Engineering with Computers (2021) 37:2931–2956
https://doi.org/10.1007/s00366-020-00971-7

ORIGINAL ARTICLE

Enhanced a hybrid moth-flame optimization algorithm using new selection schemes

Mohammad Shehab[1] · **Hanadi Alshawabkah**[2] · **Laith Abualigah**[3] · **Nagham AL-Madi**[2]

Received: 13 October 2019 / Accepted: 31 January 2020 / Published online: 21 February 2020

Abstract
This paper presents two levels of enhancing the basic Moth flame optimization (MFO) algorithm. The first step is hybridizing MFO and the local-based algorithm, hill climbing (HC), called MFOHC. The proposed algorithm takes the advantages of HC to speed up the searching, as well as enhancing the learning technique for finding the generation of candidate solutions of basic MFO. The second step is the addition of six popular selection schemes to improve the quality of the selected solution by giving a chance to solve with high fitness value to be chosen and increase the diversity. In both steps of enhancing, thirty benchmark functions and five IEEE CEC 2011 real-world problems are used to evaluate the performance of the proposed versions. In addition, well-known and recent meta-heuristic algorithms are applied to compare with the proposed versions. The experiment results illustrate that the proportional selection scheme with MFOHC, namely (PMFOHC) is outperforming the other proposed versions and algorithms in the literature.

Keywords Moth flame optimization · Hill climbing · Selection schemes · Meta-heuristic algorithms · Real-world problems

1 Introduction

Metaheuristic algorithms are classified into local search-based algorithms and population-based algorithms. Local search-based algorithms consider one solution at a time and try to enhance it using neighbourhood structures [44], such as hill climbing [26], tabu searches [16], β-hill climbing [2], and simulated annealing [25]. While the main advantage of these methods is rapid search speeds, the main drawback is their tendency to focus on exploitation rather than exploration, which, as a result, increases the likelihood of their getting stuck in local optima [43]. By contrast, population-based algorithms, which consider a population of solutions at a time, recombine the current solutions to generate one or more new solutions at each iteration. These methods are effective in identifying promising areas in the search space but are ineffective in exploiting the search space region being explored [45]. Evolutionary computation and swarm intelligence methods are classifications of population-based methods [1]. Both methods are based on the natural biological evolution or social interaction behaviour of natural creatures. Examples of swarm-based algorithms include particle swarm optimization (PSO) [24], krill herd algorithm (KHA) [13], the salp swarm algorithm (SSA) [31] and the moth-flame optimization (MFO) [28].

Swarm intelligence-based methods are inspired by animal societies and social insect colonies [4]. They mimic the behaviour of swarming social insects, schools of fish or flocks of birds. The main advantages of these methods are their flexibility and robustness [8]. MFO is a recent metaheuristic population-based method developed by Mirjalili [28] that imitate the moths' movement technique in the night, called transverse orientation for navigation. Moths fly in the night depending on the moonlight, where they maintain a fixed angle to find their path. The behavior of moths has been formulated as a novel optimization technique. MFO can be utilized to solve a wide range of problems because its procedures are simple, flexible, and easily implemented [21]. On account of these merits, MFO was successfully applied to various optimization problems. For instance, scheduling [12], inverse problem and parameter estimation

✉ Mohammad Shehab
moh.shehab12@gmail.com

1 Computer Science Department, Aqaba University of Technology, Aqaba 77110, Jordan

2 Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, Jordan

3 Faculty of Computer Sciences and Informatics, Amman Arab University, Amman, Jordan

[3, 19], classification [58], economic [51], medical [53], power energy [57], and image processing [10].

As mentioned above, the basic MFO proved its efficiency to solve various problems. However, it suffers from a weak exploitation search, low diversity, and it slows the convergence rate. Therefore, Li et al. [27] applied multi-objective moth-flame optimization algorithm (MOMFA) to improve the efficiency of using water resources. The method assisted and utilized the original moth-flame optimization algorithm, opposition-based learning, and indicator-based selection efficient mechanisms to maintain the diversity and accelerate the convergence. The algorithm tested on the Lushui River Basin and many benchmarks [49]. The algorithm can determine the optimal tradeoff of the elements and can distribute non dominated outcomes for utilization problem of the multi-objective water resources. The result is verified and compared with other algorithms, it indicated to the ability to obtain well pareto solutions for standard problems. Also, Bhesdadiya et al. [6] introduced a hybrid optimization algorithm based on integration between particle swarm optimization (PSO) and MFO. The proposed algorithm is used to solve unconstrained engineering design optimization problems in power system context. MFO is applied to overcome the limitation of PSO algorithm by increasing the exploration search during solving high complex design problem. In the conducted experiment, four benchmarking functions are used to validate the proposed algorithm in terms of exploration and exploitation. Furthermore, the proposed algorithm is compared with the two traditional swarm-based algorithm namely, particle swarm optimization (PSO) and MFO to validate the performance. Overall experiment results illustrate that the performance of the proposed algorithm is better than the compared traditional methods. Moreover, in the context of image segmentation (automated food quality inspection), Sarma et al. [37] proposed a hybrid algorithm combined between physics-based algorithm [e.g., gravitational search algorithm (GSA)] and swarm-based algorithm (i.e., MFO). The proposed algorithm is applied to solve the problem of measuring the degree of food rottenness that cloud helps to minimize monetary losses due to food and storage. Both algorithm is combined because they complete each other. For example, MFO is important due to its effectiveness in exploratory nature. While, GSA is applied due to its effectiveness in team of exploitation. The experiment study is designed to test the hybrid optimization algorithm over thirteen unimodal functions and multimodal functions. Then, the experiment results are used to compare the proposed hybrid algorithm with traditional MFO and GSA algorithms. The comparison results show that the proposed hybrid algorithm is very fast and produces safe results. In [38], the authors proposed a nondominated MFO algorithm (NSMFO) method to solve multi-objective problems. Metaheuristics search techniques are used based on MFO instead of the different optimization techniques like cuckoo search, genetic algorithms, particle swarm optimization, and differential evolutions. The method utilized the crowding distance approach and sorting of the elitist nondominated for preserving the diversity and obtaining variant nondomination levels, respectively, among the optimal set of solutions. It measured the effectiveness by multiobjective benchmark, engineering problems, distinctive feature, and the Pareto front generation [50]. The results of the method were compared with other algorithms and considered closer and better sometimes. While Reddy et al. [35] modified the MFO algorithm (MFOA) and examined characteristics of the local and global search of the basic algorithm. The algorithm is aimed to improve solving unit commitment (UC) problem by using the binary coded modified MFO algorithm (BMMFOA), the basic MFO is a natureinspired heuristic search approach that mimics the traverse navigational properties of moths around artificial lights tricked for natural moonlight, the algorithm used position update of a single-based approach between corresponding flame and the moth differently than many other swarm based approaches. The modified MFO algorithm (MMFOA) is used to improve the exploitation search of the moths and reduces the number of flames.

This paper highlights the two main weaknesses recognized in the performance trajectory of the basic version of the MFO: loss of the solutions' diversity, which leads to a slow convergence manner. Because of these weaknesses, MFO requires further refinements, to be modified or hybridized with other algorithms components or local search techniques. As a result, an improved method, by hybridizing the basic MFO and hill-climbing (HC) search strategy called MFOCH. Moreover, using several promising selection schemes for enhancing the quality of the selected solutions. The following points are summarized the main contributions of this work.

1. A new hybridization method using MFO and HC (MFOCH) is developed to improve the exploitation search.
2. Alternative selection methods in the MFOCH for global optimization problems are investigated to maintain the diversity of the solutions, as well as improving their quality.
3. The performance of the proposed algorithms is tested using thirty basic benchmarks and five IEEE CEC 2011 real world problems.

The organization of this review is as follows. Section 2 introduces MFO, HC, and the selection schemes. Then, The proposed methods are described in Sect. 3. Section 4 shows experimental results and discussions. Finally, Sect. 5 presents conclusions and future directions.

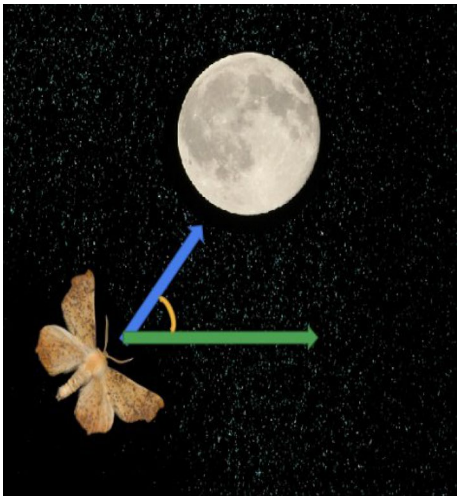


Fig. 1 Moth's transverse orientation

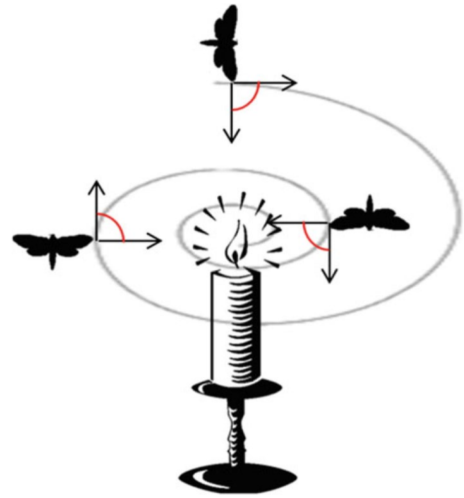


Fig. 2 Moth's spiral flying path around a light source [46]

2 Preliminaries

2.1 Moth-flame optimization algorithm

2.1.1 Origin

In nature, over 160,000 different species of moths have been documented, which are resemble butterflies in their life cycle (i.e., moth consists of two-level life: larvae and adult, where it converted to moth by cocoons) [48].

Interesting thing in the moths' life is their special navigation methods at night. They have evolved to fly in the night using the moonlight. Also, they employed a mechanism called transverse orientation for navigation. This mechanism allows the moth to fly by preserving a stable angle with respect to the moon, a very effective mechanism for travelling long distances in a straight path [14]. Figure 1 illustrates a conceptual model of transverse orientation. Since the moon is far away from the moth, this mechanism guarantees flying in a straight line. The same navigation method can be done by humans. Suppose that the moon is in the south side of the sky and a human wants to go the east. If he keeps moon of his left side when walking, he would be able to move toward the east on a straight line.

It can be observed in Fig. 2 the moths do not travel in a forward path, they fly spirally around lights. This is due to the transverse orientation method which is efficient just for the light source is very far (moonlight). In the human-made artificial light case, the moths attempt to preserve the same angle with the light source. Consequently, moths move in spirally paths around lights.

Table 1 Characteristic of the MFO algorithm

Algorithm's description	Moth-Flame's elements
Decision variable	Moth's position in each dimension
Solutions	Moth's position
Initial solutions	Random positions of moths
Current solutions	Current positions of moths
New solutions	New positions of moths
Best solution	Flame's position
Fitness function	Distance between moth and flame
Process of generating new solution	Flying in a spiral path toward a flame

2.1.2 MFO algorithm

Moth-Flame optimization (MFO) algorithm was proposed by Mirjalili [28]. It is under the population-based metaheuristic algorithms. The flow data of the MFO starts by generating moths randomly within the solution space. Then, calculating the fitness values (i.e., position) of each moth and tagging the best position by flame. After that, updating the moths' positions depends on a spiral movement function to achieve better positions tagged by a flame, as well as updating the new best individual positions. Repeating the previous processes (i.e., updating the moths' positions and generating new positions) until the termination criteria are met. Table 1 lists the characteristics of the MFO.

The MFO algorithm has three main steps. These steps as shown below. Followed by the pseudocode of the MFO as shown in Algorithm 1.

1. *Generating the initial population of Moths:*
   As mentioned in [28], Mirjalili assumed that each moth can fly in 1-D, 2-D, 3-D, or hyperdimensional space. The set of moths can be expressed:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & \cdots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & \cdots & m_{n,d} \end{bmatrix} \tag{1}$$

   where n refers to the moths' number and d refers to the dimensions's number in the solution space. Also, the fitness values for all moths are memorized in an array as follows:

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{2}$$

   The rest elements in the MFO algorithm are flames. The following matrix showing the flames in the D-dimensional space followed by their fitness function vector:

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & \cdots & F_{1,d} \\ F_{2,1} & F_{2,2} & \cdots & \cdots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & F_{n,2} & \cdots & \cdots & F_{n,d} \end{bmatrix} \tag{3}$$

$$\mathrm{OF} = \begin{bmatrix} \mathrm{OF}_1 \\ \mathrm{OF}_2 \\ \vdots \\ \mathrm{OF}_n \end{bmatrix} \tag{4}$$

   It should be noted here that moths and flames are both solutions. The difference between them is the way we treat and update them in each iteration. The moths are actual search agents that move around the search space, whereas flames are the best position of moths that are obtained so far. In other words, flames can be considered as flags or pins that are dropped by moths when searching the search space. Therefore, each moth searches around a flag (flame) and updates it in case of finding a better solution. With this mechanism, a moth never loses its best solution.
2. *Updating the Moths' positions:*
   MFO employs three different functions to convergent the global-optimal of the optimization problems. These functions are defined as follows:

$$\mathrm{MFO} = (I, P, T) \tag{5}$$

   where I refers to the first random locations of the moths ($I : \phi \to \{M, OM\}$), P refers to motion the moths in the search space($P : M \to M$), and T refers to finish the search process ($T : M \to$ true,false). The following equation represents I function, which use for implementing the random distribution.

$$M(i,j) = (\mathrm{ub}(i) - \mathrm{lb}(j)) \times \mathrm{rand}() + \mathrm{lb}(i) \tag{6}$$

   where lb and ub indicate the lower and upper bounds of variables, respectively. As mentioned previously, the moths fly in the search space using the transverse orientation. There are three conditions should abide when utilizing a logarithmic spiral subjected, as follows:

   - Spiral's initial point should start from the moth.
   - Spiral's final point should be the position of the flame.
   - Fluctuation of the range of spiral should not exceed the search space.

