An Energy-segmented Moth-flame Optimization Algorithm for Function Optimization and Performance Measures Analysis

YUANFEI WEI¹, PENGCHUAN WANG²,³, QIFANG LUO²,³*, YONGQUAN ZHOU²,³

¹Xiangsihu College of Guangxi University for Nationalities, Nanning, Guangxi 532100, CHINA
²College of Information Science and Engineering, Guangxi University for Nationalities, Nanning, Guangxi 530006, CHINA
³Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, CHINA

Abstract: The moth-flame optimization algorithm (MFO) is a novel metaheuristic algorithm for simulating the lateral positioning and navigation mechanism of moths in nature, and it has been successfully applied to various optimization problems. This paper segments the flame energy of MFO by introducing the energy factor from the Harris hawks optimization algorithm, and different updating methods are adopted for moths with different flame-detection abilities to enhance the exploration ability of MFO. A new energy-segmented moth-flame optimization algorithm (ESMFO) is proposed and is applied on 21 benchmark functions and an engineering design problem. The experimental results show that the ESMFO yields very promising results due to its enhanced exploration, exploitation, and convergence capabilities, as well as its effective avoidance of local optima, and achieves better performance than other the state-of-the-art metaheuristic algorithms in terms of the performance measures.

KeyWords: moth-flame optimization algorithm, energy-segmented moth-flame optimization, benchmark function, metaheuristic optimization

1. Introduction
Over the last three decades, some classical methods for solving various optimization problems have been successfully established. As the number of optimization problem parameters increases, the complexity of these problems increases. Therefore, when dealing with high-dimensional problems, classical methods cannot obtain accurate global minima or maxima, and so they fall into local minima. Thus, finding the exact solutions of such problems becomes a challenge for classical methods [1]. To solve this problem of classical optimization algorithms, a natural heuristic algorithm is defined under the same background. In fact, for millions of years, nature has been a great source of inspiration for the solving of real-world problems by human beings. There are many real-world processes that describe nature-inspired computations, such as those related to decision-making, immune systems, collective behaviors, and learning. Based on these phenomena, various natural heuristic optimization algorithms have been designed and are now widely applied in most research fields.

According to the inspiration sources of these algorithms, natural heuristic algorithms can be divided into four categories: evolution-based, group-based, physics-based and human-based approaches [2][3]. The evolutionary methods were developed based on the law of natural evolution. Among these methods, the most the genetic algorithm which simulates Darwin's evolutionary process, is undoubtedly the most popular [4]. In this category, other popular methods include evolutionary strategies [5] and genetic programming [6]. On the other hand, group-based approaches are designed to simulate the social and collective behaviors of animal groups (birds, insects, fish, etc.). The particle swarm optimization [7][8] algorithm, which was inspired by the social behaviors of flocking birds, is the most representative and successful example in this field. Other related methods include the ant colony optimization [9][10], artificial bee colony [11][12][46], and whale optimization algorithms [13][14], etc. In addition, there are some physics-based algorithms that have been developed on the basis of simulating the laws of physics observed in our universe. The most popular methods in this category are simulated
randomly belongs to the judgment statement. If it, is returned; and, is the principal function of the moth search (1)

The moth-flame optimization (MFO) algorithm [22-24] is a
an evolutionary algorithm that was recently proposed. The
algorithm is inspired by the navigation behavior or lateral
orientation of moths found in nature. According to their
lateral orientation, moths fly straight toward the moon by
forming a fixed angle with the moon. This phenomenon only
applies to light sources far away from moths. Therefore, any
artificial light in their path causes them to follow the light,
and they persist around it as they continue to move in a
deadly spiral path. This phenomenon was used to develop
the MFO algorithm, and since its inception, it has been applied to
solve various real-world tasks.

For any generalized algorithm, exploration and
development are its most important features. Here,
exploration refers to exploring the search space or the use of
global search, while development refers to local search.
Although MFO is a good algorithm, because there is no
perfect theorem [25], no algorithm is the most suitable one
for all optimization problems. Therefore, in order to improve
the performance of MFO, some improvement schemes are
proposed to improve the development and exploration
capabilities of the MFO algorithm and maintain a proper
balance between the two. Here, the energy classification
system of the Harris hawks optimization (HHO) algorithm
[26] is introduced to segment the flame energy in the basic
MFO algorithm and balance the exploration and exploitation
abilities of the algorithm. The new algorithm is called
energy-segmented moth-flame optimization (ESMFO),
which focuses on improving the search ability
and convergence speed of the MFO algorithm. In this paper, we
apply the ESMFO algorithm to the tasks of function
optimization and performance measures analysis.

In this paper, by introducing the energy factor from the
Harris hawks optimization algorithm, the flame energy in
MFO is segmented, and different updating methods are
adopted for moths of different flame-detection abilities
to enhance the exploration ability of MFO. The ESMFO is
applied to test 21 benchmark functions. The experimental
results show that among the compared algorithms, the
ESMFO algorithm has the best comprehensive performance
and achieves better performance than other the state-of-the-
art metaheuristic algorithms in terms of the chosen
performance measures.

The rest of the paper is organized as follows. Section II
provides a description of related works. Section III describes
the proposed energy-segmented moth-flame optimization
(ESMFO) algorithm. The results are discussed in Section IV.
Section V introduces the conclusions and future work ideas.

2. Related Work
2.1. Original Moth-flame Optimization Algorithm (MFO)

Moths are winged insects belonging to the butterfly family.
At night, moths fly by moonlight. For the purpose of flying,
they use a special navigation mechanism known as lateral
positioning. According to this method, moths maintain a
fixed angle with the moon when flying. However, moths
have been observed to fly around light in a spiral manner
rather than horizontally. This is because lateral positioning is
effective only when the light source is far away from the
moth. Because the alternate light source is far from the moon,
the straight path becomes a spiral path.

In the basic moth optimization algorithm, the position of a
moth in space is the variable of the problem (assuming that
the moth is the candidate solution). By changing the moth's
position vector, the algorithm can solve problems in low-
dimensional and multidimensional space. In addition, it is
worth noting that the moth and flame are both candidates in
the moth optimization algorithm. The moth is the main
moving body in the solution space, and the flame is the
optimal value obtained by the moth during the current
iteration of the algorithm. The flame can be understood as
the location marker at the end of the moth's final search path.
When the algorithm is locally exploited, moths search for the
optimal position in the field of flame. Under this mechanism,
the optimization accuracies of moths are improved.

The framework of the moth optimization algorithm is
as follows:

\[ MFO = (I, P, T) \]  \hspace{1cm} (1)

The function \( I \) generates a moth population \( M \) randomly
and generates the corresponding fitness matrix \( OM \).

\[ I : \phi \rightarrow \{ M, OM \} \]  \hspace{1cm} (2)

The function \( P \) is the principal function of the moth search
process in the solution space. After updating, the matrix
\( M \) is returned:

\[ P : M \rightarrow M \]  \hspace{1cm} (3)

The function \( T \) belongs to the judgment statement. If it
satisfies the termination condition, “true” is returned; if not,
“false” is returned:

\[ T : M \rightarrow \{ true, false \} \]  \hspace{1cm} (4)

As mentioned above, the algorithm is inspired by lateral
positioning. We use the following equation to update the
moth's position:

\[ M_j = S(M_i, F_j) \]  \hspace{1cm} (5)

Here, \( M_i \) denotes moth \( i \), \( F_j \) denotes flame \( j \), and \( S \) is
a spiral function.

In the moth optimization algorithm, the main updating
mechanism for a moth is simulated by a logarithmic helix
function, and the following conditions must be satisfied:
- The starting point of the helix is at the position of the moth.
- The end of the helix is at the position of the flame.
- The floating range of the helix does not exceed the search space.

Based on the above conditions, the logarithmic helix function of the moth optimization algorithm is defined as follows:

\[ S(M_i, F_j) = D_i \cdot e^{b \cdot \cos(2\pi t)} + F_j \]  

Here, the distance between the \( i \)-th moth and the \( j \)-th flame is denoted by \( D_i \). \( b \) is a constant that defines the logarithmic helix and \( t \) is a random number between \([-1,1]\). \( D_i \) is calculated by the following formula:

\[ D_i = |F_j - M_i| \]  

In the whole search space, the position of a moth is updated relative to \( n \) different positions, resulting in a decrease in the search accuracy and an increased likelihood of falling into a local optimum. Therefore, an adaptive mechanism based on the number of flames is proposed. The formula is as follows:

\[ \text{flame}_\text{no} = \text{round}\left( N - I \star \frac{N-1}{T} \right) \]  

Here, \( I \) is the current iteration number, \( N \) is the maximum number of iterations, and \( T \) is the maximum number of iterations.

### 2.2. Harris Hawks Optimization (HHO)

The main inspiration for HHO comes from the cooperative behavior and pursuit strategy of the Harris eagle, and this strategy is called a "raid" in nature [27-30]. This desert predator demonstrates the innovative evolutionary ability of teaming to track, surround, rush at and eventually attack potential prey. Harris hawks' main strategy for catching prey is called assault, also known as the "seven killing strategy." In this clever strategy, several eagles converge on their prey from different directions in an attempt to perform a surprise attack. Harris hawks can reveal many kinds of chase patterns according to the dynamic characteristics of the scene and the chosen escape mode of their prey. In this section, inspired by exploring the different attacking and raiding strategies of Harris hawks, we simulate the exploration and exploitation stages of the proposed HHO. HHO is a population-based, gradient-free optimization technique; therefore, it can be applied to any optimization problem as long as there is an appropriate formula. HHO algorithms are mainly divided into three stages: the exploratory stage, the transitional stage between the exploratory mining and exploitation stages, and the exploitation stage. Since the energy factor of our improved algorithm comes from the latter two stages of the Harris hawks optimization algorithm, we briefly introduce the latter two stages of the Harris hawks optimization algorithm.

