functions. The simulation results clearly illustrated that the RO-FOA is capable of increasing the diversity of each individual fruit fly in any iteration, without becoming trapped at a local optima. Additionally, the RO-FOA outperformed both the original and variant FOAs in terms of convergence speed and performance accuracy in finding an optimal solution. The proposed method demonstrated enhanced exploration, exploitation, and solution accuracy in all functions.

From the training of two MLPs for 5-bit and 8-bit auto-encoder problems, the results demonstrate a high level of accuracy in classification of the proposed RO-FOA trainer.

When we compare RO-FOA with the well-known DE, GSA, PSO, HS, GWO, FPA, and MFPA algorithms, the mechanisms for generating new offspring are totally different. RO-FOA generates the offspring from the best solution only, but its search ability is good because of the cooperation of the random walk, opposition-based learning algorithms, and search radius scheduling. Each of the seven competitive algorithms generates new offspring using the whole solution vectors to create a good diversification in population.

In future work, we intend to apply the proposed RO-FOA approach in the machine learning related problems, such as multi-layer perceptron training, support vector machine training, and clustering algorithms. Furthermore, we found that the idea of generating the new offspring from a single best vector is very interesting because the resulting algorithm is very compact. This idea came from the era before swarm intelligent algorithms. However, we have compared RO-FOA using several random search algorithms with the benchmark functions in this paper and found that the algorithms of that kind hardly solved them. From our experience, an algorithm that generates new offspring from a single solution requires at least two clever components (i) a proper search radius scheduling, and (ii) a proper random walk generator. These two things must be further investigated.

9. References

[1] Kennedy J, Eberhart R. Particle swarm optimization. International Conference on Neural Networks; 1995 Nov 27 - Dec 1; Perth, Australia. USA: IEEE; 1995. p. 1942-8.

[2] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern. 1996;26(1):29-41.

[3] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Appl Math Comput. 2009; 214(1):108-32.

[4] Yang X, Suash D. Cuckoo Search via Lévy flights. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC); 2009 Dec 9-11; Coimbatore, India. USA: IEEE; 2009. p. 210-4.

[5] Yang XS. A new metaheuristic bat-inspired algorithm. In: González J, Pelta D, Cruz C, Terrazas G, Krasnogor N, editors. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010): Studies in Computational Intelligence. Berlin: Springer; 2010. p. 65-74.

[6] Behera S, Sahoo S, Pati BB. A review on optimization algorithms and application to wind energy integration to grid. Renew Sustain Energ Rev. 2015;48:214-27.

[7] Su Z, Wang H. A novel robust hybrid gravitational search algorithm for reusable launch vehicle approach and landing trajectory optimization. Neurocomputing. 2015;162:116-27.

[8] Kim H, Kang S. Communication-aware task scheduling and voltage selection for total energy minimization in a multiprocessor system using ant colony optimization. Inform Sci. 2011;181(18):3995-4008.

[9] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Comput Aided Des. 2011;43(3):303-15.