   Therefore, the logarithmic spiral for the MFO algorithm can be defined as follows:

$$S(M_i, F_j) = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j \tag{7}$$

   where D_i refers to the space between the i-th moth and the *j-th* flame (see the Eq. (8). b indicates a fix to define the shape of the logarithmic spiral, and t indicates a random number between [− 1, 1].

$$D_i = \left|F_j - M_i\right|. \tag{8}$$

   In MFO, the balancing between exploitation and exploration are guaranteed by the spiral motion of the moth near the flame in the search space. Also, to avoid falling in the traps of the local optima, the optimal solutions have been kept in each repetition, and the moths fly around the flames (i.e., each moths flies surrounding the nearest flame) using the OF and OM matrices.
3. *Updating the number of flames:*
   This section highlights enhancing the exploitation of the MFO algorithm (i.e., Updating the moths' positions in n various locations in the search space may decrease a chance of exploitation of the best promising solutions). Therefore, decreasing the number of flames helps to solve this issue based on the following equation:

$$\text{flame no} = \mathrm{round}\left(N - l \times \frac{N - l}{T}\right) \tag{9}$$

where N is the maximum number of flames, l is the current number of iteration, and T indicates the maximum number of iterations.

Algorithm 1 Moth-flame optimization algorithm

```
Initialize the parameters for Moth-flame
Initialize Moth position M_i randomly
for i = 1 to n do
   Calculate the fitness function f_i
end for
while itration ⩽ Max_iterations do
   Update the position of M_i
   Calculate the number of flames using Eq.(9)
   Evaluate the fitness function f_i
   if iteration ==1 then
      F=sort(M) and OF=sort(OM)
   else
      F=sort(M_{t-1}, M_t) and OF=sort(M_{t-1}, M_t)
   end if
   for i = 1 to n do
      for j = 1 to d do
         Update the values of r and t
         Calculate the value of D respect to its corresponding moth using Eq.(8)
         Update M(i,j)respect to its corresponding moth using Eq.(7)
      end for
   end for
end while
Print the best solution
```

2.2 Hill climbing

The hill climbing (HC) technique, called local search, is the most simplistic form of local development methods. It begins with one random initial solution (x), iteratively proceeds by moving from the current solution to a better neighboring solution till it reaches a local optimum (i.e., the local optimal solution does not have a better neighboring solution, no improvement in fitness function). It only takes downhill progress where the fitness function of a neighboring solution should be better than the current solution Shehab [41]. Consequently, it can converge to the local optima fast and suddenly. However, it can quickly get stuck in local optima, which in most situations is not satisfactory. Algorithm 2 presents the pseudo-code of the HC technique. After creating the first solution x and through the iterative improvement process, a group of neighboring solutions is created utilizing the procedure *Improve*(N(x)). This procedure seeks to discover the enhanced neighboring solution from the group of neighbors utilizing any used acceptance rule such as first improvement, best improvement, sidewalk, and random walk. But, all of these rules are stopped in local optima.

2.3 Selection schemes

In this section, the selection schemes are described that used in this paper.

2.3.1 Tournament selection scheme (TSS)

Tournament selection is among the most popular selection methods in genetic algorithms. It was initially proposed by Goldberg and Holland [17]. Algorithm 3 shows the principle of tournament selection work, which starts from the random selection of t individuals from $P(t)$ population and then proceeds to the selection of the best individual from tournament t. This procedure is repeated N times. The best choice is frequently between two individuals, and this scheme is called binary tournament, where the choice is between t individuals called tournament size [7]. In other words, the efficiency of tournament selection scheme is based on the value of t. For instance, increasing the value of t will increase the diversity which leads to an increase in the quality of the selected solution, and vice versa [47].

Algorithm 2 Hill climbing technique

```
The initial solution x
x_i=LB_i+ (UB_i-LB_i) * U(0,1), ∀_i=(1, 2, ...., N)
Calculate fitness function F(x)
while (End condition is not satisfied) do
   x'=Improve((N(x))
   if F(x') ≤ F(x) then
      x=x'
   end if
end while
return x
```

Algorithm 3 Tournament Selection Scheme

Input: The population $P(_T)$ the tournament size $t \in i1, 2, ..., N$
Output: The population after selection $P(_T)'$

Description:

tournament $(J_1, ..., J_N)$:
for $i \leftarrow 1$ *to* N *do*
  $J_i' \leftarrow$ best fit individual out of t randomly piked individuals from $\{J_1, ..., J_N\}$;
endfor
return $\{J_1', ..., J_N'\}$

There are several merits of the tournament selection scheme. For instance, low susceptibility to a takeover by dominant individuals [33], it has efficient time complexity (i.e., O(n)) [40], and no requirement for fitness scaling or sorting [32].

2.3.2 Proportional selection scheme (PSS)

The proportional selection scheme or so-called roulette wheel has been proposed in [20]. In other words, each element reserves a section in the roulette wheel, where the section's size proportional with the element's fitness. The mechanism of this method is choosing the probability based on the comparison between the fitness values of any solution and the fitness value of the stored solution in MFO. As shown in algorithm 4, r has been selected from U(0,1). Then, s_i has accumulative determining the probabilities, the following equation shows the probability of solution x.

$$P_i = \frac{f(x_i)}{\sum_{j=1}^{\text{swarm size}} f(x_j)}. \tag{10}$$

The advantage of proportional selection, it offers a chance for each element to be chosen. In contrast, in population converges, it suffers from selection pressure [40]. The time complexity of the proportional selection is O(n log n).

2.3.3 Linear ranking selection scheme (LRSS)

To overcome the limitation of the proportional selection scheme, Goldberg and Holland [17] proposed Linear ranking selection scheme. It arranges the solutions based on their fitness ranks. Equation (11) shows the mechanism of calculation the selection probability by linear mapping of the solution ranks.

$$P_i = \frac{1}{N} \times \left(\eta^+ - \left(\eta^+ - \eta^-\right) \times \frac{i-1}{N-1}\right), \quad i \in 1, \ldots, N, \tag{11}$$

where i is the rank of solution location x^j, η^- is the expected value of the worst location, η^+ is the expected value of the best location. Both of η^- and η^+ set the slope of the linear function. More details are shown in Algorithm 5.

Algorithm 4 Proportional Selection Scheme

Input: The population $P(_T)$, $r \in U(0,1)$
Output: The population after selection $P(_T)'$

Description:

proportional $(J_1, ..., J_N)$:
$s_0 \leftarrow 0$
for $i \leftarrow 1$ *to* N *do*
  $s_i \leftarrow s_{i-1} + P_i$
endfor
for $i \leftarrow 1$ *to* N *do*
  $r \leftarrow random\,[0, s_N]$
  $J_i' \leftarrow J_i such that s_{i-1} \leqslant r < s_i$
endfor
return $\{J_1', ..., J_N'\}$

Algorithm 5 Linear Ranking Selection Scheme

Input: The population $P(_T)$ and the reproduction rate of the worst individual $\eta- \in [1,0]$
Output: The population after selection $P(_T)'$

Description:

$linear\_ranking(J_1, ..., J_N)$:
$\bar{J} \leftarrow sorted\ population\ \ J\ \ according\ fitness\ with\ worst\ individual\ at\ the\ first\ position$
$s_0 \leftarrow 0$
for $i \leftarrow 1\ to\ N\ do$
$s_i \leftarrow s_{i-1} + p_i$ $(Equation\ 11)$
endfor
for $i \leftarrow 1\ to\ N\ do$
$r \leftarrow random\,[0, s_N]$
$J'_i \leftarrow \bar{J}_i such that s_{i-1} \leqslant r < s_i$
endfor
return $\{J'_1, ..., J'_N\}$

The expected results of the linear ranking selection scheme with small η^+ are close to the binary tournament selection. However, the linear ranking selection scheme with big η^+ suffers from a stronger selection pressure (i.e., the time complexity is is O(n log n)) [34].

2.3.4 Exponential ranking selection scheme (ERSS)

Unlike linear ranking completely, exponential ranking selection arranging the probabilities of the ranked elements by exponentially weighted [42]. The major of the exponent c which is situated between (0, 1), where it based on parameter s. For instance, the best solution has a value of $c_1 = 1$, followed by the second solution with $c_2 = s$ (s = 0.99), the third solution has $c_3 = s^2$, and so on until the worst solution has $c_{\text{swarm size}} = s^{\text{swarm size}-1}$ [18]. Probabilities of the individuals calculated by

$$p_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}}; \quad i \in \{1, 2, \ldots, N\} \tag{12}$$

The $\sum_{j-1}^{N} c^{N-j}$ normalizes the probabilities to ensure that $\sum_{i=1}^{N} c^{N-j} p_i = 1$. As
$\sum_{j-1}^{N} c^{N-j} = \frac{c^N-1}{C-1}$ it will be as a following equation:

$$p_i = \frac{c-1}{c^N - 1} C^{N-i}; \quad i \in \{1, 2, \ldots, N\} \tag{13}$$

Algorithm 6 illustrates the exponential ranking selection algorithm, the similarity of structure between linear ranking selection and exponential ranking selection can be noticed. While the difference lies in the calculation of the selection probabilities. The time complexity of the exponential ranking selection is O(n log n).

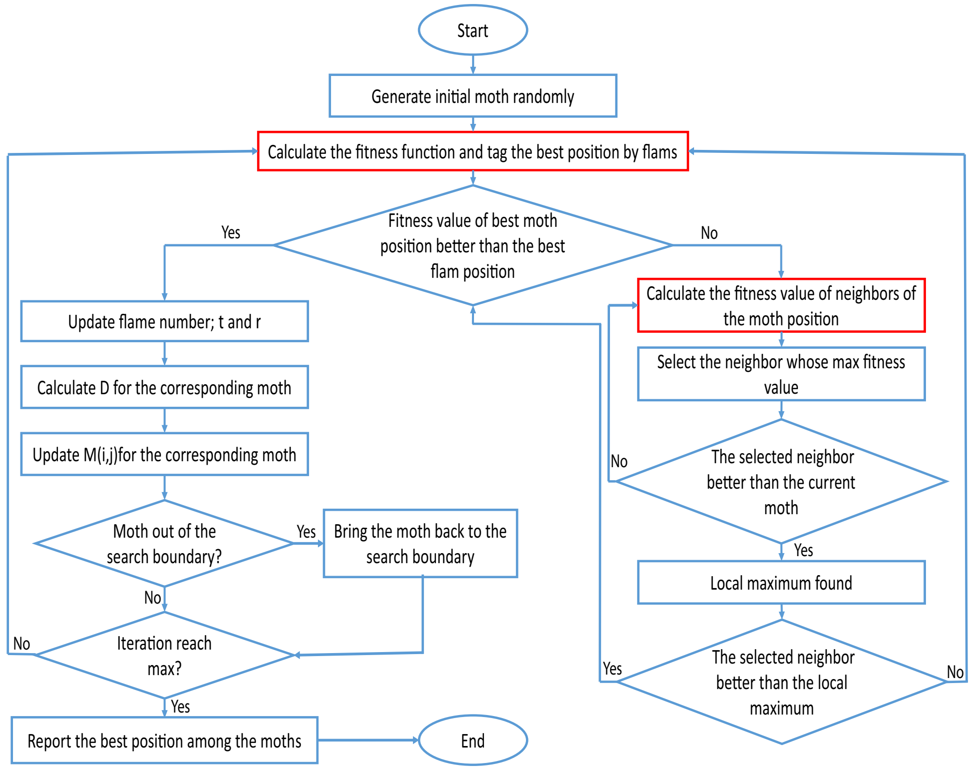
Algorithm 6 Exponential Ranking Selection Scheme

Input: The population $P(_T)$ and the ranking base $c \in [1,0]$
Output: The population after selection $P(_T)'$

Description:

$exponential\_\ ranking(c, J_1, ..., J_N)$:
$\bar{J} \leftarrow sorted\ population\ \ J\ \ according\ fitness\ with\ worst\ individual\ at\ the\ first\ position$
$s_0 \leftarrow 0$
for $i \leftarrow 1\ to\ N\ do$
$s_i \leftarrow s_{i-1} + p_i$ $(Eq.13)$
endfor
for $i \leftarrow 1\ to\ N\ do$
$r \leftarrow random\,[0, s_N]$
$J'_i \leftarrow \bar{J}_i such that s_{i-1} \leqslant r < s_i$
endfor
return $\{J'_1, ..., J'_N\}$

Fig. 3 Flowchart of the MFOHC algorithm



2.3.5 Greedy-based selection scheme (GSS)

The greedy selection scheme is called global best which was initially applied by Kennedy [23] in PSO. The technicality of greedy selection focuses to choose the three best solutions: x_α, x_β, and x_γ to avoid the local optima. Algorithm 7 shows the pseudo-code of the greedy selection scheme.