#### (1) Transition from exploration to exploitation

The HHO algorithm can transfer from the exploration phase to the exploitation phase and then change between different exploitative behaviors based on the escaping energy of the prey. The energy of a prey decreases considerably during the escaping behavior. To model this fact, the energy of a prey is modeled as:

\[ E = 2E_0 \left(1 - \frac{t}{T}\right) \]  

where \( E \) indicates the escaping energy of the prey, \( T \) is the maximum number of iterations, and \( E_0 \) is the initial state of the energy of the prey. In HHO, \( E_0 \) randomly changes within the interval \((-1,1)\) during each iteration. When the value of \( E_0 \) decreases from 0 to -1, the prey is physically flagging, but when the value of \( E_0 \) increases from 0 to 1, the prey is strengthening. This dynamic escaping energy exhibits a decreasing trend as the number of iterations increases. In short, exploration occurs when \( |E| \geq 1 \), while exploitation occurs in later steps when \( |E| < 1 \). For defining the threshold of \( E_0 \), we can refer to the original paper [26] for a more detailed introduction than the one given here.

#### (2) Exploitation phase

In this phase, the Harris hawks perform a surprise attack by attacking the prey detected in the previous phase. However, prey often attempt to escape from dangerous situations. According to the escape behaviors of the prey and the chasing strategies of the Harris hawks, four possible strategies were proposed in the HHO.

The authors proposed a probability parameter \( r \) for the successful escape of prey from pursuit. When \( r < 0.5 \), the escape is successful, and when \( r > 0.5 \), the prey does not successfully escape. To model this strategy and enable the HHO algorithm to switch between the soft and hard sieging processes, the parameter \( E \) is utilized. In this regard, when \( |E| > 0.5 \), the soft attack occurs, and when \( |E| < 0.5 \), the hard attack occurs. Since this article quotes the third form of sieging described by the original author at this stage (hard sieging with progressive, rapid dives), we only introduce this kind of sieging strategy. For more detailed information, please read the source article about HHO.

When \( |E| < 0.5 \) and \( r < 0.5 \), the prey does not have enough energy to escape and a hard siege is conducted before the surprise pounce to catch and kill the prey. Therefore, the following rule is obeyed by a hard siege:

\[ X(t+1) = \begin{cases} Y & \text{if } F(Y) < F(X(t)) \\ Z & \text{if } F(Z) < F(X(t)) \end{cases} \]  

where \( Y \) and \( Z \) are obtained using the new rules in Eqs. (12) and (13).

\[ Y = X_{\text{rabbit}}(t) - E|JX_{\text{rabbit}}(t) - X_{\text{avg}}(t)| \]  

where \( X_{\text{avg}}(t) \) is the average position of the hawks. Then, the possible result of such a movement is compared to that of
the previous dive to detect whether it be a good dive. If the result is not reasonable, they also start to perform irregular, abrupt, and rapid dives when approaching the prey. We suppose that they dive according to LF-based patterns using the following rule:

\[ Z = Y + S \times LF(D) \]  \hspace{1cm} (12)

where \( D \) is the dimension of the problem, \( S \) is a random vector of size \( 1 \times D \) and \( LF \) is the Levy flight function, which is calculated using Eq. (13) \([48]\):

\[
LF(x) = 0.01 \times \frac{u \times \sigma}{\sqrt{\pi^2}} \left( \frac{1 + \beta}{2} \times \beta \times 2^{-\beta} \right)^{\frac{1}{\beta}}
\]  \hspace{1cm} (13)

where \( u \) and \( v \) are random values bounded by (0,1) and \( \beta \) is a constant set to 1.5 by default.

### 3. Energy-segmented Moth-flame Optimization Algorithm

As we all know, the authors who developed the MFO algorithm only considered the update process for a moth in the ideal state and did not consider the influence of the distance between the moth and the light source on the moth when the it circles the artificial light source. Therefore, in order to better conform to the biological characteristics of nature, the algorithm proposed in this paper accounts for the influence of the brightness of the light source on moth flight. According to the previous section, we compare moths to Harris hawks, and then we compare flames to prey. Considering the moth's irregular flight and the effect of light intensity on the moth, we borrow the adaptive parameter for the energy factor from the HHO algorithm. This parameter is called the light source intensity \( E \).

The light source intensity \( E \) is used to distinguish between the two stages (i.e., exploration and exploitation) of the MFO algorithm; when the moth is far away from the flame, the moth feels a low amount of flame energy, and the ESMFO algorithm performs the exploration phase; otherwise, the moth and flame are close, the flame energy felt by the moth is high, and the algorithm performs the exploitation phase. \( E \) exhibits a rising trend as the number of iterations increases. When the flame energy \(|E|<0.5\), the moth searches different regions to detect the position of the flame, so the ESMFO algorithm performs the exploration phase. When \(|E|\geq0.5\), the algorithm attempts to exploit the neighborhood of the solutions during the exploitation phase. In short, exploration occurs when \(|E|<0.5\), while exploitation occurs in later steps when \(|E|\geq0.5\). The two phases of the ESMFO algorithm are described in the next subsections.

#### 3.1. Exploration Phase

In this subsection, the exploration mechanism of ESMFO is proposed. The main inspiration of the MFO algorithm is the navigation mechanism of moths in nature called transverse orientation. When the moth is far away from the light source, the energy value \( E \) felt by the moth is less than 0.5, and the proposed algorithm still uses the spiral update formula from the MFO algorithm. This behavior is modeled by the following rule:

\[
S(M_i, F_j) = D_i \cdot e^{b \cdot \cos(2\pi t)} + F_j
\]  \hspace{1cm} (14)

#### 3.2. Exploitation Phase

In this phase, the moth feels increasingly intense energy as the distance between the moth and the flame is reduced; when the energy value is greater than 0.5, the influence of the flame on the moth is intensified. The moth cannot continue to maintain the navigation mechanism (lateral positioning), so instead, it performs a useless and fatal irregular flight strategy. The method proposed in this paper uses a new update formula instead of the update formula contained in the MFO algorithm. The current positions are updated using Eq. (15):

\[
M(t+1) = \begin{cases} 
Y & \text{if } F(Y) < F(S(t)) \\
Z & \text{if } F(Z) < F(S(t)) 
\end{cases}
\]  \hspace{1cm} (15)

where \( Y \) and \( Z \) are obtained using Eqs. (16)-(17).

\[
Y = M(t) - E|F(t) - M(t)|
\]  \hspace{1cm} (16)

\[
Z = M(t) - E|F(t) - M(t)| \cdot \text{Levy}(\beta)
\]  \hspace{1cm} (17)

where \text{Levy} is the Levy flight function, which is calculated using Eq. (13).

#### 3.3. Pseudocode of the ESMFO Algorithm

The pseudocode of the proposed ESMFO algorithm is given in Algorithm 1.

**Algorithm 1** The pseudo code of the ESMFO algorithm

Update flame_no using Eq. (8)

\( OM = \text{FitnessFunction}(M) \);

\( I f \) iteration ==1

\( F = \text{sort}(M); \)

\( OF = \text{sort}(OM); \)

\( e l s e \)

\( F = \text{sort}(M_{r-1}, M_i); \)

\( OF = \text{sort}(M_{r-1}, M_i); \)

\( e n d \)

\( f o r \ i = 1:n \)

\( f o r \ j = 1:d \)

Update initial energy \( E_0 \)

Update \( r \) and \( t \)

\( I f \) \( |E| \geq 0.5 \) \( t h e n \)

\( e n d \)

\( e n d \)

\( e n d \)

\( e n d \)

\( e n d \)

\( e n d \)

\( e n d \)
Calculate $D$ using Eq. (7) with respect to the corresponding moth:

```
else ([E] < 0.5) then
    Update $M(i, j)$ using Eqs. (16) and (17) with respect to the corresponding moth;
end
```

3.4. Computational Complexity of the ESMFO Algorithm

The computational complexity of an algorithm is a key metric for evaluating its run time, and this metric can be defined based on the structure and implementation of the algorithm. The computational complexity of the ESMFO algorithm depends on the number of moths, number of variables, maximum number of iterations, and sorting mechanism for the flames in each iteration. Since the Quicksort algorithm is utilized, the computational complexity of the sorting process is of $O(n \log n)$ and $O(n^2)$ for the best and worst cases, respectively. Therefore, the overall computational complexity is defined as follows:

$$O(ESMFO) = O(t \cdot (O(\text{QuickSort})+O(\text{position update})))$$

(18)

$$O(ESMFO) = O(t \cdot (O(n^2 + n \times d))) = O(t n^2 + t n d)$$

(19)

where $n$ is the number of moths, $t$ is the maximum number of iterations, and $d$ is the number of variables.