[10] Chen Y, Wong ML, Li H. Applying ant colony optimization to configuring stacking ensembles for data mining. Expert Syst Appl. 2014;41(6):2688-702.
[11] Mylonas SK, Stavrakoudis DG, Theocharis JB. GeneSIS: A GA-based fuzzy segmentation algorithm for remote sensing images. Knowl Base Syst. 2013;54: 86-102.
[12] Pan WT. A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl Base Syst. 2012;26:69-74.
[13] Pan WT. Using modified fruit fly optimisation algorithm to perform the function test and case studies. Connect Sci. 2013;25(2-3):151-60.
[14] Shan D, Cao G, Dong H. LGMS-FOA: an improved fruit fly optimization algorithm for solving optimization problems. Math Probl Eng. 2013;2013:ID 108768.
[15] Pan QK, Sang HY, Duan JH, Gao L. An improved fruit fly optimization algorithm for continuous function optimization problems. Knowl Base Syst. 2014;62:69-83.
[16] Yuan X, Dai X, Zhao J, He Q. On a novel multi-swarm fruit fly optimization algorithm and its application. Appl Math Comput. 2014;233:260-71.
[17] Ding S, Zhang X, Yu J. Twin support vector machines based on fruit fly optimization algorithm. Int. J. Mach. Learn. & Cyber. 2016;7:193–203.
[18] Li HZ, Guo S, Li CJ, Sun JQ. A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. Knowl Base Syst. 2013;37:378-87.
[19] Abidin ZZ, Arshad MR, Ngah UK. A simulation based fly optimization algorithm for swarms of mini autonomous surface vehicles application. Indian J Mar Sci. 2011;40:250-66.
[20] Lei X, Ding Y, Fujita H, Zhang A. Identification of dynamic protein complexes based on fruit fly optimization algorithm. Knowl Base Syst. 2016;105: 270-7.
[21] Cao G, Wu L. Support vector regression with fruit fly optimization algorithm for seasonal electricity consumption forecasting. Energy. 2016;115:734-45.
[22] Hu R, Wen S, Zeng Z, Huang T. A short-term power load forecasting model based on the generalized regression neural network with decreasing step fruit fly optimization algorithm. Neurocomputing. 2017;221: 24-31.
[23] Kanarachos S, Griffin J, Fitzpatrick ME. Efficient truss optimization using the contrast-based fruit fly optimization algorithm. Comput Struct. 2017; 182:137-48.
[24] Hongde D, Guorong Z, Jianhua L, Shaowu D. Comment and improvement on "A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example" Knowl Base Syst. 2014;59: 159-160
[25] Cupertino TH, Guimarães Carneiro M, Zheng Q, Zhang J, Zhao L. A scheme for high level data classification using random walk and network measures. Expert Syst Appl. 2018;92:289-303.
[26] Faragallah OS, Abdel-Aziz G, Kelash HM. Efficient cardiac segmentation using random walk with pre-computation and intensity prior model. Appl Soft Comput. 2017;61:427-46.
[27] Lang J, Prehl J. An embarrassingly parallel algorithm for random walk simulations on random fractal structures. J Comput Sci. 2017;19:1-10.
[28] Li W, Xie J, Xin M, Mo J. An overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk. Expert Syst Appl. 2018;91:277-85.
[29] Rahnamayan S, Tizhoosh HR, Salama MMA. A novel population initialization method for accelerating evolutionary algorithms. Comput Math Appl. 2007;53(10):1605-14.
[30] Rahnamayan S, Tizhoosh HR, Salama MMA. Opposition-based differential evolution. evolutionary computation, IEEE Trans Evol Comput. 2008;12(1):64-79.
[31] Niknamfar AH, Niaki STA, Niaki SAA. Opposition-based learning for competitive hub location: a bi-objective biogeography-based optimization algorithm. Knowl Base Syst. 2017;128:1-19.
[32] Sharma TK, Pant M. Opposition based learning ingrained shuffled frog-leaping algorithm. J Comput Sci. 2017;21:307-15.
[33] Ahandani MA. Opposition-based learning in the shuffled bidirectional differential evolution algorithm. Swarm Evol Comput. 2016;26:64-85.
[34] Chen X, Yu K, Du W, Zhao W, Liu G. Parameters identification of solar cell models using generalized oppositional teaching learning based optimization. Energy. 2016;99:170-80.
[35] Odum EP. Basic Ecology. Philadelphia: Saunders College Publishing; 1983.
[36] Lianghong W, Cili Z, Hongqiang Z. A cloud model based fruit fly optimization algorithm. Knowl Base Syst. 2015;89: 603-617.
[37] Marko M, Najdan V, Milica P, Zoran M. Chaotic fruit fly optimization algorithm. Knowl Base Syst. 2015;89: 446-458.
[38] Jinwei N, Cili Z, Weimin Z, Yi L, Na L, Feng Q. Fruit fly optimization algorithm based on differential evolution and its application on gasification process operation optimization. Knowl Base Syst. 2015;88: 253-263.
[39] Zhang Y, Cui G, Wu J, Pan WT, He Q. A novel multi-scale cooperative mutation fruit fly optimization algorithm. Knowl Base Syst. 2016;114:24-35.
[40] Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm – a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct. 2012;110-111:151-66.
[41] Yu JJQ, Li VOK. A social spider algorithm for global optimization. Appl Soft Comput. 2015;30:614-27.
[42] Mirjalili S. The ant lion optimizer. Adv Eng Software. 2015;83:80-98.
[43] Tizhoosh HR. Opposition-based learning: a new scheme for machine intelligence. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce; 2005 Nov 28-30; Vienna, Austria. USA: IEEE; 2005. p. 695-701.
[44] Xu Q, Wang L, Wang N, Hei X, Zhao L. A review of opposition-based learning from 2005 to 2012. Eng Appl Artif Intell. 2014;29:1-12.
[45] Yao X, Liu Y, Lin G. Evolutionary programming made faster. IEEE Trans Evol Comput. 1999;3(2):82-102.
[46] Beyer HG, Schwefel HP, Wegener I. How to analyse evolutionary algorithms. Theor Comput Sci. 2002; 287(1):101-30.