As mentioned above, the greedy choose the best three solutions and ignored the other solutions. Therefore, the diversity of the search space might be lost which leads to prematurely converge and quickly stagnate without efficient results. The time complexity of the greedy selection scheme is O(n log n).

Algorithm 7 Greedy-based Selection Scheme

Input: The population $P(_T)$
Output: The population after selection $P(_T)'$

Description:

```
for j ← 1 to |J| do
for t ← 1 to |T| do
    w_jt ← 0
endfor
endfor
for i ← 1 to |I| do
    Obtain a new patrol using DP and let a* = a*_jt be the obtained optimal patrol
    Assign patrol a* to team i
foreach (j,t) with a* = a*_jt = 1 do
    w_jt ← 1
endfor
endfor
return {J'_1, ..., J'_N}
```

2.3.6 Truncation selection scheme (TrSS)

Truncation selection is considered as the simplest selection scheme comparing with other selection schemes. The truncation chooses elements by saving a certain percentage until reaching the population size [39]. This selection is equal to (μ, λ) -selection utilized in development strategies with $T = \frac{\mu}{\lambda}$ [5].

Table 2 The parameters values of the comparative algorithms

Algorithms	Parameter
ABC	Colony size = 50, limit = 1000
BA	$F_{\min} = 0, F_{\max} = 2, r = 0.5, A = 0.25, \alpha, \gamma = 0.9$
DE	CR = 0.9, $F = 0.6$
GA	Crossover type is 1; crossover probability = 1; mutation probability = 0.01
HS	HMCR = 0.9, PAR = 0.5, BW = 0.01
KH	$N^{\max} = 0.01, V_f = 0.02, D^{\max} = 0.005, C_t = 0.4$
GWO	$\alpha_0 = 2$

Algorithm 8 Truncation Selection Scheme

Input: The population $P(_T)$, the truncation threshold $T \in [0, 1]$
Output: The population after selection $P(_T)'$
Description:
Truncation $(T, J_1, ..., J_N)$:
$\bar{j}$ ←sorted population J according fitness with worst individual at the first posit
for $i \leftarrow 1$ *to* N *do*
$r \leftarrow random\{[(1-T)N], \cdots, N\}$
$j'_i \leftarrow \bar{j}_r$
endfor
return $\{J'_1, ..., J'_N\}$

From Truncation's pseudo-code, it can be noticed that the ease of implementation of this selection. However, it neglects the solutions with a low fitness value which have an ability to improve into better solutions. This may lead to premature convergence. As a sorting of the population is required, truncation selection has a time complexity of O (n ln n).

3 The proposed methods

This section presents two new methods for improving basic MFO.

3.1 Hybrid Moth-flam optimization algorithm and hill climbing

The first improvement is hybridized basic MFO and HC (i.e., MFOHC) to enhance the exploitation mechanism as well as the convergence rate. As shown in Fig. 3, the flowchart of MFOHC starts by generating initial moth randomly, then calculating the moths' fitness function and determining the best flam's position. The usage of the HC components start in case the output of the first condition is "No". In other words, if the fitness value of the selected moth is worse than the value of the best flam position, then it should search for another moth with better fitness value using the exploitation mechanism of the HC. After that, the selected solution will be compared again with the best flame position. The rest steps are similar to the basic MFO, such as updating the flam, calculating the distance between the moth and the updated flam, etc.

3.1.1 Computational complexity

Note that, the computational complexity for running the proposed MFOHC algorithm is depended on the number of salp solutions (X), the dimensions (d), and the maximum number of repetitions (t). Hence, the computational complexity of sorting procedure in each iteration is $O(t \times n^2)$ in the worst case. The computational complexity of the initialization procedure is $O(n)$. Updating the positions of all search agents is $O(t \times n \times d)$. Therefore, the computational complexity of the basic MFO is $O(n \log n)$ and $O(n^2)$ in the best and worst case, where n denotes the number of moths. Moreover, the time complexity to determine if the hill-climbing process has reach a local optimum is $O(n^3)$. Therefore, the final complexity of the MFOHC is $O(T \times n^3(n^2 + n \times v))$, where T is the maximum number of iterations and v is the number of variables. Thus, the time complexity of each MOFHC's version can be fined by adding the time complexity of each selection scheme as mentioned breviously.

Table 3 Description of unimodal benchmark functions

No.	Function	Equation	Range	f_{min}
f1	Beale	$f_1(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	[−4.5,4.5]	0
f2	Watson	$f_2(x) = \sum_{i=0}^{29}\left\{\sum_{j=0}^{4}((j-1)\alpha_i^j x_{j+1}) - \left[\sum_{j=0}^{5}\alpha_i^j x_{j+1}\right]^2 - 1\right\}^2 + x_1^2$	[−5,5]	0.002288
f3	Dixon and price	$f_3(x) = (x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$	[−10,10]	0
f4	Quartic with noise	$f_4(x) = \sum_{i=1}^{30} ix^4 + \text{random}[0, 1)$	[−1.28,1.28]	0
f5	Schwefel 1.2	$f_5(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	[−100,100]	0
f6	Schwefel 2.22	$f_6(x) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	[−100,100]	0
f7	Schwefel 2.21	$f_7(x) = \sum_{i=1}^{n}\lvert x_i\rvert$	[−100,100]	0
f8	Sphere	$f_8(x) = \sum_{i=1}^{d} x_i^2$	[−5.12,5.12]	0
f9	Step	$f_9(x) = \sum_{i=1}^{n}\lfloor x_i^2\rfloor$	[−100,100]	0
f10	Zakharov	$f_{10}(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4$	[−5,10]	0

Table 4 Description of multimodal benchmark functions

No.	Function	Equation	Range	$f_{\min}$
f11	Easom	$f_{11}(x) = -\cos(x_1)\cos(x_2)exp(-(x_1 - \pi)^2 - (x_{2-\pi})^2)$	[−100,100]	0
f12	Shubert	$f_{12}(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$	[−10,10]	−186.7309
f13	Wolfe	$f_{13}(x) = \frac{3}{4}(x_1^2 + x_2^2 - x_1 \cdot x_2)^{0.75} + x_3$	[0,2]	0
f14	Colville	$f_{14}(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4) + 10.1((x_2 - 1)^2)$ $+19.8(x_2 - 1)(x_4 - 1)$	[−10,10]	0
f15	Ackley	$f_{15}(x) = -\alpha\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + \alpha + \exp(1))^{**}$	[−32.768,32.768]	0
f16	Griewank	$f_{16}(x) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[−600,600]	0
f17	Levy	$f_{17}(x) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10\sin^2(\pi w_i + 1)] + (w_d - 1)^2[1 + \sin^2(2\pi w_d)]^{**}$	[−10,10]	0
f18	Perm	$f_{18}(x) = \sum_{i=1}^{d}\left(\sum_{j=1}^{d}(j+\beta)\left(x_j^i - \frac{1}{j^i}\right)\right)^2$	[−d,d]	0
f19	Rastrigin	$f_{19}(x) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	[−5.12,5.12]	0
f20	Rosenbrock	$f_{20}(x) = \sum_{i=1}^{d-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	[−5,10]	0
f21	Egg Holder	$f_{21}(x) = -(x_2 + 47)\sin\left(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert}\right) - x_1\sin\left(\sqrt{\lvert x_1 - (x_2 + 47)\rvert}\right)$	[−5.12,5.12]	−959.6407
f22	Michalewicz	$f_{22}(x) = -\sum_{i=1}^{d}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right), m = 10$	[0,π]	−1.8013

**In f_{15}, $\alpha = 20$, b = 0.2, and c = 2π

In f_{17}, $w_i = 1 + \frac{x_i - 1}{4}$

For instance, the time complexity of PMFOHC is estimated as $O(T \times n^3(n^2 + n \times v)) + O(n\log n)$.

3.2 Improved MFOHC using various selection schemes

The second improvement is using new selection schemes to enhance the quality of the selected solution, as well as diversity. Six selection schemes have been chosen based on their features. For instance, TSS has the time complexity $O(N)$ and diversity is inversely proportional to the t size. While PSS provides a probability for each solution to be selected based on their proportions. LRSS and ERSS focus on improving the convergence rate. GSS gives priority to the global search with avoiding the local optima. Finally, in TrSS, the worst six solutions (i.e., worst fitness values)

Table 5 Description of fixed-dimension multimodal benchmark functions

No.	Function	Equation	Range	f_{min}
f23	Branin	$f_{23}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	[$x_1 \in [-5, 10], x_2 \in [0, 15]$]	0.397887
f24	Goldstein Price	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times[30 + (2x_1 - 3x_2) \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	[−2,2]	3
f25	Hartman 1	$f_{25}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{4} \alpha_{ij}(x_j - p_{ij})^2\right]$	[−1,3]	−3.86
f26	Hartman 2	$f_{26}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} \alpha_{ij}(x_j - p_{ij})^2\right]$	[0,1]	−3.32
f27	Kowalik	$f_{27}(x) = \sum_{i=0}^{10}\left[\alpha_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^{2}$**	[−5,5]	0.0003074861
f28	Shekel 1	$f_{28}(x) = -\sum_{i=1}^{5}\left[(x - \alpha_i)(x - \alpha_i)^T + c_i\right]^{-1}$	[0,10]	−10.1532
f29	Shekel 2	$f_{29}(x) = -\sum_{i=1}^{7}\left[(x - \alpha_i)(x - \alpha_i)^T + c_i\right]^{-1}$	[0,10]	−10.4028
f30	Shekel 3	$f_{30}(x) = -\sum_{i=1}^{10}\left[(x - \alpha_i)(x - \alpha_i)^T + c_i\right]^{-1}$	[0,10]	−10.5363

**$\alpha = [4, 2, 1, 1/2, 1/4, 1/8, 1/10, 1/12, 1/14, 1/16]$

$b = [0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246]$

will never be neglected, thus it will speed up the search processes.

In Fig. 3, the red rectangles show the locations of using each one of the selection schemes. In other words, the enhancing of the MFOHC using the selection schemes is presented after generating the population, where the selection schemes aid to select the best solution to compare it with the best flame. While the second location of using the selection scheme in the local search part, where it replace the basic selection in HC (i.e., random selection).

4 Simulations

4.1 Experiments settings

- *Normalization measure* is the process of regularizing data with respect to the difference in values between samples. In the experiments, the effects of different values of the dimensions and the search agents are compared with one another. This procedure is difficult due to the wide gap between solutions. Therefore, normalization improves data integrity [52]. In this work, normalization is calculated based on the following equation:

$$z_i = \frac{x_i - \mu}{S}, \tag{14}$$

where is $x = (x_1, \ldots, x_n)$, n denotes the total number of data, z_i denotes the normalized data for element ith, μ is the mean and S is the standard deviation. Finally, the minimum element of the data will be 1 in the normalization results.

- *The best measure* is utilized to calculate the best-obtained value by the algorithm to be evaluated for several predefined numbers of runs, which can be measured as follows:

$$\text{Best} = \min_{1 \le i \le N_r} F_i^* \tag{15}$$

where, N_r denoted to the number of various runs and F_i^* denoted to the best-obtained value.

- *The average measure (avg)* is utilized to calculate the mean of the best-obtained values by the algorithm to be evaluated for several predefined numbers of runs, which can be measured as follows:

$$\mu_F = \frac{1}{N_r}\sum_{i=1}^{N_r} F_i^* \tag{16}$$

- *The standard deviation (std)* is a measure utilized to test if the algorithm to be evaluated can obtain the same best value in several various runs and examine the repeatability test of the algorithm results, which can be measured as follows:

$$\text{STD}_F = \sqrt{\frac{1}{N_r - 1}\sum_{i=1}^{N_r}(F_i - \mu_F)^2} \tag{17}$$

Also, convergence trajectories are shown to display the behavior of the comparative algorithms to give the optimal value. Note, the parameters settings of the comparative algorithms are shown in Table 2.

There are two levels of evaluation performed in this work. The first step is evaluating the performance of the HMFO using a set of benchmark functions (see Sect. 4.1.1). The second step is applying the HMFO versions Using a set of IEEE CEC 2011 real world problems (see Sect. 4.1.2). All the experiments run using Matlab R2015a and Windows 7 Professional, Intel(R) Core(TM) i5-4590 CPU @ 3.30 GHz with a memory of 6.00 GB.