4. Results and Discussion

In this section, we analyze the effectiveness of the proposed algorithm using benchmark functions. Additionally, the results of ESMFO are compared with those of the basic MFO algorithm as well as with the results of other metaheuristic algorithms. The results are obtained for different dimension sizes to evaluate the effect of the algorithm.

4.1. Parameter Settings

For performance evaluation purposes, the results are obtained using a computer with a 64-bit Windows 10 operating system, an Intel Core i5-7200U, CPU@ 2.50 GHZ, 8 GB RAM, and MATLAB version R2017a. All the simulation experiments are performed on MATLAB. All the algorithms are run 30 times, each with 1000 iterations. The results of ESMFO are compared with those of the moth-flame optimization algorithm (MFO) [22-24], Levy flight moth-flame optimization algorithm (LMFO) [32], Harris hawks optimization algorithm (HHO) [26], whale optimization algorithm (WOA) [10][31][33], bat algorithm (BA) [34-37], and cuckoo search (CS) algorithm [38-41]. The various parameters of these algorithms are given in Table 1.

| Algorithm | Parameter | Value |
|-----------|-----------|-------|
| BA        | $A$ Loudness; $r$ Pulse rate | 0.5; 0.5 |
| CS        | Discovery rate of unknown solutions $pa$ | 0.25 |
| WOA       | $\alpha$ | Linearly decreased from 2 to 0 |
| MFO       | Spiral factor $b$ | 1 |
| LMFO      | Spiral factor $b$ | 1 |
| ESMFO     | Spiral factor $b$ | 1 |

4.2. Test Functions

To check its performance, the algorithm proposed in this paper is been verified on twenty one benchmark functions [42-44]. These benchmark functions include unimodal functions, multimodal functions and multimodal functions with fixed dimensions. Unimodal functions have only one global mining operation. On the other hand, multimodal functions have a large number of local mining operations. Here, it should be noted that unimodal functions help to check the exploitative ability of an algorithm, whereas multimodal functions help keep a check on the explorative capabilities of any algorithm. The benchmark functions are discussed in detail in Tables 2, 3 and 4.

| Benchmark function | Dim | Range $x_i \in [-100,100]$ | $f_{min}$ |
|-------------------|-----|------------------------|----------|
| $f_1(x) = \sum_{i=1}^{n} x_i^2$ | 30  | $x_i \in [-100,100]$ | 0        |
| $f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$ | 30  | $x_i \in [-10,10]$  | 0        |
\[
\begin{align*}
&f_3(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2 \quad 30 \quad x_i \in [-100,100] \quad 0 \\
&f_4(x) = \max_i \{|x_i|, 1 \leq i \leq D\} \quad 30 \quad x_i \in [-100,100] \quad 0 \\
&f_5(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad 30 \quad x_i \in [-30,30] \quad 0 \\
&f_6(x) = \sum_{i=1}^{n} x_i^4 + \text{random}(0,1) \quad 30 \quad x_i \in [-1.28,1.28] \quad 0
\end{align*}
\]

**Table 3**

MULTIMODAL, HIGH-DIMENSIONAL BENCHMARK TEST FUNCTION

| Benchmark function | Dim | Range            | \( f_{\text{min}} \) |
|--------------------|-----|-----------------|----------------------|
| \( f_7(x) = \sum_{i=1}^{n} [x_i^2 - 10 \cos(2\pi x_i) + 10] \) | 30  | \( x_i \in [-5.12,5.12] \) | 0                    |
| \( f_8(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos 2\pi x_i\right)\right) + 20 + e \) | 30  | \( x_i \in [-32,32] \) | 0                    |
| \( f_9(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i^2) - \prod_{j=1}^{n} \cos \left(\frac{x_i}{\sqrt{j}}\right) + 1 \) | 30  | \( x_i \in [-600,600] \) | 0                    |
| \( f_{10}(x) = 0.1 \sin^2(3\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + \sin^2(3\pi y_{i+1}) + (y_n - 1)^2(1 + \sin^2(2\pi y_n))\right] \) | 30  | \( x_i \in [-50,50] \) | 0                    |
| \( f_{11}(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \sin(x_i) + 0.1 x_j \right) \) | 30  | \( x_i \in [-10,10] \) | 0                    |
| \( f_{12}(x) = \sum_{i=1}^{n} x_i^4 + \left( \sum_{i=1}^{n} 0.5i x_i \right)^2 + \left( \sum_{i=1}^{n} 0.5i x_i \right)^4 \) | 30  | \( x_i \in [-5,10] \) | 0                    |

**Table 4**

MULTIMODAL, FIXED-DIMENSIONAL BENCHMARK TEST FUNCTION

| Benchmark function | Dim | Range   | \( f_{\text{min}} \) |
|--------------------|-----|---------|----------------------|
| \( f_{13}(x) = \left( \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_j)^6} \right)^{-1} \) | 2   | \( x_i \in [-65,65] \) | 1                    |
| \( f_{14}(x) = \sum_{i=1}^{11} \left[ a_i - \frac{x_i (b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2 \) | 4   | \( x_i \in [-5.5] \) | 0.0003075            |
| \( f_{15}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_4^2 \) | 2   | \( x_i \in [-5.5] \) | -1.0316285           |
In this section, we perform simulation experiments on the algorithm, analyze the effectiveness of its HHO-based modification, and verify its optimization performance. Tables 5 and 6 show the comparison of the ESMFO algorithms with other algorithms based on fixed-dimensional and high-dimensional functions. Table 5 shows that the ESMFO algorithm performs better than other algorithms on nine benchmark functions for fixed-dimensional functions, ranking first in the optimization rankings of all seven algorithms. This demonstrates the efficiency of the optimization performance achieved by the ESMFO algorithm.

\[
f_{16}(x) = (x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 + 6)^2 + 10(1 - \frac{1}{8\pi}) \cos x_1 + 10
\]

\[
f_{17}(x) = [1 + (x_1 + x_2 + 1)]^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 48x_2 - 36x_1x_2 + 27x_2^2)]
\]

\[
f_{18}(x) = \sum_{i=1}^{4} c_i \exp \left[ \sum_{j=3}^{13} a_{ij} (x_j - p_j)^2 \right]
\]

\[
f_{19}(x) = -\cos(12\sqrt{x_1^2 + x_2^2}) \frac{0.5(x_1^2 + x_2^2)}{2}
\]

\[
f_{20}(x) = 0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2)^2)}
\]

\[
f_{21}(x) = -\cos(x_1) \cos(x_2) \exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)
\]

### 4.3. Test Function Results and Analysis

In this section, we perform simulation experiments on the algorithm, analyze the effectiveness of its HHO-based modification, and verify its optimization performance. Tables 5 and 6 show the comparison of the ESMFO algorithms with other algorithms based on fixed-dimensional and high-dimensional functions. Table 5 shows that the ESMFO algorithm performs better than other algorithms on nine benchmark functions for fixed-dimensional functions, ranking first in the optimization rankings of all seven algorithms. This demonstrates the efficiency of the optimization performance achieved by the ESMFO algorithm.

Fig. 1 to Fig. 9 show the best fitness curves for each function, and Fig. 10 to Fig. 18 show the variance diagrams for each function. From the curves and variance diagrams, we can see that the ESMFO algorithm presented in this paper has the highest optimization accuracy and fastest convergence speed on the fixed-dimensional test function. At the same time, we can see that the algorithm shows strong stability. Therefore, the ESMFO algorithm achieves excellent performance on fixed-dimensional benchmark functions.

| Algorithm | MFO | LMFO | HHMFO | HHO | WOA | BA | CS | RANK |
|-----------|-----|------|-------|-----|-----|----|----|------|
| **f_{13}-2dim** | **Best** | 9.98E-01 | 9.98E-01 | 9.98E-01 | 9.98E-01 | 9.98E-01 | 9.98E-01 | 9.98E-01 |
|           | **Worst** | 4.95E+00 | 1.26E+01 | 5.93E+00 | 1.08E+01 | 1.27E+01 | 9.98E-01 | 9.98E-01 |
|           | **Mean**  | 1.69E+00 | 4.98E+00 | 5.42E+00 | 1.20E+00 | 1.82E+00 | 1.16E+01 | 9.98E-01 |
|           | **Std**   | 9.78E-01 | 4.06E-00 | 4.75E+00 | 9.12E-01 | 1.87E+00 | 2.90E+00 | 0.00E+00 |
| **f_{14}-4dim** | **Best** | 6.27E-04 | 3.10E-04 | 3.08E-04 | 3.08E-04 | 3.08E-04 | 3.16E-04 | 3.08E-04 | 3.07E-04 |
|          | **Worst** | 2.04E-02 | 1.24E-03 | 1.22E-03 | 1.49E-03 | 2.24E-03 | 5.69E-03 | 3.08E-04 | 1 |
|          | **Mean**  | 0.001556 | 3.85E-04 | 3.95E-04 | 3.68E-04 | 8.29E-04 | 1.20E-03 | 3.08E-04 |
|          | **Std**   | 0.003565 | 1.69E-04 | 2.32E-04 | 2.13E-04 | 4.59E-04 | 1.53E-03 | 4.84E-08 |
| **f_{15}-2dim** | **Best** | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 |
|          | **Worst** | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 |
|          | **Mean**  | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 | -1.03E+00 |
|          | **Std**   | 6.78E-16 | 1.14E-05 | 3.69E-06 | 1.61E-11 | 3.21E-11 | 6.54E-08 | 6.78E-16 |
| Function | Best | Worst | Mean | Std   |
|----------|------|-------|------|-------|
| f_{16-2dim} | 3.98E-01 | 3.98E-01 | 3.98E-01 | 0.00E+00 |
| f_{17-2dim} | 3.98E-01 | 3.98E-01 | 3.98E-01 | 0.00E+00 |
| f_{18-3dim} | 3.98E-01 | 3.98E-01 | 3.98E-01 | 0.00E+00 |
| f_{19-2dim} | -3.86E+00 | -3.86E+00 | -3.86E+00 | -3.86E+00 |
| f_{20-2dim} | -1.00E+00 | -1.00E+00 | -1.00E+00 | -1.00E+00 |
| f_{21-2dim} | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |

**FIGURE 1** Evolution curves of the fitness values for $f_{13}$

**FIGURE 2** Evolution curves of the fitness values for $f_{14}$
Evolution curves of the fitness values for $f_{15}$

Evolution curves of the fitness values for $f_{16}$

Evolution curves of the fitness values for $f_{17}$

Evolution curves of the fitness values for $f_{18}$

Figure 3

Evolution curves of the fitness values for $f_{19}$

Evolution curves of the fitness values for $f_{20}$

Evolution curves of the fitness values for $f_{21}$

Variance diagrams for $f_{13}$

Figure 7

Figure 8

Figure 9

Figure 10
From Table 6, we can see that the ESMFO algorithm has strong competitiveness in the remaining 11 high-dimensional standard test functions (except for the midstream of the best fitness values obtained on the $f_{10}$ function), and its optimization accuracy ranks first. Fig. 19 to Fig. 42 represent the curves and variance graphs for 12 high-dimensional functions. Similarly, we can see that except for the instability of ESMFO for individual functions and the fact that it occasionally falls into local optima, the functions have strong stability in search accuracy and convergence speed.

![Figure 19: Evolution curves of the fitness values for $f_1$](image1)

![Figure 20: Evolution curves of the fitness values for $f_2$](image2)

![Figure 21: Evolution curves of the fitness values for $f_3$](image3)

### Table 6

RESULTS OF ESMFO COMPARED WITH THOSE OF THE OTHER ALGORITHMS FOR THE CASE WITH DIMENSION SIZE = 50

| Algorithm | MFO | LMFO | ESMFO | HHO | WOA | BA | CS | RANK |
|-----------|-----|------|-------|-----|-----|----|----|------|
| $f_1$-50dim | Best | 1.42E+00 | 4.17E-275 | 0 | 2.58E-216 | 1.23E-177 | 3.93E-03 | 6.82E-01 | 6 |
|           | Worst | 2.00E+04 | 2.69E-201 | 4.72E-285 | 6.65E-181 | 2.08E-143 | 5.94E-03 | 2.74E+00 | 1 |
|           | Mean | 5.37E+03 | 8.95E-203 | 1.66E-286 | 2.22E-182 | 6.95E-145 | 4.82E-03 | 1.71E+00 | 1 |
|           | Std  | 5.69E+03 | 0 | 0 | 0 | 3.80E-144 | 4.88E-04 | 6.27E-01 | 1 |
| $f_2$-50dim | Best | 3.01E+01 | 2.01E-127 | 1.73E-174 | 2.67E-106 | 3.01E-112 | 4.22E-01 | 2.78E+01 | 1 |
|           | Worst | 1.30E+02 | 5.89E-108 | 1.11E-144 | 1.75E-94 | 1.78E-103 | 2.83E+00 | 1.00E+10 | 1 |
|           | Mean | 7.32E+01 | 4.36E-109 | 4.19E-146 | 1.23E-95 | 9.72E-105 | 1.34E+00 | 6.67E+09 | 1 |
|           | Std  | 3.26E+01 | 1.37E-108 | 2.03E-145 | 3.90E-95 | 3.34E-104 | 6.25E-01 | 4.79E+09 | 1 |
| $f_3$-50dim | Best | 1.99E+04 | 8.63E-222 | 0 | 1.34E-186 | 7.20E+04 | 7.16E-02 | 2.33E+03 | 1 |
|           | Worst | 7.38E+04 | 5.82E-167 | 4.21E-274 | 3.00E-136 | 1.81E+05 | 1.71E-01 | 4.92E+03 | 1 |
|           | Mean | 4.31E+04 | 1.94E-168 | 1.40E-275 | 1.00E-137 | 1.25E+05 | 1.20E-01 | 3.56E+03 | 1 |
|           | Std  | 1.52E+04 | 0 | 0 | 5.47E-137 | 2.70E+04 | 2.62E-02 | 7.95E+02 | 1 |
| $f_4$-50dim | Best | 7.37E+01 | 4.10E-118 | 2.25E-165 | 1.17E-105 | 1.68E+01 | 2.05E-01 | 1.02E+01 | 1 |
|           | Worst | 9.19E+01 | 4.41E-93 | 2.84E-138 | 1.57E-91 | 9.28E-01 | 6.56E-01 | 1.69E+01 | 1 |
|           | Mean | 8.40E+01 | 1.48E-94 | 9.74E-140 | 5.98E-93 | 6.54E-01 | 4.76E-01 | 1.39E+01 | 1 |
|           | Std  | 4.71E+00 | 8.05E-94 | 5.19E-139 | 2.86E-92 | 2.48E-01 | 1.08E-01 | 2.18E+00 | 1 |
| $f_5$-50dim | Best | 7.59E+02 | 4.66E+01 | 4.66E+01 | 2.99E-05 | 4.68E+01 | 4.54E+01 | 1.42E+02 | 2 |
|           | Worst | 8.00E+07 | 4.87E+01 | 4.87E+01 | 3.86E-02 | 4.85E+01 | 1.10E+02 | 1.25E+03 | 2 |
|           | Mean | 7.88E+06 | 4.75E+01 | 4.77E+01 | 6.07E-03 | 4.76E+01 | 5.21E+01 | 5.39E+02 | 2 |
| Function | Best | Worst | Mean | Std |
|----------|------|-------|------|-----|
| f_4 | 3.05E-01 | 7.57E+01 | 1.71E+01 | 1.84E+01 |
| f_5 | 2.00E+01 | 8.88E-16 | 1.93E+01 | 8.14E-01 |
| f_6 | 9.89E-01 | 2.71E+02 | 5.91E+01 | 7.23E+01 |
| f_7 | 1.74E+01 | 3.92E+00 | 1.93E+01 | 7.23E+01 |
| f_8 | 2.06E-02 | 5.22E+02 | 3.50E+02 | 2.03E-01 |
| f_9 | 2.13E+02 | 4.08E+02 | 1.20E+02 | 3.50E+02 |
| f_10 | 3.05E-01 | 7.57E+01 | 1.71E+01 | 1.84E+01 |
| f_11 | 2.00E+01 | 8.88E-16 | 1.93E+01 | 8.14E-01 |
| f_12 | 9.89E-01 | 2.71E+02 | 5.91E+01 | 7.23E+01 |
| f_13 | 1.74E+01 | 3.92E+00 | 1.93E+01 | 7.23E+01 |

**FIGURE 22** Evolution curves of the fitness values for f_4

**FIGURE 23** Evolution curves of the fitness values for f_5
FIGURE 32 Variance diagrams for $f_2$

FIGURE 33 Variance diagrams for $f_3$

FIGURE 34 Variance diagrams for $f_4$

FIGURE 35 Variance diagrams for $f_5$

FIGURE 36 Variance diagrams for $f_6$

FIGURE 37 Variance diagrams for $f_7$

FIGURE 38 Variance diagrams for $f_8$

FIGURE 39 Variance diagrams for $f_9$
It can be seen from the above figures that the energy factor in the HHO algorithm provides MFO with a variety of optimization solutions, which improve the exploration task and enable the algorithm to achieve the global minima of more functions than those of the HHO algorithm. The results show that almost all the functions can provide good optimal values and standard deviation values for fixed-dimensional and high-dimensional benchmark functions. The next section details the impact of the number of dimensions on ESMFO performance.

### 4.4. Effect of the Number of Dimensions (Scalability Study)

The performance of the method is evaluated with different numbers of dimensions. Four different dimensions (50, 100, 200, and 500) are considered to investigate their effects on MFO and ESMFO. They are compared with meta-heuristic algorithms such as the BA, HHO, WOA, and CS algorithms. The results of MFO and ESMFO for different dimension sizes are evaluated only on unimodal and multimodal functions which are given in Tables 2 and 3. The performances of the 12 functions are evaluated; however, the dimensions of the fixed-dimensional function cannot be changed. In this experiment, the population size is taken as 30.