[47] Yelghi A, Köse C. A modified firefly algorithm for global minimum optimization. Appl Soft Comput. 2018;62:29-44.
[48] Wang CF, Song WX. A novel firefly algorithm based on gender difference and its convergence. Appl Soft Comput. 2019;80:107-24.
[49] Shi Y, Eberhart RC. Empirical study of particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99; 1999 Jul 6-9; Washington, USA. USA: IEEE; 1999. p. 1945-50.
[50] Vesterstrom J, Thomsen R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Proceedings of the 2004 Congress on Evolutionary Computation; 2004 Jun 19-23; Portland, USA. USA: IEEE; 2004. p. 1980-7.
[51] Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Meth Appl Mech Eng. 2005;194(36-38):3902-33.
[52] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm. Inform Sci. 2009;179(13): 2232-48.
[53] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Software. 2014;69:46-61.
[54] Yang XS. Flower pollination algorithm for global optimization. In: Durand-Lose J, Jonoska N, editors. International Conference on Unconventional Computing and Natural Computation; 2012 Sep 3-7; Orléan, France. Berlin: Springer; 2012. p. 240-9.
[55] Nabil E. A modified flower pollination algorithm for global optimization. Expert Syst Appl. 2016;57:192-203.
[56] Hrstka O, Kučerová A. Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. Adv Eng Software. 2004;35(3-4):237-46.
[57] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Network. 1989;2(5):359-66.
[58] Faris H, Aljarah I, Mirjalili S. Training feedforward neural networks using multi-verse optimizer for binary classification problems. Appl Intell. 2016;45:322-32.
[59] Mirjalili S. How effective is the grey wolf optimizer in training multi-layer perceptrons. Appl Intell. 2015; 43:150-61.

Appendix A

Descriptions of test problems

1. Sphere model

$$f_1(x) = \sum_{i=1}^{n} x_i^2, with -5.12 \le x_i \le 5.12$$

$$\min(f_1) = f_1(0,...,0) = 0.$$

2. Axis parallel hyperellipsoid

$$f_2(x) = \sum_{i=1}^{n} i x_i^2, with -5.12 \le x_i \le 5.12$$

$$\min(f_2) = f_2(0,...,0) = 0.$$

3. Schwefel problem 1.2

$$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2, with -65 \le x_i \le 65$$

$$\min(f_3) = f_3(0,...,0) = 0.$$

4. Rosenbrock's valley

$$f_4(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2 \right], with -2 \le x_i \le 2$$

$$\min(f_4) = f_4(1,...,1) = 0.$$

5. Rastrigin function

$$f_5(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) \right), with -5.12 \le x_i \le 5.12$$

$$\min(f_5) = f_5(0,...,0) = 0.$$

6. Griewangk function

$$f_6(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1, with -600 \le x_i \le 600$$