4.1.1 Benchmark functions

The proposed MFOHC method is verified based on using 30 classical benchmark test functions listed in tables 3, 4, and 5. This well-knowing benchmarks include 30 test functions, which are classified in to unimodal (it means optimization functions with only one local optimum) and multimodal (it means optimization functions that frequently contain multiple global and local optima) problems. Moreover, these functions are chosen with various dimensions and diverse difficulty levels including 10 scalable unimodal functions, 12 scalable multimodal functions, and 8 fixed-dimension multimodal functions. These features make the investigation process more fitting for testing the exploration and exploitation functions in the proposed method.

4.1.2 IEEE CEC 2011 real world problems

This subsection describes seven real-world problems that used in CEC 2011, more details can be found in [9]. These problems are utilized to evaluate the performance of HMFO versions.

1. *CEC-P1*: Static economic load dispatch (ELD) Problem

   This problem (i.e., static ELD) is focused on minimizing the fuel cost of producing units in a specific period, which is usually set by one hour. Thus, determining the optimal production dispatch during the operating units, as well as keeping the system load demand. The objective function is based on the non-smooth cost and smooth cost functions, more details are shown below:

$$\text{Minimizing}: \; F = \sum_{i=1}^{N_G} F_i(P_i), \tag{18}$$

   where

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i, \; i = 1, 2, 3, \ldots, N_G, \tag{19}$$

   where $F_i(P_i)$ refers to the cost function and a_i, b_i, and c_i indicate to its cost coefficient. $N_(G)$ refers to the number of online producing units and P_i the real power output in a time t. The following equation shows the cost function for the unit with valve point loading influence.

$$F_i(P_i) = a_i P_i^2 + b_i P_i + c_i + \left| e_i \sin\left(f_i\left(P_i^{\min} - P_i\right)\right)\right|, \tag{20}$$

   where f_i and e_i indicate to the cost coefficients identical to the valve point loading influence.

2. *CEC-P2*: Optimal control of a non-linear stirred tank reactor

   In the chemical area, the chemical reaction proceeds in the continuous stirred tank reactor (CSTR) which can be included under the multimodal optimization problem. Thus, it can be used to evaluate the performance of the metaheuristic algorithms, exactly like the standard benchmark functions. The following equations illustrate the mathematical model of this problem.

$$\dot{x_1} = -(2 + u)(x_1 + 0.25) + (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right), \tag{21}$$

$$\dot{x_2} = 0.5 - x_2 - (x_2 + 0.5)\exp\left(\frac{25x_1}{x_1 + 2}\right), \tag{22}$$

   where u refers to the flow rate of the cooling fluid, x_1 and x_2 indicate to state temperature and deviation, respectively. The objective function is determined by an appropriate value of u to enhance the performance index, the following equation shows the calculation process.

$$J = \int_0^{t_{f=0.72}} (x_1^2 + x_2^2 + 0.1u^2)\mathrm{d}t. \tag{23}$$

3. *CEC-P3*: Large scale transmission pricing problem

   In modern power systems, the estimation price of the transmission considers a controversial problem [9]. The estimation price is based on various take-holders. Thus, it depends on different factors. The equivalent bilateral exchange (EBE) is one of the common factors (linearized model) used to estimate the price of the transmission. EBA creates a matrix of the load-generation interaction, the following equation illustrates the total of equivalent bilateral exchange.

$$\mathrm{GD}_{ij} = \frac{P_{Gi} P_{Dj}}{P_D^{\mathrm{sys}}}, \tag{24}$$

   where i and j refer to generator and load, respectively. P_D^{sys} refers to the total load. While Eq. (25) represents the portion of power flow *pf* inline k, which used (i.e., pf_k) to examine the all equivalent dual power exchanges.

$$pf_k = \sum_i \sum_j \left|\gamma_{ij}^k\right| \mathrm{GD}_{ij}. \tag{25}$$

Table 6 The best normalized results for MFO with population sizes

Function	Population sizes							
	5	10	15	20	50	100	250	500
F_1	1.49E+01	9.91E+02	**1.00**	5.94E+02	**1.00**	5.41E+01	1.93E+02	1.17E+02
F_2	2.56E+00	**1.00**	**1.00**	2.60E+01	5.16E+01	5.23E+01	7.13E+00	**1.00**
F_3	1.17E+00	1.04E+00	**1.00**	1.04E+00	1.04E+00	1.04E+00	**1.00**	1.04E+00
F_4	1.93E+00	2.19E+01	3.70E+00	**1.00**	4.90E+01	2.41E+01	8.37E+00	1.12E+01
F_5	2.36E+01	6.65E+01	**1.00**	2.16E+01	3.22E+01	1.74E+01	6.89E+00	2.39E+01
F_6	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_7	1.90E+00	**1.00**	1.04E+00	1.24E+00	1.14E+00	1.23E+00	1.04E+00	1.09E+00
F_8	2.68E+01	3.80E+00	**1.00**	5.96E+00	1.62E+01	1.01E+01	1.20E+01	1.09E+00
F_9	5.68E+00	1.24E+00	**1.00**	1.10E+00	2.55E+00	1.04E+00	2.57E+00	2.27E+00
F_{10}	1.86E+00	**1.00**	2.87E+01	6.39E+00	2.58E+01	1.33E+01	1.05E+01	1.86E+00
F_{11}	**1.00**	2.53E+00	1.05E+00	5.00E+00	5.34E+00	2.28E+00	7.58E+00	2.85E+00
F_{12}	**1.00**	**1.00**	**1.00**	**1.00**	1.29E+00	**1.00**	1.32E+00	8.20E+00
F_{13}	1.66E+01	9.32E+01	**1.00**	1.08E+02	1.09E+02	2.83E+01	2.49E+01	1.83E+00
F_{14}	1.49E+00	1.28E+00	1.10E+01	**1.00**	1.12E+01	2.23E+01	3.87E+01	1.02E+01
F_{15}	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_{16}	1.15E+00	**1.00**	1.16E+00	1.20E+00	1.17E+00	1.58E+00	1.53E+00	1.23E+00
F_{17}	**1.00**	**1.00**	1.02E+00	1.11E+00	1.18E+00	2.23E+00	1.60E+00	1.42E+00
F_{18}	1.93E+00	1.81E+00	**1.00**	1.87E+00	1.66E+00	2.69E+00	2.32E+00	2.54E+00
F_{19}	**1.00**	**1.00**	1.52E+01	1.02E+02	7.62E+01	7.62E+00	2.62E+01	2.26E+01
F_{20}	**1.00**	1.04E+00	1.01E+00	1.06E+00	1.02E+00	1.01E+00	1.02E+00	1.02E+00
F_{21}	**1.00**	1.29E+00	**1.00**	1.24E+00	1.21E+00	1.14E+00	1.29E+00	1.28E+00
F_{22}	3.43E+00	2.79E+00	**1.00**	2.01E+00	1.24E+01	1.42E+02	1.15E+01	1.16E+01
F_{23}	1.27+00	1.32E+00	1.32E+00	**1.00**	1.35E+00	1.43E+00	1.22E+00	1.21E+00
F_{24}	1.19E+00	**1.00**	**1.00**	1.10E+00	1.02E+00	1.11E+00	7.58E+00	1.25E+00
F_{25}	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_{26}	**1.00**	**1.00**	**1.00**	1.02E+00	**1.00**	1.02E+00	1.03E+00	1.04E+00
F_{27}	1.38E+00	2.71E+00	**1.00**	1.07E+00	1.55E+01	1.21E+00	1.33E+00	1.01E+00
F_{28}	**1.00**	**1.00**	1.49E+00	1.16E+00	1.02E+00	1.60E+00	2.23E+00	1.44E+01
F_{29}	**1.00**	**1.00**	1.01E+00	**1.00**	1.01E+00	1.01E+00	1.29E+00	1.06E+00
F_{30}	1.03E+00	1.03E+00	**1.00**	2.04E+00	6.81E+00	1.60E+00	1.43E+00	1.01E+00
Total best	**12**	**14**	**17**	**8**	**5**	**4**	**4**	**4**

4. *CEC-P4*: Hydrothermal scheduling problem

   Hydrothermal scheduling is divided into long term (i.e., from week(s) to months) and short term (i.e., 24 h and less) problems. This problem aims to schedule the power generations of the thermal and hydro units in a fixed period of time and minimum fuel cost. However, the hydrothermal system is very complicated and includes nonlinear connections of the resolution variables, water carry retards, and time connection among the consecutive schedules. So, detecting the minimum fuel cost is so difficult by utilizing the basic optimization algorithms.

   The main objective to achieve the maximum results of the hydro units, at the same time each unit consumed the lowest load. The description of the objective function is expressed below.

$$F = \sum_{i=1}^{M} f_i(P_{Ti}), \tag{26}$$

where M refers to the number of intervals. In Eq. (27), the f_i indicates to the cost function connected with the identical thermal unit's power producer P_{Ti} :

$$f_i(P_{Ti}) = a_i P_{Ti}^2 + b_i P_{Ti} + c_i + \left| e_i \sin\left(f_i\left(P_{Ti}^{\min} - P_{Ti}\right)\right)\right|. \tag{27}$$

5. *CEC-P5*: Spread spectrum radar polly phase code design

   waveform is considered as one of the most important factors in designing radar-system which is based on pulse compression. Various studies have been proposed for polyphase pulse compression code synthesis, especially those depending on the characteristic of the aperiodic

Table 7 The best normalized results for MFO with different dimensional spaces

Function	Dimensional spaces							
	5	10	15	20	25	30	35	40
F_1	**1.00**	3.35E+00	5.18E+00	1.16E+01	2.82E+01	1.73E+01	3.01E+01	1.88E+01
F_2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_3	**1.00**	6.91E+00	1.64E+01	8.59E+00	4.22E+01	8.58E+00	3.38E+01	1.42E+01
F_4	**1.00**	1.17E+00	1.06E+00	1.23E+01	1.15E+00	1.02E+00	1.11E+00	1.08E+00
F_5	3.31E+01	5.11E+00	**1.00**	7.01E+00	5.17E+00	1.06E+00	4.23E+00	1.22E+01
F_6	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_7	7.63E+00	**1.00**	5.63E+00	1.40E+00	1.94E+00	6.99E+00	1.13E+01	2.55E+00
F_8	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_9	1.06E+01	1.06E+01	**1.00**	1.06E+01	1.06E+01	1.06E+01	1.06E+01	1.06E+01
F_{10}	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_{11}	**1.00**	1.10E+01	6.24E+00	3.67E+01	3.46E+00	2.18E+00	2.18E+00	5.15E+00
F_{12}	2.02E+00	**1.00**	4.86E+00	7.52E+00	1.12E+00	6.82E+00	2.54E+00	1.13E+01
F_{13}	**1.00**	1.46E+01	2.93E+01	1.08E+01	1.21E+01	3.97E+01	1.27+01	4.47E+00
F_{14}	1.21E+00	**1.00**	1.28E+00	1.29E+00	1.25E+00	1.40E+00	1.05E+00	1.19E+00
F_{15}	9.21E+00	**1.00**	1.32E+00	4.54E+00	5.25E+00	1.57E+00	8.78E+00	4.42E+00
F_{16}	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_{17}	**1.00**	4.28E+00	3.77E+00	4.69E+00	4.93E+00	1.48E+00	9.72E+00	2.40E+00
F_{18}	1.08E+00	**1.00**	1.41E+00	1.32E+00	1.09E+00	1.15E+00	3.52E+03	3.01E+00
F_{19}	2.74E+00	**1.00**	3.90E+01	1.97E+00	1.96E+00	2.49E+01	5.31E+01	1.01E+01
F_{20}	**1.00**	6.83E+00	5.80E+01	3.08E+01	4.92E+01	2.59E+00	4.71E+00	1.65E+00
F_{21}	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F_{22}	4.00E+00	**1.00**	6.99E+00	9.67E+00	2.09E+00	4.91E+00	2.51E+00	1.74E+00
F_{23}	3.06E+00	**1.00**	4.92E+01	4.42E+01	4.97E+00	4.74E+00	1.37E+01	1.64E+01
F_{24}	**1.00**	4.22E+01	5.24E+01	8.63E+01	9.09E+00	7.58E+00	1.31E+01	3.94E+01
F_{25}	**1.00**	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00	1.32E+00
F_{26}	1.04E+00	**1.00**	1.74E+00	1.43E+00	1.34E+00	1.23E+00	1.12E+00	1.17E+00
F_{27}	1.24E+00	**1.00**	1.70E+00	1.74E+00	1.55E+00	1.34E+00	1.51E+00	1.73E+00
F_{28}	**1.00**	1.24E+00	1.16E+00	1.19E+00	1.13E+00	1.16E+00	1.08E+00	1.11E+00
F_{29}	1.30E+00	1.23E+00	**1.00**	1.10E+00	1.24E+00	1.10E+00	1.02E+00	1.33E+00
F_{30}	**1.00**	1.03E+00	1.01E+00	1.01E+00	1.01E+00	1.02E+00	1.01E+00	1.02E+00
Total best	**17**	**16**	**9**	**6**	**6**	**6**	**6**	**6**

autocorrelation function. Thus, CEC-P5 can be treated like a continuous optimization problem. The mathematical model is described in the following equations.