The results obtained with 50 dimensions are discussed in the previous section. It is apparent from Table 6 that ESMFO performs best in 12 functions, and with 50 dimensions, HHO performs well on function \( f_{10} \). In the case where dimension=100, ESMFO reaches a global minimum for each function except \( f_5 \) and \( f_{10} \), as shown in Table 7. For functions \( f_5 \) and \( f_{10} \), HHO provides the best results. Table 8 shows the simulation results when the number of dimensions is 500, and the results show that ESMFO performs well on all functions. The simulation results with dimension=1000 are shown in Table 9. The results show that HHO achieves good results on \( f_5 \), and ESMFO has excellent performance for all other functions. For the \( f_5 \) and \( f_{10} \) functions, all algorithms provide competitive results in terms of the best results, worst results, average results and standard deviations. However, HHO is considered to be the best due to the fact that it obtains the best optimal values.

| Function | Algorithm | MFO | LMFO | ESMFO | HHO | WOA | BA | CS | RANK |
|----------|-----------|-----|------|-------|-----|-----|----|----|------|
| \( f_{10} \) | Best | 8.35E+03 | 1.75E-240 | 0 | 6.86E-205 | 2.62E-165 | 2.12E-02 | 1.37E-02 | 1.37E+02 |
| | Worst | 5.45E+04 | 5.66E-203 | 2.87E-274 | 3.61E-178 | 6.94E-146 | 2.60E-02 | 3.86E-02 | 3.86E+02 |
| | Mean | 3.19E+06 | 1.89E-204 | 9.56E-276 | 1.20E-179 | 2.34E-147 | 2.36E-02 | 2.69E+02 | 2 |
| | Std | 1.31E+04 | 0 | 0 | 0 | 1.27E-146 | 1.19E-03 | 7.30E+01 | |
| \( f_{100} \) | Best | 9.05E+01 | 8.42E-127 | \textbf{2.63E-173} | 7.93E-110 | 4.19E-112 | 2.66E+00 | 1.00E+10 | 1 |
| | Worst | 2.65E+02 | 1.13E-104 | 4.34E-143 | 5.07E-95 | 3.10E-09 | 7.03E+00 | 1.00E+10 | 1 |
|       | F_1-100dim |       | F_2-100dim |       | F_3-100dim |       | F_4-100dim |       | F_5-100dim |       | F_6-100dim |       | F_7-100dim |       | F_8-100dim |       | F_9-100dim |       | F_10-100dim |       | F_11-100dim |       | F_12-100dim |       |
|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|-----------|-------|
| **Mean** | 1.77E+02  | 3.90E-106 | 1.49E-144 | 2.62E-96 | 1.04E-100 | 4.25E+00 | 1.00E+10 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Std**  | 4.45E+01  | 2.05E-105 | 7.92E-144 | 1.03E-95 | 5.66E-100 | 1.20E+00 | 0.00E+00 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Best** | 1.00E+05  | 9.36E-224 | 6.36E-320 | 6.60E-168 | 5.47E+05  | 3.02E+00 | 2.38E+04 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Worst** | 3.37E+05  | 1.26E-177 | 1.65E-272 | 6.07E-108 | 1.23E+06 | 5.35E+00 | 4.81E+04 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Mean** | 2.04E+05  | 4.21E-179 | 5.60E-274 | 2.02E-109 | 8.66E+05 | 4.43E+00 | 3.18E+04 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Std**  | 6.02E+04  | 0.00E+00  | 0.00E+00  | 1.11E-108 | 1.63E+05 | 6.59E-01 | 6.09E+03 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Best** | 1.00E+05  | 9.36E-224 | 6.36E-320 | 6.60E-168 | 5.47E+05 | 3.02E+00 | 2.38E+04 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Worst** | 3.37E+05  | 1.26E-177 | 1.65E-272 | 6.07E-108 | 1.23E+06 | 5.35E+00 | 4.81E+04 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Mean** | 2.04E+05  | 4.21E-179 | 5.60E-274 | 2.02E-109 | 8.66E+05 | 4.43E+00 | 3.18E+04 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |
| **Std**  | 6.02E+04  | 0.00E+00  | 0.00E+00  | 1.11E-108 | 1.63E+05 | 6.59E-01 | 6.09E+03 |       |          |       |          |       |          |       |          |       |          |       |          |       |          |       |

Yuanfei Wei, Pengchuan Wang, Qifang Luo, Yongquan Zhou

E-ISSN: 2224-266X

Volume 19, 2020
| Function | Algorithm | MFO | LMFO | ESMFO | HHO | WOA | BA | CS | RANK |
|----------|-----------|-----|------|-------|-----|-----|----|----|------|
| **F1-500dim** | Best | 8.74E+05 | 5.57E-247 | 0 | 3.54E-209 | 2.07E-164 | 1.03E+00 | 3.24E+04 | |
| Worst | 1.01E+06 | 1.94E-191 | 3.20E-280 | 4.50E-183 | 1.08E-146 | 1.22E+00 | 5.01E+04 | 1 |
| Mean | 9.52E+05 | 7.16E-193 | 1.22E-281 | 1.99E-184 | 5.31E-148 | 1.13E+00 | 3.91E+04 | |
| Std | 3.32E+04 | 0 | 0 | 0 | 2.05E-147 | 5.72E-02 | 3.94E+03 | |
| **F2-500dim** | Best | 2.00E+03 | 1.31E-123 | **2.16E-177** | 4.80E-108 | 5.62E-109 | 5.18E+01 | 1.00E+10 | 1 |
| Worst | 2.54E+03 | 1.50E-96 | 2.95E-177 | 1.06E-94 | 6.36E-99 | 7.12E+01 | 1.00E+10 | |
| Mean | 2.24E+03 | 5.02E-98 | 9.96E-141 | 5.29E-96 | 2.26E-100 | 6.30E+01 | 1.00E+10 | |
| Std | 9.92E+01 | 2.74E-97 | 5.38E-140 | 1.98E-95 | 1.16E-99 | 4.36E+00 | 1.00E+00 | |
| **F3-500dim** | Best | 2.67E+06 | 4.83E-206 | 0 | 2.93E-146 | 1.43E+07 | 3.03E+02 | 9.11E+05 | |
| Worst | 5.33E+06 | 1.94E-191 | 3.20E-280 | 4.50E-183 | 1.08E-146 | 1.22E+00 | 5.01E+04 | 1 |
| Mean | 3.81E+06 | 7.61E-169 | 1.11E-252 | 1.57E-82 | 5.31E-148 | 1.13E+00 | 3.91E+04 | |
| Std | 7.02E+05 | 0 | 0 | 0 | 8.44E-107 | 2.18E+01 | 1.60E+05 | |
| **F4-500dim** | Best | 9.83E+01 | 6.25E-112 | **4.74E-167** | 1.99E-103 | 4.08E+00 | 1.61E+00 | 3.30E+01 | |
| Worst | 9.95E+01 | 9.38E-85 | 2.36E-138 | 1.95E-90 | 1.99E+00 | 1.41E+01 | 3.67E+01 | 1 |
| Mean | 9.90E+01 | 3.18E-85 | 6.64E-92 | 7.52E+01 | 1.79E+00 | 1.47E+06 | 3.67E+01 | |
| Std | 3.23E-01 | 1.74E-85 | 4.32E-139 | 3.55E-91 | 2.65E+01 | 8.21E+00 | 2.07E+00 | |
| **F5-500dim** | Best | 2.59E+04 | 1.68E-06 | **8.74E-07** | 7.89E-05 | 1.11E+01 | 6.53E+01 | 2.27E+06 | 3 |
| Worst | 4.57E+04 | 2.51E-04 | 6.26E-04 | 1.06E-02 | 1.94E+01 | 1.56E+02 | 1.73E+07 | |
| Mean | 4.08E+04 | 3.99E-02 | 8.75E-02 | 4.96E+02 | 8.05E+03 | 8.16E+06 | 1.16E+07 | |
| Std | 2.41E+08 | 1.21E-01 | 1.23E-01 | 1.25E-01 | 4.71E+01 | 8.53E+01 | 2.27E+06 | |
| **F6-500dim** | Best | 6.08E+03 | 0 | 0 | 0 | 0 | 6.52E+02 | 3.37E+03 | |
| Worst | 6.64E+03 | 0 | 0 | 0 | 0 | 7.77E+02 | 3.87E+03 | 1 |
| Mean | 6.45E+03 | 0 | 0 | 0 | 0 | 7.16E+02 | 3.67E+03 | |
| Std | 1.39E+02 | 0 | 0 | 0 | 0 | 3.49E+01 | 1.08E+02 | |
| **F7-500dim** | Best | 2.00E+01 | **8.88E-16** | **8.88E-16** | **8.88E-16** | **8.88E-16** | 3.25E+00 | 1.13E+01 | 1 |
| Worst | 2.03E+01 | 8.88E-16 | 8.88E-16 | 8.88E-16 | 7.99E-15 | 3.56E+00 | 1.48E+01 | |
| Mean | 2.01E+01 | 8.88E-16 | 8.88E-16 | 8.88E-16 | 3.49E-15 | 3.39E+00 | 1.23E+01 | |
| Std | 1.32E-01 | 0 | 0 | 0 | 0 | 2.46E-15 | 7.91E-02 | 1.07E+00 | |
| **F8-500dim** | Best | 8.00E+03 | 0 | 0 | 0 | 0 | 1.34E+02 | 2.99E+02 | |
| Worst | 9.46E+03 | 0 | 0 | 0 | 0 | 1.76E-02 | 4.13E+02 | 1 |
| Mean | 8.62E+03 | 0 | 0 | 0 | 0 | 1.58E-02 | 3.65E+02 | |
| Std | 3.37E+02 | 0 | 0 | 0 | 0 | 1.17E-03 | 3.04E+01 | |
| **F9-500dim** | Best | 1.56E+10 | 4.95E+01 | 4.97E+01 | **4.28E-06** | 5.25E+00 | 8.40E+01 | 1.00E+10 | 4 |
| Worst | 2.06E+10 | 4.98E+01 | 4.99E+01 | 4.08E-04 | 1.99E+01 | 1.00E+02 | 1.00E+10 | |
### Table 9