$$\min(f_6) = f_6(0,...,0) = 0.$$

7. Sum of different power

$$f_7(x) = \sum_{i=1}^{n} |x_i|^{i+1}, with -1 \le x_i \le 1$$

$$\min(f_7) = f_7(0,...,0) = 0.$$

8. Ackley's path function

$$f_8(x) = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=1}^{n}\cos(2\pi x_i)}{n}\right) + 20 + e, with -32 \le x_i \le 32$$

$$\min(f_8) = f_8(0,...,0) = 0.$$

9. Beale function

$$f_9(x) = \left[1.5 - x_1(1-x_2)\right]^2 + \left[2.25 - x_1\left(1-x_2^2\right)\right]^2 + \left[2.625 - x_1\left(1-x_2^3\right)\right]^2, with -4.5 \le x_i \le 4.5$$

$$\min(f_9) = f_9(3,0.5) = 0.$$

10. Colville function

$$f_{10}(x) = 100\left(x_2 - x_1^2\right)^2 + (1-x_1)^2 + 90\left(x_4 - x_3^2\right)^2 + (1-x_3)^2 + 10.1\left((x_2-1)^2 + (x_4-1)^2\right) + 19.8(x_2-1)(x_4-1),$$

$$with -10 \le x_i \le 10$$

$$\min(f_{10}) = f_{10}(1,1,1,1) = 0.$$

11. Easom function

$$f_{11}(x) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2 - (x_2-\pi)^2\right), with -100 \le x_i \le 100$$

$$\min(f_{11}) = f_{11}(\pi,\pi) = -1.$$

12. Hartman function 1

$$f_{12}(x) = -\sum_{i=1}^{4}\alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}\left(x_j - P_{ij}\right)^2\right), \quad with\ 0 \le x_i \le 1$$

$$\min(f_{12}) = f_{12}(0.114614, 0.555649, 0.852547) = -3.86278.$$

where $\alpha = [1 \quad 1.2 \quad 3 \quad 3.2]$

$$A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.36890 & 0.11700 & 0.26730 \\ 0.46990 & 0.43870 & 0.74700 \\ 0.10910 & 0.87320 & 0.55470 \\ 0.03815 & 0.57430 & 0.88280 \end{bmatrix}$$

13. Hartman function 2

$$f_{13}(x) = -\sum_{i=1}^{4}\alpha_i \exp\left(-\sum_{j=1}^{6} B_{ij}\left(x_j - Q_{ij}\right)^2\right), \quad with\ 0 \le x_i \le 1$$

$$\min\left(f_{13}\right) = f_{13}(0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573) = -3.32237$$

where $\alpha = [1 \quad 1.2 \quad 3 \quad 3.2]$

$$B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$Q=\begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

14. Six Hump Camel back function

$$f_{14}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4,\ with\ -5\le x_i\le 5$$

$$\min(f_{14})=f_{14}(0.0898,-0.7126)/(-0.0898,0.7126)=-1.0316.$$

15. Levy function

$$f_{15}(x)=\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\times\left(1+\sin^2(3\pi x_{i+1})\right)+(x_n-1)\left(1+\sin^2(2\pi x_n)\right),\ with -10\le x_i\le 10$$

$$\min(f_{15})=f_{15}(1,...,1)=0.$$

16. Matyas function

$$f_{16}(x)=0.26\left(x_1^2+x_2^2\right)-0.48x_1x_2,\ with -10\le x_i\le 10$$

$$\min(f_{16})=f_{16}(0,0)=0.$$

17. Perm function

$$f_{17}(x)=\sum_{k=1}^{n}\left[\sum_{i=1}^{n}\left(i^k+0.5\right)\left(\left(\frac{1}{i}x_i\right)^k-1\right)\right]^2,\ with -n\le x_i\le n$$

$$\min(f_{17})=f_{17}(1,2,3,...,n)=0.$$

18. Michalewicz function

$$f_{18}(x)=-\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(ix_i^2/\pi\right)\right)^{2m},\ with\ 0\le x_i\le\pi, m=10$$