$$\text{global} \min_{x \in X} f(x) = \max\left\{\phi_1(x), \dots, \phi_{2m}(x)\right\}, \tag{28}$$

where $X = \left\{(x_1, \dots, x_n) \in R^n \mid 0 \le x_j \le 2\pi, j = 1, \dots, n\right\}$ and $m = 2n - 1$

$$\phi_{2i-1}(x) = \sum_{j=i}^{n} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right),\ i = 1, \dots, n \tag{29}$$

$$\phi_{2i}(x) = 0.5 + \sum_{j=i+1}^{n} \cos\left(\sum_{k=|2i-j|+1}^{j} x_k\right),\ i = 1, \dots, n-1 \tag{30}$$

$$\phi_{m+i}(x) = -\phi_i(x),\ i = 1, \dots, m \tag{31}$$

4.2 Results and discussions

4.2.1 Influence of control parameter

The experiments start with evaluating the parameter settings of the MFO to set them in subsequent experiments. It can be noticed that the parameters tuning include experiments of the population size (n) with a set of common values to

Table 8 Best, average (Avg), and standard deviation (Std) for comparing the proposed MFOHC with basic MFO and other algorithms

Function	Metric	Comparative algorithms									
		ABC	BA	SSA	DE	GA	HS	KH	GWO	MFO	MFOHC
F_{1}	Best	3.28E−05	1.48E−01	1.15E−01	7.77E−01	2.43E−05	4.27E+00	5.31E−02	3.13E−02	3.00E−03	1.37E−07
	Avg	2.99E−01	8.46E+00	2.82E+00	9.36E+00	1.21E−03	2.02E+01	5.29E+00	2.38E+00	4.33E−02	2.00E−03
	Std	1.07E−16	5.15E−01	5.68E+00	2.75E+02	1.08E−05	8.36E−01	2.37E+00	2.14E−02	4.36E−01	4.67E−03
F_{2}	Best	4.89E−06	2.56E−01	2.20E−02	7.17E−03	4.09E−06	2.15E−01	1.22E−01	3.53E−01	3.69E−03	4.56E−07
	Avg	2.85E−01	7.31E+00	1.49E+00	4.94E−01	1.55E−03	1.02E+00	1.03E+00	6.17E+00	4.39E−01	1.86E−03
	Std	9.85E−17	7.35E−01	2.99E−01	1.90E+04	2.41E−04	5.70E+00	1.74E+01	1.47E−01	4.03E+00	7.09E−04
F_{3}	Best	9.98E−01	2.31E+00	3.75E+00	1.03E+00	1.41E+00	9.98E−01	9.98E−01	2.99E+00	2.47E−01	4.67E−05
	Avg	8.49E+00	9.14E+01	2.46E+01	6.87E+01	1.58E+01	1.62E+01	4.06E+00	1.20E+00	2.55E+00	6.31E−02
	Std	0.00E+00	2.19E+00	3.30E+00	1.81E−01	5.54E−01	7.15E−07	3.63E−16	3.42E+00	9.32E−01	2.01E−02
F_{4}	Best	3.42E−04	1.08E−03	4.25E−03	1.30E−03	7.24E−04	4.35E−04	5.33E−03	1.71E−03	6.33E−04	1.89E−07
	Avg	1.42E−01	1.97E+00	4.68E−01	1.27E+00	3.52E−01	5.56E−01	8.40E+00	2.07E−01	3.57E−01	1.33E−03
	Std	1.67E−04	4.81E−04	7.33E−03	4.44E−04	5.82E−05	9.62E−05	8.08E−03	5.08E−03	7.59E−02	2.86E−02
F_{5}	Best	4.25E−04	2.68E−03	3.88E−03	1.18E−04	1.74E−02	5.77E−03	3.74E−02	2.65E−01	1.39E−03	4.88E−06
	Avg	5.70E+00	5.70E+00	3.00E+00	3.00E+00	3.00E+00	1.48E+00	1.48E+00	2.25E+00	2.17E−01	1.25E−03
	Std	5.66E−04	1.46E+07	1.11E+03	7.07E+04	3.47E+02	3.08E+03	3.88E+02	8.72E−01	3.12E−01	6.78E−02
F_{6}	Best	1.05E−07	3.30E−02	1.37E−01	8.56E−02	2.69E−03	1.72E−02	1.45E−02	4.38E−03	3.12E−06	4.12E−09
	Avg	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01	1.88E−14	1.62E+00	5.63E−01	2.14E−03	9.01E−04
	Std	3.76E−15	1.81E+03	2.20E+00	4.29E+02	8.98E−04	5.30E+01	1.83E+00	3.09E−01	7.64E−02	4.36E−03
F_{7}	Best	6.81E−05	1.74E+00	1.01E−01	2.49E−01	4.56E−03	1.36E−01	1.42E−02	6.54E−04	4.01E−05	7.03E−09
	Avg	7.24E−02	4.35E+01	5.33E+00	1.71E+00	8.92E−01	5.82E+00	9.62E+00	8.08E−01	6.31E−02	1.01E−04
	Std	8.69E−05	3.51E+00	8.11E+00	1.57E−01	1.74E−03	6.65E−02	6.85E−03	4.22E−04	7.09E−02	4.18E−03
F_{8}	Best	0.00E+00	1.55E−06	2.58E−07	1.49E−12	0.00E+00	1.16E−12	8.69E−09	2.87E−11	0.00E+00	0.00E+00
	Avg	1.22E−85	3.53E−01	5.44E−03	6.98E−04	1.05E−52	1.74E−03	2.24E−04	1.92E−02	2.08E−088	0.00E+00
	Std	0.00E+00	3.83E−01	2.23E−01	3.20E−01	9.23E−14	2.09E−01	2.89E−01	1.57E+00	0.00E+00	0.00E+00
F_{9}	Best	8.88E−14	1.37E−01	4.75E+00	7.83E+00	1.32E−16	9.49E+00	3.50E+00	1.39E−14	3.14E−14	1.05E−16
	Avg	1.87E−04	3.07E+00	6.13E+01	1.02E+01	1.96E−06	3.57E+01	5.03E+01	1.58E−04	6.37E−06	1.88E−09
	Std	0.00E+00	8.38E+00	2.56E+00	1.37E+00	2.00E−03	1.35E+00	1.11E+00	2.69E+00	6.42E−04	2.78E−09
F_{10}	Best	0.00E+00	1.20E−21	5.53E−12	6.07E−17	0.00E+00	2.50E−11	5.39E−09	1.62E−13	0.00E+00	0.00E+00
	Avg	1.48E−120	5.59E−02	4.48E−03	1.01E−01	1.67E−82	5.54E−02	4.65E−02	9.99E−03	3.02E−89	0.00E+00
	Std	0.00E+00	3.12E−01	6.28E−02	2.80E+00	0.00E+00	5.16E−01	1.27E−01	4.32E−03	0.00E+00	0.00E+00
F_{11}	Best	1.28E−02	9.36E−06	9.50E−01	1.73E−02	2.91E−02	6.06E−02	1.42E−01	2.00E−01	0.00E+00	0.00E+00
	Avg	1.03E+00	1.03E+00	1.03E+00	1.03E+00	1.03E+00	1.85E+00	3.90E+00	2.14E+00	1.02E−45	0.00E+00
	Std	3.40E−02	1.35E−06	9.61E+01	5.59E+01	1.12E+03	3.15E+02	1.49E+00	4.07E+03	0.00E+00	0.00E+00
F_{12}	Best	5.32E−06	2.60E−12	6.96E+00	2.20E−01	1.19E−01	1.24E+01	1.67E+01	3.20E+01	7.09E−17	3.64E−21
	Avg	4.72E+00	9.65E+00	3.20E+00	8.48E+00	1.05E+00	2.70E+00	1.55E+00	2.04E+00	6.31E−07	6.31E−08
	Std	8.50E−06	2.83E−12	1.71E+00	7.90E+00	6.88E+00	5.77E+00	1.22E+00	2.15E+01	1.02E−07	0.00E+00
F_{13}	Best	3.69E+01	1.82E−17	1.56E+00	3.26E+00	6.44E+01	7.33E−01	5.20E−01	1.52E−01	0.00E+00	0.00E+00
	Avg	3.27E+00	3.30E−02	2.89E+01	3.24E+01	3.25E+00	6.04E+00	4.90E+00	3.32E+00	5.74E−12	0.00E+00
	Std	4.63E+03	6.04E−17	9.16E+02	1.01E+02	3.02E+02	5.49E+03	1.17E+01	1.05E+04	0.00E+00	0.00E+00
F_{14}	Best	3.17E−02	6.05E−01	1.71E+00	1.91E−01	5.66E−02	1.63E−01	9.11E+00	3.29E−16	3.20E−16	4.15E−19
	Avg	3.80E+00	3.86E+00	3.85E+01	3.86E+00	3.86E+00	2.09E+00	1.65E+01	2.17E−03	6.47E−09	5.12E−10
	Std	4.29E+00	1.15E+01	2.26E−01	7.20E+00	1.37E−02	2.20E+00	3.62E+00	4.14E−01	4.97E−04	4.01E−10
F_{15}	Best	1.03E−02	3.25E−02	5.66E−02	6.33E−03	8.21E−02	6.01E−02	4.43E−03	6.87E−02	2.58E−03	3.97E−08
	Avg	2.71E+00	2.39E+00	6.83E−01	2.69E+00	2.80E+00	4.90E+00	1.86E+01	5.96E−01	3.33E−01	1.08E−03
	Std	6.78E−02	6.78E−02	3.94E−01	9.32E−02	1.44E−01	1.94E−01	1.85E−01	3.90E−01	4.04E−01	1.82E−03
F_{16}	Best	3.98E−01	6.03E−01	2.21E−02	7.21E−01	8.01E−01	1.15E+00	7.91E−01	3.62E−02	3.25E−04	6.85E−09
	Avg	1.75E+00	5.56E+00	9.88E−01	1.09E+00	4.59E+00	5.92E+01	3.71E+00	4.19E−01	1.02E−01	1.35E−05
	Std	7.30E+00	2.04E+00	2.72E−01	5.42E−01	3.35E+00	3.01E+00	1.88E−01	1.62E−01	3.42E−01	4.01E−05

Table 8 (continued)