RESULTS OF ESMFO COMPARED THOSE OF THE OTHER ALGORITHMS FOR THE CASE WITH DIMENSION SIZE = 1000

| Function | Algorithm | MFO | LMFO | ESMFO | HHO | WOA | BA | CS | RANK |
|----------|-----------|-----|------|-------|-----|-----|----|----|------|
| **F1-1000dim** | Best | 2.35E+06 | 1.72E-229 | 0 | 3.34E-206 | 3.88E-158 | 8.48E+00 | 1.12E+05 | 1 |
| | Worst | 2.56E+06 | 5.94E-184 | 2.56E-270 | 1.44E-179 | 9.93E-145 | 1.12E+01 | 1.39E+05 | 1 |
| | Mean | 2.47E+06 | 4.69E-122 | 1.39E-163 | 2.08E-103 | 3.77E-115 | 9.90E+00 | 1.26E+05 | 1 |
| | Std | 5.14E+04 | 4.09E+03 | 7.16E+03 | 6.84E+02 | 3.51E+01 | 2.23E+03 | 6.57E+03 | 1 |
| **F2-1000dim** | Best | Inf | 4.36E-121 | 1.39E-163 | 2.08E-103 | 3.77E-115 | 9.90E+00 | 1.26E+05 | 1 |
| | Worst | Inf | 2.10E-99 | 5.90E-139 | 3.16E-92 | 2.31E-99 | 2.27E+02 | 1.00E+10 | 1 |
| | Mean | Inf | 7.02E-101 | 1.92E-102 | 7.71E-64 | 1.18E+03 | 4.26E+02 | 1.08E+02 | 1.00E+10 | 1 |
| | Std | NaN | 2.57E+06 | 1.08E+02 | 4.26E+02 | 1.08E+02 | 8.43E+05 | 5.26E+06 | 1 |
| **F3-1000dim** | Best | 9.92E+01 | 9.03E-109 | 2.04E-102 | 2.71E-108 | 3.54E+01 | 1.96E+00 | 3.53E+01 | 1 |
| | Worst | 9.97E+01 | 2.27E-80 | 6.32E-137 | 1.15E-89 | 9.95E+01 | 2.23E+00 | 4.60E+01 | 1 |
| | Mean | 9.95E+01 | 7.57E-82 | 3.83E-138 | 3.96E-91 | 7.92E+01 | 2.09E+00 | 4.00E+01 | 1 |
| | Std | 2.57E+06 | 5.34E-161 | 0 | 4.22E-63 | 4.26E+02 | 1.08E+02 | 8.43E+05 | 1 |
| **F4-1000dim** | Best | 9.92E+01 | 9.03E-109 | 2.04E-102 | 2.71E-108 | 3.54E+01 | 1.96E+00 | 3.53E+01 | 1 |
| | Worst | 9.97E+01 | 2.27E-80 | 6.32E-137 | 1.15E-89 | 9.95E+01 | 2.23E+00 | 4.60E+01 | 1 |
| | Mean | 9.95E+01 | 7.57E-82 | 3.83E-138 | 3.96E-91 | 7.92E+01 | 2.09E+00 | 4.00E+01 | 1 |
| | Std | 2.57E+06 | 5.34E-161 | 0 | 4.22E-63 | 4.26E+02 | 1.08E+02 | 8.43E+05 | 1 |
| **F5-1000dim** | Best | 1.03E+10 | 9.98E+02 | 9.98E+02 | 1.33E-04 | 9.91E+02 | 3.81E+03 | 2.83E+07 | 3 |
| | Worst | 1.16E+10 | 9.99E+02 | 9.98E+02 | 6.08E+01 | 9.95E+02 | 4.79E+03 | 5.74E+07 | 3 |
| | Mean | 1.11E+10 | 9.99E+02 | 9.98E+02 | 1.20E+01 | 9.93E+02 | 4.25E+03 | 4.10E+07 | 3 |
| | Std | 3.78E+08 | 7.42E-02 | 1.31E-01 | 1.57E-02 | 9.57E-02 | 2.90E+02 | 6.41E+06 | 3 |
| **F6-1000dim** | Best | 1.59E+05 | 1.31E-06 | 2.10E-06 | 3.11E-06 | 9.85E-05 | 3.35E+02 | 5.00E+02 | 1 |
| | Worst | 1.87E+05 | 3.07E-04 | 2.39E-04 | 3.11E-04 | 1.10E-02 | 6.27E+02 | 8.97E+02 | 1 |
| | Mean | 1.73E+05 | 6.83E-05 | 5.36E-05 | 6.67E-05 | 2.45E-03 | 4.90E+02 | 6.51E+02 | 1 |
| | Std | 7.03E+03 | 7.45E-05 | 4.95E-05 | 7.04E-05 | 2.54E-03 | 6.35E+01 | 9.13E+01 | 1 |
| **F7-1000dim** | Best | 1.43E+04 | 0 | 0 | 0 | 0 | 1.93E+03 | 8.08E+03 | 1 |
| | Worst | 1.52E+04 | 0 | 0 | 0 | 0 | 2.18E+03 | 8.70E+03 | 1 |
| | Mean | 1.48E+04 | 0 | 0 | 0 | 0 | 2.10E+03 | 8.31E+03 | 1 |
The effects of using 50, 100, 200, and 500 dimensions are evaluated. We randomly select several functions from the 12 total test functions to compare the optimal values obtained by the algorithms in three-dimensional stereo histograms and record the optimal fitness values of each algorithm in different dimensions, as shown in Figs. 43-48. From the graph, we can see that ESMFO has obvious advantages over the MFO, WOA, BA and CS algorithms in several test functions. As the number of dimensions increases, the convergence accuracy of each of the four algorithms decreases successively. Compared with the accuracy of the LMFO and HHO algorithms, this difference is not obvious, but we can see from the local comparison graphs for the f2 function (Figs. 45-46), the performance of ESMFO is better than those of the LMFO and HHO algorithms. The results show that ESMFO is superior to the other algorithms in all dimensions. ESMFO provides good results for all functions except f5 and f10. The results show that the performance of the ESMFO algorithm decreases as the number of dimensions increases, but it is better than those of the other algorithms.

The effects of using 50, 100, 200, and 500 dimensions are evaluated. We randomly select several functions from the 12 total test functions to compare the optimal values obtained by the algorithms in three-dimensional stereo histograms and record the optimal fitness values of each algorithm in different dimensions, as shown in Figs. 43-48. From the graph, we can see that ESMFO has obvious advantages over the MFO, WOA, BA and CS algorithms in several test functions. As the number of dimensions increases, the convergence accuracy of each of the four algorithms decreases successively. Compared with the accuracy of the LMFO and HHO algorithms, this difference is not obvious, but we can see from the local comparison graphs for the f2 function (Figs. 45-46), the performance of ESMFO is better than those of the LMFO and HHO algorithms. The results show that ESMFO is superior to the other algorithms in all dimensions. ESMFO provides good results for all functions except f5 and f10. The results show that the performance of the ESMFO algorithm decreases as the number of dimensions increases, but it is better than those of the other algorithms.

| Algorithm | Std | Best | Mean | Worst |
|-----------|-----|------|------|-------|
| F8-1000dim | 246.3925 | 8.88E-16 | 8.88E-16 | 8.88E-16 |
| F9-1000dim | 2.02E+01 | 8.88E-16 | 8.88E-16 | 8.88E-16 |
| F10-1000dim | 2.05E+01 | 8.88E-16 | 8.88E-16 | 8.88E-16 |
| F11-1000dim | 2.07E+01 | 8.88E-16 | 8.88E-16 | 8.88E-16 |
| F12-1000dim | 2.09E+01 | 8.88E-16 | 8.88E-16 | 8.88E-16 |

FIGURE 43 Comparison of the optimal fitness values obtained by the algorithms based on the number of dimensions of f1.
FIGURE 44 Comparison of the optimal fitness values obtained by the algorithms based on the number of dimensions of $f_2$

FIGURE 45 Local comparison of the optimal fitness values obtained by the algorithms based on the number of dimensions of $f_2$

FIGURE 46 Local comparison of the optimal fitness values obtained by the algorithms based on the number of dimensions of $f_2$

FIGURE 47 Comparison of the optimal fitness values obtained by the algorithms based on the number of dimensions of $f_2$
In the basic MFO algorithm, the parameter $b$ defines the shape of the spiral. The spiral chosen for this paper is a logarithmic spiral whose equation is as follows:

$$r = e^{b\theta} \quad (18)$$

where $r$ and $\theta$ are the polar coordinates of the curve that define the distance from the origin and the angle from the $x$-axis, respectively, and $b$ is any real positive constant. The rate of change of the radius with respect to $\theta$ is given as:

$$\frac{dr}{d\theta} = b \cdot e^{b\theta} = b \cdot r \quad (19)$$

From these equations, it is observed that $b$ is the parameter that defines how tightly and in which direction a spiral is spun. There are two extreme cases for the values of $b$: for $b = 0$, the spiral is converted into a circle, and for $b = \infty$, the spiral becomes a straight line. The value chosen in the original article about the MFO algorithm for this parameter is 1, and no parametric study for its use has been given. Therefore, in order to evaluate the effect of $b$ on the performance of the algorithm, four different values of this parameter are compared.