$$\min(f_{18(n=2)})=-1.8013,$$

$$\min(f_{18(n=5)})=-4.687658,$$

$$\min(f_{18(n=10)})=-9.66015.$$

19. Zakharov function

$$f_{19}(x)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4,\ with -5\le x_i\le 10$$

$$\min(f_{19})=f_{19}(0,...,0)=0.$$

20. Branin function

$$f_{20}(x)=\alpha\left(x_2-bx_1^2+cx_1-d\right)^2+e(1-f)\cos(x_1)+e,\ with -5\le x_i\le 10,\ \ 0\le x_2\le 15$$

$$where\ \alpha=1,\ b=5.1/\left(4\pi^2\right),\ c=5/\pi,\ d=6,\ e=10,\ f=1/(8\pi)$$

$$\min(f_{20})=f_{20}(-\pi,12.275)/(9.42478,\ 2.475)=0.3979.$$

21. Schwefel problem 2.22

$$f_{21}(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|,\ with -10\le x_i$$

$$\min(f_{21})=f_{21}(0,...,0)=0.$$

22. Schwefel problem 2.21

$$f_{22}(x)=\max_i\{|x_i|,1\le i\le n\},\ with -100\le x_i\le 100$$

$$\min(f_{22})=f_{22}(0,...,0)=0.$$

23. Step function

$$f_{23}(x)=\sum_{i=1}^{n}\left(\lfloor x_i+0.5\rfloor\right)^2,\ with -100\le x_i\le 100$$

$$\min(f_{23})=f_{23}(-0.5\le x_i\le 0.5)=0.$$

24. Quartic function

$$f_{24}(x)=\sum_{i=1}^{n} ix_i^4 + random[0,1),\ with -1.28 \le x_i \le 1.28$$
$$\min(f_{24}) = f_{24}(0,...,0) = 0.$$

25. Kowalik's function

$$f_{25}(x)=\sum_{i=1}^{11}\left[\alpha_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2,\ with -5 \le x_i \le 5$$
$$\min(f_{25}) = f_{25}(0.19,0.19,0.12,0.14) = 0.0003075$$
where
$$\alpha = [0.1957 \quad 0.1947 \quad 0.1735 \quad 0.1600 \quad 0.0844 \quad 0.0627 \quad 0.0456 \quad 0.0342 \quad 0.0323 \quad 0.0235 \quad 0.0246]$$
$$b^{-1} = [0.25 \quad 0.5 \quad 1 \quad 2 \quad 4 \quad 6 \quad 8 \quad 10 \quad 12 \quad 14 \quad 16]$$

26-28. Shekel's Family

$$f(x) = -\sum_{i=1}^{m}\left[(x_i-\alpha_i)(x_i-\alpha_i)^T + c_i\right]^{-1},\ with\ m = 5,7,and 10$$
for $f_{26}(x), f_{27}(x), and\ f_{28}(x), respectivdy,\ 0 \le x_j \le 10,$
$$\min(f_{26}) = f_{26}(4,4,4,4) = -10.2,$$
$$\min(f_{27}) = f_{27}(4,4,4,4) = -10.4,$$
$$\min(f_{28}) = f_{28}(4,4,4,4) = -10.5.$$
where

$$\alpha = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}, \quad c = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}.$$

29. Tripod function

$$f_{29}(x) = p(x_2)(1+p(x_1)) + |(x_1 + 50\,p(x_2)(1-2\,p(x_1)))| + |(x_2 + 50(1-2\,p(x_2)))|,\ with -100 \le x_i \le 100$$
$$\min(f_{29}) = f_{29}(0,-50) = 0$$
where
$$p(x) = 1\ for\ x \ge 0,\ otherwise,\ p(x) = 0.$$

30. De Jong function 4 (no noise)

$$f_{30}(x)=\sum_{i=1}^{n} ix_i^4,\ with -1.28 \le x_i \le 1.28$$
$$\min(f_{30}) = f_{30}(0,...,0) = 0.$$