Function	Metric	Comparative algorithms									
		ABC	BA	SSA	DE	GA	HS	KH	GWO	MFO	MFOHC
F_{17}	Best	3.00E+00	4.85E−01	1.12E−02	5.23E+00	1.09E+00	6.36E+00	7.89E−01	1.12E−02	4.66E−02	1.92E−07
	Avg	2.37E+01	6.91E+00	1.34E+00	6.96E+01	4.55E+01	2.09E+02	9.54E+00	8.35E−01	1.01E+00	6.83E−04
	Std	2.18E+00	2.19E−01	2.69E−02	9.93E+00	1.36E+00	5.13E+00	1.48E+01	4.33E+01	4.29E−01	2.91E−03
F_{18}	Best	3.86E−01	7.55E+00	2.46E−03	5.79E+00	8.24E−02	6.42E−01	7.68E−06	2.99E−01	4.32E−06	4.78E−09
	Avg	3.40E+00	2.86E+01	2.06E−01	4.82E+01	4.44E+00	4.35E+00	2.73E−01	3.60E+00	5.68E−02	2.03E−05
	Std	2.65E−01	2.71E+00	1.02E−02	4.00E+00	3.67E−01	1.42E−01	2.09E−02	1.65E−01	1.97E−01	7.11E−04
F_{19}	Best	3.27E+00	3.23E+00	2.90E+00	3.27E+00	3.30E+00	3.32E+00	3.27E+00	3.30E+00	1.03E+00	6.52E−05
	Avg	3.30E+02	7.39E+01	1.51E+01	1.44E+01	9.51E+01	2.50E+02	1.01E+01	1.24E+01	3.22E+00	7.89E−02
	Std	6.03E−02	5.63E−02	1.20E−01	7.60E−02	4.84E−02	3.39E−05	6.04E−02	4.90E−02	4.66E−01	7.96E−04
F_{20}	Best	4.59E−01	2.60E−09	1.01E−02	7.66E−02	8.79E−06	1.86E−12	1.84E−01	3.45E−11	4.22E−12	3.26E−16
	Avg	1.02E+01	6.81E−02	5.14E+00	7.78E+00	8.66E−01	1.02E−01	4.72E+00	9.65E−02	7.09E−06	6.35E−09
	Std	6.96E+00	3.50E+00	2.76E+00	2.56E+00	3.04E+00	1.05E−02	2.70E+00	1.55E+00	1.89E−02	2.71E−07
F_{21}	Best	4.42E+00	1.05E−08	1.10E−02	2.34E−01	7.50E+00	4.52E−12	1.27E−02	3.10E−11	2.05E−12	7.82E−15
	Avg	1.04E+01	7.45E−01	6.19E+00	8.34E+00	1.01E+01	1.04E−03	6.34E+00	1.02E−01	3.75E−06	3.01E−09
	Std	1.19E+00	3.30E+00	3.16E+00	2.79E+00	1.40E+00	1.66E−04	3.69E+00	9.70E−01	7.66E−03	5.21E−08
F_{22}	Best	1.05E−01	8.48E+00	5.24E+00	8.30E+00	1.05E−01	1.05E−01	6.94E+00	1.00E−01	1.05E−01	7.93E−07
	Avg	1.60E+00	1.13E+02	2.68E+01	3.62E+02	2.87E+01	4.43E+01	1.34E+02	6.38E+00	4.67E+00	2.22E−04
	Std	1.75E+00	3.26E+00	2.99E+00	2.81E+00	7.38E+00	8.24E−01	3.94E+00	1.98E+00	5.82E+00	4.63E−04
F_{23}	Best	9.29E+00	9.69E+00	1.58E+00	3.73E+00	1.04E+01	4.86E+00	1.04E+01	1.04E+01	4.32E+00	1.02E−06
	Avg	9.57E+02	2.37E+01	2.40E+01	3.12E+01	3.29E+02	3.30E+01	3.46E+01	3.76E+02	3.11E+01	7.88E−04
	Std	2.58E+00	1.84E+00	3.02E+00	8.49E−01	1.07E+00	1.21E+00	7.00E+01	6.46E+00	3.85E+00	5.74E−04
F_{24}	Best	9.22E+00	8.99E+00	9.73E+00	3.65E+00	1.05E−01	5.28E+00	1.05E−01	1.05E−01	4.36E−02	2.78E−06
	Avg	1.37E+01	5.79E+01	3.19E+02	7.99E+01	8.80E+00	1.96E+01	4.42E+01	2.28E+01	7.35E+00	4.00E−04
	Std	2.74E+00	2.93E+00	2.14E+00	7.74E+01	3.17E−01	1.25E+00	9.32E+00	6.92E+01	8.02E−01	6.37E−03
F_{25}	Best	3.86E+00	3.85E−02	5.54E+00	2.91E−01	6.61E−01	4.44E+00	8.63E−01	7.99E−01	1.99E−02	2.46E−07
	Avg	4.63E+03	8.26E+01	8.13E+03	7.95E+00	6.16E+00	4.04E+03	6.29E+01	2.57E+01	6.97E−01	7.32E−04
	Std	1.36E+00	2.17E+00	6.63E+00	2.73E+00	2.71E+00	8.17E+00	2.26E+00	3.82E+00	2.08E−01	3.75E−04
F_{26}	Best	3.24E+00	3.25E+00	3.26E+00	1.73E+00	3.32E+00	2.83E+00	3.32E+00	3.29E+00	4.89E+00	3.55E−03
	Avg	3.69E+03	1.11E+02	1.47E+04	2.41E+01	7.32E+02	2.48E+02	3.67E+02	1.22E+01	7.06E+01	1.02E−01
	Std	5.74E−02	8.74E−02	6.05E−02	4.12E−01	7.73E−04	2.25E−01	1.23E−03	1.31E−02	2.34E+00	6.78E−01
F_{27}	Best	8.50E−01	7.02E−01	2.77E−16	7.69E−02	1.18E−05	1.30E−08	4.42E−01	5.79E−07	3.57E−13	1.59E−16
	Avg	8.48E+00	9.22E+00	7.63E−03	3.92E+00	1.02E−01	4.83E−02	1.02E+01	1.01E−02	2.64E−07	1.35E−09
	Std	2.90E+00	2.15E+00	2.81E+00	7.89E−01	1.18E−02	1.35E+00	4.91E−03	6.97E−03	7.65E−06	4.86E−08
F_{28}	Best	1.03E+00	1.03E+00	1.03E+00	1.72E−01	1.03E+00	1.03E+00	1.03E+00	1.03E+00	7.36E−01	1.02E−04
	Avg	5.32E+01	6.01E+01	6.60E+01	3.33E+00	2.64E+02	4.40E+01	7.93E+01	1.11E+01	2.22E+00	7.13E−02
	Std	7.01E+00	8.16E+00	8.62E+00	1.62E+00	6.78E+00	1.16E+00	5.98E+00	1.03E+00	1.01E+00	5.92E−02
F_{29}	Best	3.40E−02	2.22E−09	5.10E−02	2.03E−09	2.00E−12	1.70E−01	5.16E−11	4.55E−03	0.00E+00	0.00E+00
	Avg	3.98E+00	3.98E−01	3.98E−01	1.37E−02	3.98E−03	3.98E+00	3.98E−03	3.98E−01	0.00E+00	0.00E+00
	Std	2.21E−02	4.58E−07	2.46E−02	9.41E−04	4.93E−02	5.42E−01	2.48E−03	5.33E−01	0.00E+00	0.00E+00
F_{30}	Best	6.98E−02	1.05E−01	5.05E−02	5.78E−01	2.38E−12	2.14E−01	6.39E−11	9.64E−02	5.78E−12	8.64E−17
	Avg	1.28E−01	9.21E+00	5.80E+00	2.31E+00	2.32E−03	3.82E+00	1.22E−02	4.29E−01	7.02E−07	2.07E−09
	Std	3.41E−01	9.04E−02	3.49E−02	8.29E−02	4.63E−12	3.89E+00	8.99E−11	8.47E−01	4.33E−06	1.64E−09

determine the optimal value. After that, repeat the experiments with the selected value of the population size (n) and different common values for the dimension (D) to find its optimal value. Thus, the best values of the n and D will be uses in the rest of the experiments.

Table 9 Best, average (Avg), and standard deviation (Std) for comparing the MFOHC versions with other algorithms

Function	Metric	Comparative algorithms									
		LGCMFO	CLSGMFO	MFODE	OMFO	GMFOHC	TMFOHC	TrMFOHC	PMFOHC	LRMFOHC	ERMFOHC
F_1	Best	···	···	···	···	4.77E−10	1.11E−09	4.58E−10	**1.26E−10**	3.98E−10	2.02E−09
	Avg	···	···	···	···	3.36E−07	3.10E−07	4.32E−07	1.03E−07	5.24E−07	3.12E−07
	Std	···	···	···	···	4.20E−07	3.05E−07	1.46E−06	4.57E−07	1.75E−07	3.21E−07
F_2	Best	···	···	···	···	4.31E−06	1.79E−07	5.14E−07	**7.29E−08**	1.05E−07	5.66E−07
	Avg	···	···	···	···	1.95E−05	4.43E−05	7.63E−05	6.40E−05	5.37E−05	3.60E−05
	Std	···	···	···	···	2.97E−05	5.00E−05	1.51E−04	1.05E−04	1.03E−04	1.03E−04
F_3	Best	···	···	···	···	4.02E−06	1.09E−06	4.50E−06	4.61E−07	**3.06E−07**	5.99E−07
	Avg	···	···	···	···	2.83E−04	3.22E−04	3.05E−04	4.20E−04	3.30E−04	2.89E−04
	Std	···	···	···	···	4.04E−04	4.04E−04	4.24E−04	5.02E−04	5.21E−04	3.83E−04
F_4	Best	···	···	1.63E−01	···	1.00E−07	9.63E−06	**1.00E−09**	1.70E−08	2.48E−06	2.20E−06
	Avg	1.88E−04	6.81E -05	···	2.72E−04	7.80E−03	8.00E−03	1.11E−06	3.67E−05	8.30E−03	7.52E−03
	Std	1.53E−04	8.69E -05	3.80E−04	2.86E−02	2.10E−03	4.38E−02	5.87E−06	1.76E−05	1.56E−03	2.18E−03
F_5	Best	···	···	···	···	6.24E−08	5.72E−07	1.10E−06	**2.60E−08**	4.34E−08	6.17E−07
	Avg	···	···	···	···	7.39E−06	5.22E−04	3.02E−04	4.55E−03	7.60E−05	1.09E−04
	Std	···	···	···	···	7.52E−02	5.78E−02	1.89E−06	5.15E−02	3.98E−02	6.15E−02
F_6	Best	···	···	···	···	**7.02E−13**	4.82E−10	2.55E−09	7.16E−11	3.54E−10	1.54E−11
	Avg	···	···	···	···	4.37E−08	8.29E−09	7.24E−08	5.32E−08	1.81E−09	4.26E−11
	Std	···	···	···	···	2.07E−06	4.33E−06	3.99E−08	2.53E−06	1.08E−06	1.75E−05
F_7	Best	···	···	**0.00E+00**	···	2.11E−10	5.72E−11	2.91E−10	2.77E−11	2.81E−11	2.77E−10
	Avg	1.90E−193	3.17E−282	···	3.26E−11	3.09E−07	3.03E−06	3.01E−06	3.02E−06	3.02E−08	3.02E−05
	Std	0.00E+00	0.00E+00	0.00E+00	4.18E−03	2.50E−02	3.65E−02	3.84E−02	4.43E−03	3.52E−02	3.87E−04
F_8	Best	···	···	**0.00E+00**	···	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Avg	0.00E+00	0.00E+00	···	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_9	Best	···	···	**1.64E−22**	···	8.85E−17	7.42E−17	7.08E−17	5.41E−17	5.78E−17	5.22E−17
	Avg	5.40E−06	1.05E−15	···	1.41E−02	7.14E−12	5.25E−11	7.14E−12	3.11E−10	2.47E−11	6.06E−14
	Std	2.89E−06	3.76E−15	3.32E−22	2.78E−09	4.70E−12	2.91E−12	1.08E−12	4.35E−10	4.29E−11	2.14E−14
F_{10}	Best	···	···	**0.00E+00**	···	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Avg	1.32E−200	2.43E−295	···	1.80E−08	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	4.36E−03	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{11}	Best	···	···	**0.00E+00**	···	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Avg	0.00E+00	0.00E+00	···	3.10E−09	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	6.78E−02	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Table 9 (continued)

Function	Metric	Comparative algorithms									
		LGCMFO	CLSGMFO	MFODE	OMFO	GMFOHC	TMFOHC	TrMFOHC	PMFOHC	LRMFOHC	ERMFOHC
F_{12}	Best	···	···	···	···	8.38E−22	4.43E−22	6.42E−26	**9.68E−27**	3.17E−25	4.16E−22
	Avg	···	···	···	···	2.04E−09	1.64E−08	2.14E−09	2.17E−11	1.52E−10	1.62E−09
	Std	···	···	···	···	1.11E−07	1.04E−07	1.45E−08	1.30E−11	8.98E−09	8.63E−09
F_{13}	Best	···	···	···	···	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Avg	···	···	···	···	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	···	···	···	···	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{14}	Best	···	···	···	···	8.24E−26	8.56E−24	9.89E−25	**0.00E+00**	6.60E−27	8.35E−25
	Avg	···	···	···	···	7.59E−18	5.61E−20	6.61E−20	0.00E+00	5.21E−22	9.98E−20
	Std	···	···	···	···	1.24E−19	3.86E−17	3.57E−18	0.00E+00	4.68E−15	5.79E−17
F_{15}	Best	···	···	8.88E−12	···	1.16E−10	3.22E−09	4.30E−10	**1.81E−12**	2.88E−12	5.31E−12
	Avg	6.45E−16	3.22E -16	···	8.88E−10	3.22E−04	1.08E−03	3.30E−04	3.00E−04	3.81E−04	3.31E−04
	Std	0.00E+00	0.00E+00	0.00E+00	1.82E−03	3.83E−04	4.04E−04	4.58E−04	5.12E−04	5.02E−04	4.99E−04
F_{16}	Best	···	···	8.88E−16	···	2.15E−13	1.21E−13	7.47E−12	**1.07E−16**	6.95E−16	1.82E−12
	Avg	6.82E−06	4.89E -17	···	4.21E−11	3.82E−10	1.17E−10	4.10E−08	5.84E−09	9.91E−09	6.58E−11
	Std	1.99E−01	9.85E -17	0.00E+00	4.01E−05	7.83E−06	2.02E−06	5.81E−06	1.38E−07	1.79E−07	1.19E−06
F_{17}	Best	···	···	2.32E−03	···	9.03E−09	4.32E−08	1.53E−09	2.84E−09	**1.13E−10**	5.43E−09
	Avg	2.88E−07	2.37E−07	···	···	1.17E−06	1.07E−06	9.86E−06	7.54E−06	2.91E−06	1.03E−04
	Std	4.85E−07	1.07E−06	4.32E−03	···	9.90E−04	9.84E−04	1.11E−06	7.41E−06	3.57E−06	5.87E−05
F_{18}	Best	···	···	···	···	2.92E−15	2.27E−14	2.42E−12	**2.33E−16**	4.48E−16	2.41E−15
	Avg	···	···	···	···	8.06E−08	7.28E−09	5.83E−09	6.73E−09	5.23E−11	5.91E−09
	Std	···	···	···	···	4.55E−06	2.65E−06	7.63E−06	1.70E−08	2.77E−06	7.67E−07
F_{19}	Best	···	···	1.77E−05	···	1.98E−10	2.01E−10	2.00E−11	2.02E−10	2.06E−12	**1.93E−12**
	Avg	1.50E−04	1.03E+00	···	4.29E−07	3.20E−06	6.33E−06	4.50E−06	2.54E−09	7.02E−07	4.22E−07
	Std	1.50E−04	6.78E -16	8.47E−05	7.96E−04	6.28E−04	4.54E−04	4.79E−04	4.94E−04	4.25E−03	5.60E−04
F_{20}	Best	···	···	2.87E−01	···	4.07E−20	1.38E−20	1.02E−18	**7.96E−22**	7.97E−20	7.96E−19
	Avg	2.50E−01	4.25E−04	···	2.70E−01	9.59E−12	1.14E−10	1.10E−11	1.80E−14	1.80E−10	1.80E−09
	Std	2.33E−01	5.66E -04	3.27E−02	2.71E−07	1.70E−09	3.08E−08	2.68E−08	6.94E−07	5.38E−09	6.99E−08
F_{21}	Best	···	···	···	···	3.18E−24	3.07E−24	4.23E−19	1.82E−25	**2.77E−27**	5.98E−26
	Avg	···	···	···	···	4.49E−10	2.62E−10	1.35E−12	1.77E−11	4.96E−13	3.01E−10
	Std	···	···	···	···	2.42E−10	2.47E−10	5.65E−10	2.49E−11	2.33E−10	3.52E−11
F_{22}	Best	···	···	···	···	3.26E−08	2.25E−08	3.21E−09	**4.49E−10**	4.68E−09	2.84E−08
	Avg	···	···	···	···	1.23E−05	1.21E−05	8.18E−05	1.11E−05	7.93E−07	1.34E−06
	Std	···	···	···	···	1.27E−04	9.09E−04	7.46E−04	1.15E−05	6.70E−05	5.37E−05