**Experimental analysis:** The results are obtained for different values of $b$, such as 0.5, 1 and 2. The population size and number of iterations are set as 30 and 1000, respectively, during the experimental analysis. This analysis is conducted on both the MFO and ESMFO algorithms. First, for MFO with $b = 0.5$, the algorithm achieves the best results for six functions in terms of the best results, worst results, average results and standard deviations. When the
experiment is performed with \( b = 1 \), five functions yield good results, whereas at \( b = 2 \) only one function yields good results. Eight functions yield highly competitive results for all values of \( b \). Furthermore, for the ESMFO algorithm, it is concluded from Table 10 that twelve functions offer highly competitive results for all values of \( b \). The five functions \( f_2, f_4, f_5, f_{10} \) and \( f_{12} \) are successful in obtaining global optima for \( b = 1 \); on the other hand, for \( b = 0.5 \), only 3 out of 21 functions, i.e., \( f_6, f_{11} \) and \( f_{14} \), yield good results. For \( b = 2 \), only one function is successful in obtaining optimal results (Table 10).

**Inference:** In the basic MFO algorithm, the parameter \( b \) basically defines the shape of the spiral, and its value is set to 1. The author does not provide any explanation for the chosen value. Therefore, an experimental analysis is conducted to analyze the effect of \( b \) on the algorithm. Different values of \( b \) are chosen: 0.5, 1 and 2. It is clear from the above analysis that \( b = 1 \) is the ideal value for ESMFO.
| Function | Average | SD | Best | Worst | Average | SD |
|---------|---------|----|------|-------|---------|----|
| f8      | 1.69E+02 | 3.86E+01 | 1.65E+00 | 1.99E+01 | 1.24E+01 | 7.63E+00 |
|         | 0       | 0   | 8.88E-16 | 1.99E+00 | 1.43E+01 | 7.98E+00 |
| f9      | 7.91E-06 | 3.29E-02 | 7.91E-06 | 2.55E-05 | 2.98E-03 | 8.88E-16 |
| f10     | 1.00E-04 | 4.71E-06 | 1.07E-180 | 4.19E-05 | 2.11E-04 | 7.91E-06 |
|         | 1.43E+00 | 9.42E-04 | 1.93E+00 | 4.02E-14 | 5.45E-01 | 1.93E+00 |
| f11     | 3.56E-04 | 1.86E+01 | 7.02E-322 | 2.11E-179 | 5.26E-00 | 3.56E-04 |
|         | 3.08E-04 | 6.18E-04 | 1.34E-141 | 3.35E-02 | 7.33E-14 | 3.56E-04 |
| f12     | 3.77E-03 | 2.35E-02 | 1.34E-01 | 2.35E-02 | 4.53E-00 | 3.77E-03 |
|         | 3.08E-03 | 6.18E-04 | 1.34E-01 | 2.35E-02 | 4.53E-00 | 3.77E-03 |
| f13     | -1.306  | -1.306  | -1.306  | -1.306  | -1.306  | -1.306  |
| f15     | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 |
|         | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 |
| f16     | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 |
|         | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 | 0.39798 |
| f17     | 3       | 3     | 3      | 3      | 3      | 3      |
4.6. Statistical Testing

The statistical testing used includes the Wilcoxon rank-sum test [38][39], which tests the performances of two different algorithms. Here, the performance of ESMFO is checked with respect to those of the BA, CS, WOA, HHO, MFO and LMFO algorithms. Basically, this test is used to find the difference in performance between two algorithms, and at the end of this test we obtain a \( p \)-value as the output. This \( p \)-value signifies the significance of the algorithm being tested. If the \( p \)-value is lower than 0.05, then the corresponding algorithm is said to be statically significant. The data shown by the red horizontal line represent a large similarity between ESMFO and another algorithm. It can be seen from the table that ESMFO has similarity with two or more algorithms for functions f5, f6, and f18. The data obtained from Table 5 and Table 6 can be seen in these three functions. The algorithm reaches the convergence state and the optimization obtains the optimal value. It is proven that the comparison of the data is statistically significant. The results shown in Table 10 demonstrate that ESMFO performs better than the competing algorithms on twenty functions. An NA value for any algorithm shows the algorithm’s superiority over other algorithms, and the \( p \)-values of these algorithms are given with respect to the superior algorithm. The statistical results prove that ESMFO is a better algorithm than the others (Table 11).

| Function | MFO | LMFO | HHO | WOA | BA | CS |
|----------|-----|------|-----|-----|----|----|
| f18      | -0.30048 | -0.30048 | -0.30048 | -0.30048 | -0.30048 | -0.30048 |
| f19      | -1 | -1 | -1 | -1 | -1 | -1 |
| f20      | -0.99974 | -0.99994 | -1 | -1 | -1 | -1 |
| f21      | 0.8534 | 0.9117 | 1.69E-09 | 3.02E-11 | 3.02E-11 | 3.02E-11 |

The data shown by the red horizontal line represent a large similarity between ESMFO and another algorithm. It can be seen from the table that ESMFO has similarity with two or more algorithms for functions f5, f6, and f18. The data obtained from Table 5 and Table 6 can be seen in these three functions. The algorithm reaches the convergence state and the optimization obtains the optimal value. It is proven that the comparison of the data is statistically significant. The results shown in Table 10 demonstrate that ESMFO performs better than the competing algorithms on twenty functions. An NA value for any algorithm shows the algorithm’s superiority over other algorithms, and the \( p \)-values of these algorithms are given with respect to the superior algorithm. The statistical results prove that ESMFO is a better algorithm than the others (Table 11).
4.7. Pressure Vessel Design Problem

This problem was first proposed by Kannan and Kramer (1994) to minimize the total cost of the materials and the processes of forming and welding cylindrical containers. The schematic view of the pressure vessel design problem is shown in Fig. 51. There are four design variables:

![Fig. 51 Pressure vessel design problem]

\[ T_s \text{ ( } x_1 \text{ , thickness of the shell)}; \ T_h \text{ ( } x_2 \text{ , thickness of the head)}; \ R \text{ ( } x_3 \text{ , inner radius)}; \text{ and } L \text{ ( } x_4 \text{ , length of the cylindrical section without considering the head).} \]

Consider \( x = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L] \):

Minimize

\[ f(x) = 0.6224x_1x_2x_4 + 1.778lx_2x_3^2 + 3.166lx_1^2x_4 + 19.84x_3^2x_3 \]  

Subject to:

\[ g_1(x) = x_4 - 240 \leq 0, \]

\[ 1 \times 0.0625 \leq x_1, \ x_2 < 99 \times 0.0625, \ 10.0 \leq x_3, \text{ and} \]

\[ x_4 \leq 200.0. \]

The structure of the pressure vessel is optimized with IMFO and the results are compared to those of the MFO, GSA, GA, ES, DE, IMFO and ACO algorithms. The results in Table 12 show the superiority of ESMFO with respect to the other algorithms.

### Table 12

| Algorithm | Optimal values for the variables | Optimum cost |
|-----------|----------------------------------|--------------|
| ESMFO     | 0.78246 0.37243 40.21238 177.897865 5867.4375 |              |
| IMFO[50]  | 0.77555 0.38320 40.31962 200.00000 5870.12398 |              |
| MFO[22]   | 0.8125 0.4375 42.098445 176.636596 6059.7143 |              |
| GSA[16]   | 1.1250 0.6250 55.988659 84.4542025 8538.8359 |              |
| PSO[7]    | 0.8125 0.4375 42.09411 176.74050 6061.0777 |              |
| DE[6]     | 0.8125 0.4375 42.098411 176.673690 6059.7340 |              |
| ACO[9]    | 0.8125 0.4375 42.103624 176.572656 6059.0888 |              |
| ES[51]    | 0.8125 0.4375 42.090808 176.580518 6059.7456 |              |
| GA[52]    | 0.8125 0.4375 42.097398 176.654050 6059.9463 |              |

The proposed approach is verified on 21 benchmark functions and compared with the LMFO, HHO, MFO, BA, WOA and CS algorithms. It is clear from the results (comparison between the improved ESMFO algorithm and the other algorithms based on solution quality) that this approach is successful in obtaining optimal results and avoiding local minima. Additionally, there is wide discussion about parameter b, whose explanation is not given in the original paper; this paper clearly describes the importance of b. In this paper, the effect of problems with different numbers of dimensions (i.e., 50, 100, 500 and 1000 dimensions) on MFO, ESMFO and other algorithms is also analyzed. In addition, the pressure vessel design...
problem is examined. It is found that the proposed ESMFO algorithm yields highly competitive results for most of the

5. Conclusions and Future Work
In this paper, by introducing the energy factor from the Harris hawks algorithm, the flame energy of the MFO algorithm is segmented, and the energy-segmented moth-flame optimization algorithm (ESMFO) is proposed. The algorithm is applied to 21 benchmark functions, along with several mainstream heuristic algorithms. The simulation experiment is carried out and the influence of changes in the number of dimensions on the algorithm is considered. The results are statistically tested (Wilkerson value test), and the statistical conclusions that the ESMFO algorithm exhibits high convergence speed and high convergence accuracy are obtained. For future work, there are many ways by which ESMFO can be efficiently improved. These include applying ESMFO in different applications, such as PV parameter estimation, neural network applications, image processing applications, text and data mining applications, big data applications, signal denoising, recourse management applications, network applications, industry and engineering applications, other benchmark test functions, smart home applications, feature selection, image segmentation, task scheduling, and more. The algorithm can also be extended to real-world applications that depend on binary, discrete, and multiobjective optimization. Moreover, the performance of ESMFO can be improved by combining it with Levy flight, disruption, mutation, other stochastic components, local search methods, global search methods, and other evolutionary operators. Additionally, binary and multiobjective versions can be developed to solve practical optimization problems.