31. Alpine function

$$f_{31}(x)=\sum_{i=1}^{n}\left|x_i \sin(x_i) + 0.1x_i\right|,\ with -10 \le x_i \le 10$$
$$\min(f_{31}) = f_{31}(0,...,0) = 0.$$

32. Schaffer function 6

$$f_{32}(x) = 0.5 + \frac{\sin^2\sqrt{\left(x_i^2 + x_2^2\right) - 0.5}}{1 + 0.01\left(x_i^2 + x_2^2\right)^2},\ with -10 \le x_i \le 10$$

$$\min\left(f_{32}\right) = f_{32}(0,0) = 0.$$

33. Pathological function

$$f_{33}(x) = \sum_{i=1}^{n-1}\left(0.5 + \frac{\sin^2\sqrt{\left(100x_i^2 + x_{i+1}^2\right) - 0.5}}{1 + 0.001\left(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2\right)^2}\right),\ with -100 \le x_i \le 100$$

$$\min\left(f_{33}\right) = f_{33}(0,...,0) = 0.$$

34. Inverted cosine wave function (Masters)

$$f_{34}(x) = -\sum_{i=1}^{n-1}\left(\exp\left(\frac{-\left(x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}\right)}{8}\right) \times \cos\left(4\sqrt{x_i^2 + x_{i+1}^2} + 0.5x_i x_{i+1}\right)\right),\ with -5 \le x_i \le 5$$

$$\min\left(f_{34}\right) = f_{34}(0,...,0) = -n + 1.$$

Appendix B

The initial parameters of algorithms

Algorithm	Parameter	Value
RO-FOA	Population size	50 for function optimization, 200 for MLP
	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
FFO	Population size	50 for function optimization, 200 for MLP
[60]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
LGMS	Population size	50 for function optimization, 200 for MLP
[14]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The searching coefficient (*n*)	0.005
	The initial weight (w_0)	1
	The weight coefficient (α)	0.95
IFFO	Population size	50 for function optimization, 200 for MLP
[15]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The maximum radius (γ_{max})	$UB - LB/2$
	The minimum radius ($\gamma_{\min}$)	10^{-5}
MFOA	Population size	50 for function optimization, 200 for MLP
[16]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The sub-swarms number (M)	5
	The fine-tuning of solution vectors (θ)	2
MSFOA	Population size	50 for function optimization, 200 for MLP
[39]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The search coefficient (*n*)	0.005
	The initial weight (w_0)	1
	The weight coefficient (α)	0.95
	The scale number (*M*)	5
PSO	Population size	50 for function optimization, 200 for MLP
[49]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The particle's confidence (c1, c2)	2, 1.5
	The inertia weight (w_{max} , w_{min})	1.3, 0.3
	The velocity parameter (v_{max})	limited to 20% of the domain
DE	Population size	50 for function optimization, 200 for MLP
[61]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The scaling factor (F)	0.5
	The crossover probability constant (CR)	0.9
HS	Population size	50 for function optimization, 200 for MLP
[51]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The harmony memory size (HMS)	*sizepop*
	The harmony memory consideration rate (HMCR)	0.90
	The pitch adjusting rate (PAR)	0.35
GSA	Population size	50 for function optimization, 200 for MLP
[52]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The initial gravitational constant (G_0)	1
	The constant descending coefficient (α)	1
	The constant (β)	10
GWO	Population size	50 for function optimization, 200 for MLP
[53]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
FPA	Population size	50 for function optimization, 200 for MLP
[54]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The switch probability (P)	0.8
	The step size scaling factor (γ)	0.01
	The Levy flight (λ)	1.5
MFPA	Population size	50 for function optimization, 200 for MLP
[55]	Maximum iteration number (*maxgen*)	1000 for function optimization, 250 for MLP
	The switch probability (P)	0.8
	The step size scaling factor (γ_1, γ_2)	1, 3
	The Levy flight (λ)	1.5
	The cloning array	[9, 8, 7, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1]