Table 9 (continued)

Function	Metric	Comparative algorithms									
		LGCMFO	CLSGMFO	MFODE	OMFO	GMFOHC	TMFOHC	TrMFOHC	PMFOHC	LRMFOHC	ERMFOHC
F_{23}	Best	⋯	⋯	3.98E−01	⋯	7.14E−08	3.60E−08	**4.27E−10**	7.27E−10	8.56E−09	4.05E−08
	Avg	3.98E−01	3.98E−01	⋯	⋯	6.54E−07	8.06E−07	7.10E−06	9.45E−06	7.81E−07	6.71E−08
	Std	0.00E+00	0.00E+00	0.00E+00	⋯	9.08E−05	9.76E−05	9.88E−05	1.18E−05	1.21E−05	8.77E−05
F_{24}	Best	⋯	⋯	2.32E−03	⋯	8.74E−08	1.66E−09	1.45E−08	**1.06E−09**	3.62E−08	8.53E−07
	Avg	1.88E−07	3.98E−01	⋯	3.00E−03	4.15E−08	8.65E−08	7.96E−07	5.04E−08	2.16E−08	3.50E−07
	Std	1.03E−15	2.18E−15	5.98E−20	6.37E−03	2.64E−05	3.80E−05	8.50E−05	5.36E−05	6.37E−05	1.48E−04
F_{25}	Best	⋯	⋯	2.56E−08	⋯	6.73E−09	5.82E−09	2.16E−11	8.32E−11	**1.53E−12**	4.62E−09
	Avg	3.86E−04	3.86E−04	⋯	⋯	4.46E−07	4.47E−07	4.45E−07	4.46E−07	4.47E−07	4.44E−07
	Std	2.67E−15	2.65E−15	3.73E−22	⋯	6.19E−05	2.99E−05	5.26E−05	6.21E−06	3.57E−05	2.12E−05
F_{26}	Best	⋯	⋯	2.32E−03	⋯	1.32E−04	**1.28E−04**	1.32E−04	1.31E−04	1.31E−04	1.32E−04
	Avg	4.57E−04	9.64E−02	⋯	1.02E−01	1.75E−02	1.39E−02	2.00E−03	1.20E−02	2.37E−02	1.76E−02
	Std	2.84E−04	8.47E−03	4.32E−03	6.78E−01	3.13E−02	3.40E−02	2.26E−02	1.18E−01	1.35E−01	3.05E−02
F_{27}	Best	⋯	⋯	5.21E−03	⋯	2.83E−19	1.55E−19	9.18E−19	**1.36E−21**	2.39E−20	8.47E−20
	Avg	5.00E−04	3.42E−04	⋯	⋯	2.95E−11	1.67E−10	1.04E−10	1.48E−12	2.51E−11	9.67E−11
	Std	2.65E−04	1.67E−04	3.42E−04	⋯	5.12E−09	3.01E−09	1.47E−09	3.50E−10	4.53E−09	1.98E−09
F_{28}	Best	⋯	⋯	⋯	⋯	2.96E−07	2.74E−06	2.93E−05	2.99E−07	3.01E−05	**2.92E−07**
	Avg	2.02E−05	1.43E−05	⋯	7.13E−02	3.88E−03	2.57E−04	6.73E−04	1.84E−04	1.06E−03	7.80E−03
	Std	3.39E−08	6.96E−15	⋯	5.92E−02	1.36E−02	3.45E−03	9.92E−03	2.20E−02	3.59E−02	9.98E−03
F_{29}	Best	⋯	⋯	⋯	⋯	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Avg	0.00E+00	0.00E+00	⋯	⋯	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	⋯	⋯	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{30}	Best	⋯	⋯	⋯	⋯	5.36E−21	3.80E−21	8.50E−21	**2.64E−21**	6.37E−21	1.48E−20
	Avg	1.05E−07	1.05E−09	⋯	⋯	2.14E−10	2.54E−10	2.46E−10	4.02E−10	4.02E−11	4.02E−10
	Std	7.75E−06	1.75E−04	⋯	⋯	3.78E−09	6.85E−09	5.98E−08	5.67E−09	4.08E−08	5.67E−09

Table 10 Average rankings based on Friedman's test for CEC2011 problem

No.	Algorithm	Rank	No.	Algorithm	Rank
1	PMFOHC	6.02	6	TMFOHC	7.33
2	LRMFOHC	6.91	7	GMFOHC	7.39
3	CLSGMFO	6.99	8	LGCMFO	7.46
4	ERMFOHC	7.08	9	MFODE	7.50
5	TrMFOHC	7.11	10	OMFO	8.31

1. **Population size: *n***

   To demonstrate the influence of the population sizes, the experiments are produced using several values for population sizes (i.e., $P = 5$, 10, 15, 20, 50, 100, 250, and 500) for the utilized 30 benchmark functions. Table 6 shows the results for different population sizes.

   As shown in Table 6, we can see that the best-normalized results for MFO with population sizes. The MFO obtained the best results (17 times) when the population size is equal to 15. Furthermore, for the 10 scalable unimodal functions, the MFO got the most of the best results when $P = 20$, it got 7 out of 10 best cases. For 12 scalable multimodal functions, the MFO got the most of the best results when $P = 20$, it got 6 out of 12 best cases. For the 8 fixed-dimension multimodal functions, the MFO got the most of the best results when $P = 20$, it got 5 out of 8 best cases. It is clearly observed that when the population size is equal to 15, it is the most suitable size for all benchmark test functions.
2. **Dimension: *D***

   In this part, to analyze the influence of the problem dimensional spaces, experiments are produced for several potential dimensional spaces (i.e., $D = 5$, 10, 15, 20, 25, 30, 35, 40, 45, and 50) as reported in the literature using the utilized 30 benchmark functions. The results for 30 functions are illustrated in Table 7 using the best normalized values.

   As shown in Table 7, the MFO obtained the overall best results when $D = 5$, it got the best results on 17 cases. Furthermore, for the 10 scalable unimodal functions, the MFO got the most of the best results when $D = 5$, it got 7 out of 10 best cases in both dimensions. For 12 scalable multimodal functions, the MFO got the most of the best results when $D = 12$, it got 6 out of 12 best cases. For the 8 fixed-dimension multimodal functions, the MFO got the most of the best results when $D = 5$, it got 4 out of 8 best cases. From these results, we concluded that increasing the overall performance of MFO is observed by increasing the problem dimensional space. Usually, the MFO is unable to solve the problem before getting the maximum number of iterations. However, as seen, MFO gives better results for high-dimensional problems.

4.2.2 Comparisons MFOHC with other methods using the benchmark functions

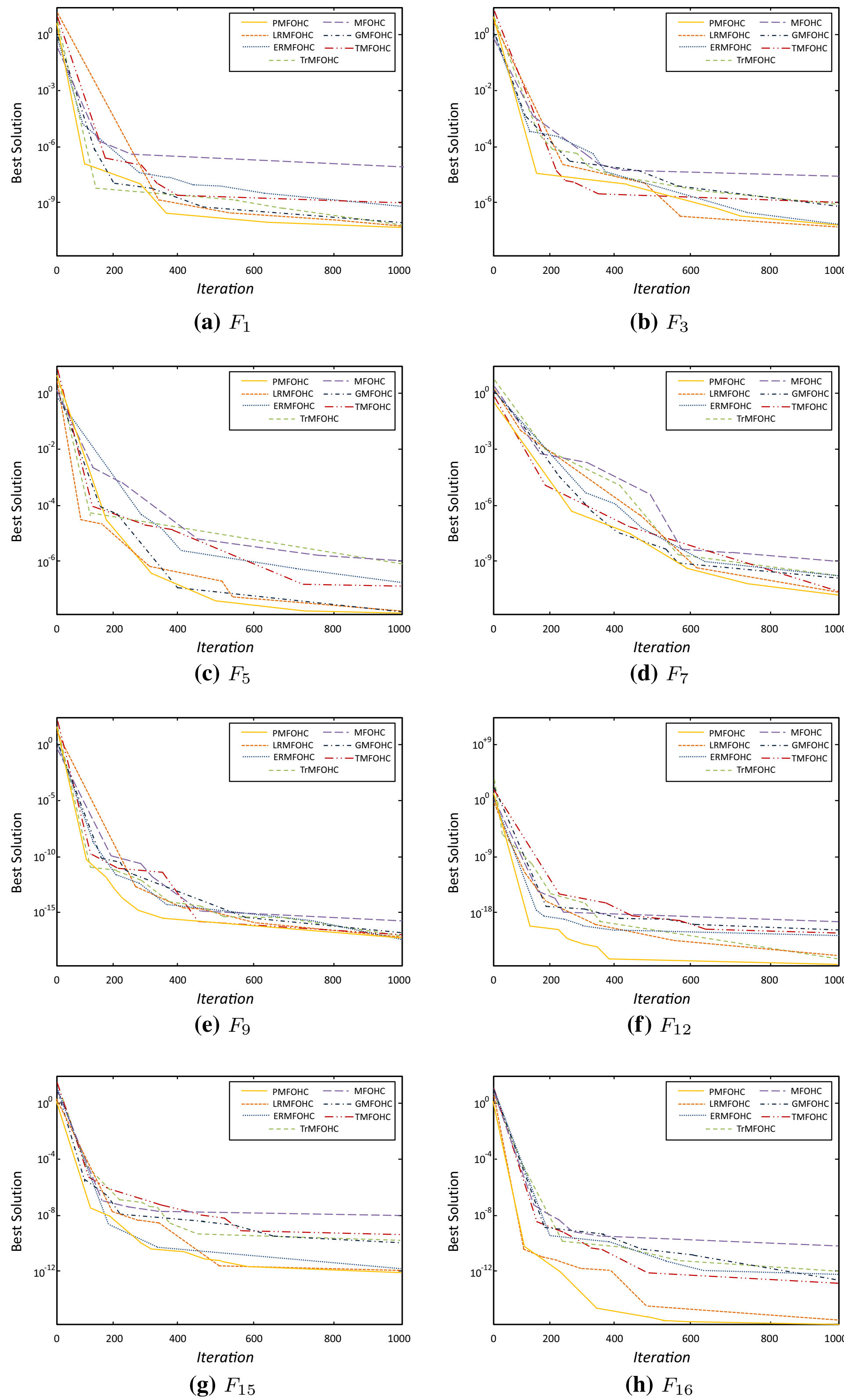
For a clear comparison, as shown in Table 8, the proposed MFOHC is compared with the basic MFO [28] and other similar nine optimization algorithms, namely, Ant Bee Colony (ABC) Algorithm [22], Bat-inspired Algorithm (BA) Yang [56], Salp Swarm Optimization (SSA) [31] , Dragonfly Algorithm (DE) Mirjalili [29], Genetic Algorithm (GA) [17], Harmony Search (HS) Algorithm Geem et al. [15], Krill Herd (KH) Algorithm [13], and Grey Wolf Optimizer (GWO) Algorithm Mirjalili et al. [30]. Table 8shows the best, average (Avg), the standard division (Std) of fitness values obtained by all comparative algorithms over 30 runs, respectively.