Acknowledgements
This work is supported by the National Science Foundation of China under Grant Nos. 62066605 and 61563008 and by the Project of Guangxi Natural Science Foundation under Grant No. 2018GXSNSFA138146.

Compliance With Ethical Standards
Conflict of interest The authors declare that they have no conflicts of interest.

References
[1] Cuevas E, Daz Corts M A, Navarro D A O. Advances of Evolutionary Computation: Methods and Operators. 2016
[2] Fausto F, Reyna-Orta A, Cuevas E, et al. From ants to whales: metaheuristics for all tastes. Artificial Intelligence Review, 2015, 1(11)
[3] Mirjalili S, Lewis A. The Whale Optimization Algorithm. Advances in Engineering Software, 2016, 95:51-67
[4] Faris H, Mafarja M M, Heidari A A, et al. An Efficient Binary Salp Swarm Algorithm with Crossover Scheme for Feature Selection Problems. Knowledge-Based Systems, 2014, 154(2018) 43-67
[5] Koza J R. Genetic Programming II: Automatic Discovery of Reasusable Programs. Artificial Life, 2014, (14)(4):439-441
[6] Storn R, Price K. Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization, 1997, 11(4):341-359
[7] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. MHS'95 Sixth International Symposium on Micro Machine & Human Science. 2002
[8] Sengupta S, Basak S, Peters R A. Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. Machine Learning and Knowledge Extraction, 2019, 1(1): 157-191
[9] Dorigo M, Oca M A M D, Oliveira S, et al. Ant Colony Optimization. 2004
[10] Dorigo M, Stützle T. Ant colony optimization: overview and recent advances. Handbook of metaheuristics. Springer, Cham, 2019: 311-351
[11] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. 2007
[12] Agarwal S K, Yadav S. A comprehensive survey on artificial bee colony algorithm as a frontier in swarm intelligence. Ambient communications and computer systems. Springer, Singapore, 2019: 125-134
[13] Mirjalili S, Lewis A. The Whale Optimization Algorithm. Advances in Engineering Software, 2016, 95:51-67
[14] Gharehchopogh F S, Gholizadeh H. A comprehensive survey: Whale Optimization Algorithm and its applications. Swarm and Evolutionary Computation, 2019, 48: 1-24
[15] Rutenbar R A. Simulated annealing algorithms: an overview. Circuits & Devices Magazine IEEE, 1989, 5(1):19-26
[16] Rashedi E, Nezamabadi-Pour H, Saryazdi S. QSA: A Gravitational Search Algorithm. Information Sciences, 2009, 179(13):2232-2248
[17] Rather S A, Shahid M, Bala P S. A Comprehensive Survey on Solving Clustering and Classification Problems Using Gravitational Search Algorithm. 2019 IEEE 9th International Conference on Advanced Computing (IACC). IEEE, 2019: 13-18
[18] Cuevas E, Echavarria A, Ramirez-Ortegon M A. An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. Applied Intelligence, 2014, 40(2):256-272
[19] Geem Z W, Kim J H, Loganathan G V. A New Heuristic Optimization Algorithm: Harmony Search. Simulation, 2001, 76(2):60-68
[20] Assad A. state-of-the-art review on applications of harmony search metaheuristic algorithm. Technology, 2019, 10(1): 166-178
[21] Rao R V, Savsani V J, Vakharia D P. Teaching–Learning-Based Optimization: An optimization method for continuous non-linear largescale problems. Information Sciences, 2012, 183(1):1-15
[22] Mirjalili S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems, 2015, 89:228-249
[23] Bahrami M., Bozorg-Haddad O., Chu X. Moth-Flame Optimization (MFO) Algorithm. In: Bozorg-Haddad O. (eds) Advanced Optimization by Nature-Inspired Algorithms. Studies in Computational Intelligence, 2018, vol 720. Springer, Singapore
[24] Banaja Mohanty, B.V.S. Acharyulu , P.K. Hota. Moth-flame optimization algorithm optimized dual-mode controller for multiarea hybrid sources AGC system. Optimal Control Applications and Methods. 2018 39(2):720-734
[25] Wolpert D H, Macready W G. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997, 1(1):67–82
[26] L. Lefebvre, P. Whittle, E. Lascaris, A. Finkelstein, Feeding innovations and forebrain size in birds, Animal Behaviour 53 (1997) 549-560
[27] D. Sol, R. P. Duncan, T. M. Blackburn, P. Cassey, L. Lefebvre, Big brains, enhanced cognition, and response of birds to novel environments, Proceedings of the National Academy of Sciences of the United States of America 102 (2005) 5460-5465
[28] F. Dubois, L.-A. Giraldeau, I. M. Hamilton, J. W. Grant, L. Lefebvre, Distraction sneakers decrease the expected level of aggression within groups: a game-theoretic model, The American Naturalist 164 (2004) E32–E45.
[29] Eurek Alert AAAS, Bird iq test takes flight, 2005
[30] Yang X S. A New Metaheuristic Bat-Inspired Algorithm. Computer Knowledge & Technology, 2010, 284:65-74.

[31] Ling Y, Zhou Y, Luo Q. Lévy flight trajectory-based whale optimization algorithm for global optimization. IEEE access, 2017, 5: 6168-6186.

[32] Li Z, Zhou Y, Zhang S, et al. Lévy-flight moth-flame algorithm for function optimization and engineering design problems. Mathematical Problems in Engineering, 2016, 2016.

[33] Luo Q, Ling Y, Zhou Y. Modified Whale Optimization Algorithm for Infinitive Impulse Response System Identification. Arabian Journal for Science and Engineering, 2019: 1-14.

[34] Yang X S. A New Metaheuristic Bat-Inspired Algorithm. Computer Knowledge & Technology, 2010, 284:65-74.

[35] Yang X S, Hossein Gandomi A. Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, 2012, 29(5): 464-483.

[36] Yang X S. Bat algorithm for multi-objective optimisation. arXiv preprint arXiv:1203.6571, 2012.

[37] Yang X S. Bat algorithm: literature review and applications. arXiv preprint arXiv:1308.3900, 2013.

[38] Yang X S, Deb S. Cuckoo Search via Lévy flights. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). IEEE, 2009.

[39] Gandomi A H, Yang X S, Alavi A H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with computers, 2013, 29(1): 17-35.

[40] Walton S, Hassan O, Morgan K, et al. Modified cuckoo search: a new gradient free optimisation algorithm. Chaos, Solitons & Fractals, 2011, 44(9): 710-718.

[41] Yang X S, Deb S. Multiobjective cuckoo search for design optimization. Computers & Operations Research, 2013, 40(6): 1616-1624.

[42] Yang X S. Test problems in optimization. arXiv preprint arXiv:1008.0549, 2010.

[43] Saremi S, Mirjalili S Z, Mirjalili S M. Evolutionary population dynamics and grey wolf optimizer. Neural Computing and Applications, 2015, 26(5): 1257-1263.

[44] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization. Soft Computing, 2019, 23(3): 715-734.

[45] Bingul Z. Adaptive genetic algorithms applied to dynamic multiobjective problems. Applied Soft Computing, 2007, 7(3):791-799.

[46] Bingul Z, Karahan O. Comparison of PID and FOPID controllers tuned by PSO and ABC algorithms for unstable and integrating systems with time delay. Optimal Control Applications and Methods, 2018. 39(4):1431–1450.

[47] Qamar Askari, Mehreen Saeed, Irfan Younas. Heap-based optimizer inspired by corporate rank hierarchy for global optimization. Expert Systems with Applications, 161 (2020) 113702.

[48] Shimin Li, Huiling Chen, Mingjing Wang, Ali Asghar Heidari, Seyedali Mirjalili. slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems, 111 (2020) 300–323.

[49] Inman Ahmadianfar, Omid Bozorg-Haddad, Xuefeng Chu. Gradient-based optimizer: A new metaheuristic optimization algorithm. Information Sciences 540 (2020) 131–159.

[50] Danilo Pelusi, Raffaele Mascella, Luca Tallini, Janmenjoy Nayak, Bighnaraj Naik, Yong Deng. An Improved Moth-Flame Optimization algorithm with hybrid search phase. Knowledge-Based Systems 191 (2020) 105277.

[51] E. Mezura-Montes, C.A.C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (2008) 443–473.

[52] K. Deb. An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Engrg. 186 (2000) 311–338.