As shown in Table 8, the basic MFO has some weakness (weak local search) in achieving excellent results in unimodal functions (i.e., F1, F2, F4, F5, F6, and F9). Consequently, the hybrid MFO with HC is proposed to improve the exploitation searchability of MFO. Thus, functions F1–F10 are scalable unimodal benchmarks since they have just one global optimum. These functions support assessing the exploitation ability of the examined optimization algorithms. It can be seen from Table 8 that MFOHC is a very competitive algorithm compared to other similar algorithms. Mainly, it was the most effective algorithm for functions F1 and F10 in most test problems. The proposed MFOHC hence provide perfect exploitation. MFOHC got better results in solving unimodal functions compared to the proposed MFOHC where, it almost obtained all best results in unimodal functions as well as other test functions (i.e., multimodal F11–F22 and fixed-dimension multimodal F23–F30). Although the results indicate that MFOHC also has excellent exploration searchability, it is possible to further improve the exploration search to make a balance between exploitation and exploration search. Moreover, performance, diversity, and the convergence rate of MFOHC can be enhanced.

4.2.3 A comparison of MFOHC versions using Benchmark functions

In this part, as shown in the previous section that the MFOHC can further improve its exploration search abilities, new experiments series conducted to investigate the skills of the selection schemes in enhancing the global search abilities. Various selection scheme mechanisms (tournament selection scheme (TMFOHC), proportional selection scheme (PMFOHC), linear ranking selection scheme (LRMFOHC), exponential ranking selection scheme (ERMFOHC), greedy-based selection scheme (GMFOHC), and truncation selection scheme (TrMFOHC)) have been tested on the MFOHC to improve its exploration search abilities, as well as, various versions of MFO from the literature have been used (i.e.,

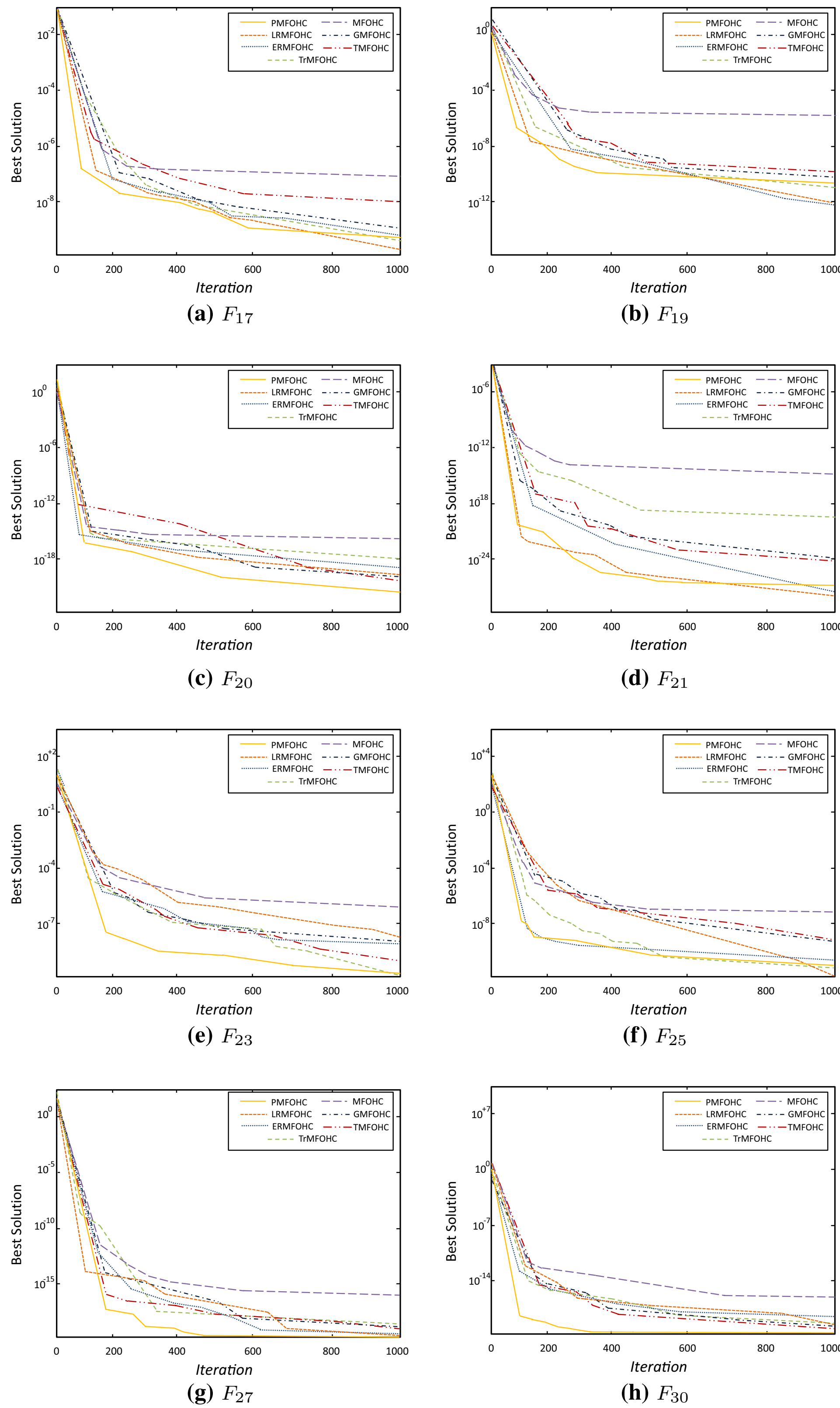
Fig. 4 Convergence graphs of the benchmark functions



LGCMFO [55], CLSGMFO [54], MFODE [11], and OMFO [36]) to evaluate the performance of the MOFHC versions.

Contrary to unimodal functions, multimodal functions cover many local optima, where number grows exponentially with the number of decision variables (problem size).

Fig. 5 Convergence graphs of the benchmark functions



(a) F_{17} **(b)** F_{19} **(c)** F_{20} **(d)** F_{21} **(e)** F_{23} **(f)** F_{25} **(g)** F_{27} **(h)** F_{30}

Accordingly, this kind of benchmark functions becomes very beneficial if the objective is to evaluate the exploration search ability of an optimization algorithm.

Optimization of benchmark functions is a very challenging job because just a precise balance between exploration and exploitation supports local optima to be evaded. Optimization results listed in Table 9 show that the proposed

Table 11 Best, average (Avg), and standard deviation (Std) for comparing the MFOHC versions using five CEC 2011 real-world problems

Function	Metric	Selection schemes						
		MFOHC	GMFOHC	TMFOHC	TrMFOHC	PMFOHC	LRMFOHC	ERMFOHC
*CEC–P*1	Best	7.8+E07	5.7+E06	3.9+E07	6.6+E07	**2.1+E06**	1.8+E07	3.4+E07
	Avg	8.3+E08	5.2+E07	2.8+E08	1.2+E08	**1.0+E07**	3.8+E07	3.4+E08
	Std	3.2+E07	2.0+E07	2.4+E07	2.4+E07	**2.6+E05**	2.3+E07	2.5+E07
*CEC–P*2	Best	24.02	23.93	20.93	14.39	**13.79**	23.93	17.53
	Avg	24.83	23.93	22.22	20.47	**18.22**	23.93	19.02
	Std	0.81	**7.2E−15**	0.64	1.78	3.15	**7.2E−15**	0.66
*CEC–P*3	Best	1.9+E06	**8.9+E05**	1.1+E06	1.4+E06	9.6+E05	1.0+E06	1.6+E06
	Avg	2.0+E06	1.2+E06	1.3+E06	1.7+E06	**1.1+E06**	1.3+E06	1.5+E06
	Std	1.7+E05	1.0+E05	9.3+E04	1.3+E05	**8.7+E04**	1.3+E05	1.4+E05
*CEC–P*4	Best	8.2+E07	**4.6+E06**	4.0+E07	7.9+E07	5.0+E06	1.4+E07	2.0+E07
	Avg	1.5+E08	1.7+E07	6.2+E07	1.1+E08	**1.2+E07**	3.7+E07	6.3+E07
	Std	2.2+E07	8.5+E06	1.1+E07	1.3+E07	**6.4+E06**	9.3+E06	7.2+E06
*CEC–P*5	Best	1.92	1.73	1.52	1.95	**1.45**	1.62	1.98
	Avg	2.84	2.13	2.44	2.42	**2.06**	2.17	2.56
	Std	0.24	**0.20**	0.22	0.21	0.22	0.30	0.21

hybrid MFO with HC using proportional selection scheme (PMFOHC) is almost the best optimizer in all test problems and overcomes other similar comparative algorithms[1]. It is definitely demonstrated that the proposed PMFOHC support exploration and exploitation phases to be balanced. Moreover, the results indicate that PMFOHC also has excellent exploration search ability. However, the proposed PMFOHC always will be the most useful algorithm in the majority of function problems.

The performance of the proposed versions of the MFOHC algorithm is further evaluated using Friedman's statistical tests. Table 10 provides the average ranking of the proposed MFOHC versions against the comparative methods using Friedman's test. It can be noticed that the proposed PMFOHC version is ranked first, followed by LRMFOHC, ERMFOHC, TrMFOHC, TMFOHC, and GMFOHC versions, which ranked second, fourth, fifth, sixth, and seventh, respectively. The overall P value computed by Friedman's test is 9.43E−11, which is below the significant level (i.e., $\alpha = 0.05$). This value indicates that there are significant differences between the performance of the comparative methods used.

Figures 4 and 5 shows the convergence graphs of the unimodal benchmark functions (F_1, F_3, F_5, F_7, and F_9), multimodal benchmark functions (F_{12}, F_{15}, F_{16}, F_{17}, F_{19}, F_{20}, and F_{21}), and fixed-dimension multimodal benchmark functions (F_{23}, F_{25}, F_{27}, and F_{30}). The convergence graphs are plotted between the best solutions of each algorithm and the number of iterations based on the results acquired through 30 independent runs.

It is observed from the convergence graphs of the unimodal functions that the PMFOHC outperformed the other versions in F_1, F_5, and F_7. While it achieved close results from the LRMFOHC and ERMFOHC in F_3 and F_9 with superiority to LRMFOHC and ERMFOHC. Thus, it can be summarized that the PMFOHC is the most efficient version in dealing with unimodal benchmark functions. However, PMFOHC still has a weak at the beginning (i.e., from start until 200–400 iterations). Thus, it suffers from slow convergence when dealing with the local search functions.

Similar to the mentioned above (i.e., unimodal functions) the convergence performance of the PMFOHC achieved best results in 4 out of 7 multimodal benchmark functions [i.e., (F_{12}, F_{15}, F_{16}, and F_{20})]. In F_{17} and F_{21} the PMFOHC is the fastest method for finding the best solutions in the first part, while in the last part (i.e, after iteration 600) the LRMFOHC was the best. In F_{19} ERMFOHC achieved the best results compared with the other versions. Consequently, although PMFOHC outperformed the other algorithms, it needs more enhancements to achieve the best solutions in all global search functions.

In the fixed-dimension multimodal benchmark functions, PMFOHC got the best results in 3 of 4 of the functions (F_{23}, F_{27}, and F_{30}), while in F_{25} the superiority was obvious to the LRMFOHC, followed by TrMFOHC and PMFOHC.

Based on the above, it can be noticed that PMFOHC proved its performance in most functions of the three categories of the benchmarks. The experiment results are convincing because of the structure of PMFOHC combines the feature of MFO in the exploration search, supported by the feature of HC in the exploitation search, and distinguished

[1] The set of benchmark functions in our work is not matched totally with the other sets in the literature. Thus, we selected a group of benchmark function which matched with our work

from the rest of the proposed methods by using the proportional selection schemes to increase the quality of the selected solutions.

4.2.4 A comparison of MFOHC versions using real world problems

The real-world problems are presented in Sect. 4.1.2 where it can be considered as discrete or continuous problems. Thus, can be used to evaluate the performance of different metaheuristic algorithms. All results in Table 11 are gained by 50 separate runs on the five real-world problems.

PMFOHC determines the best solutions on three out of five real problems (except *CEC–P3* and *CEC–P4*) followed by GMFOHC which achieved best solution in both *CEC–P3* and *CEC–P4*. Regarding the mean solution, PMFOHC outperforms the other methods in all real problems. Finally, the std results show that the PMFOHC obtained the best results in *CEC–P1*, *CEC–P3*, and *CEC–P4*. GMFOHC obtained the best results in *CEC–P2* and *CEC–P5*. The summary of the results in Table 11 refer that PMFOHC shows the best performance comparing with the other six methods.

5 Conclusion and future works

This paper presented new alternative methods using moth-flame optimization (MFO). The proposed methods include two main steps: in the first step, the basic MFO is hybridized with hill climbing (HC) local search to improve its exploitation search, called MFOHC. In the second step, six popular selection schemes are investigated, and the proportional selection scheme is selected as the best to improve the exploration search of the MFOHC by maintaining the diversity of the solutions, called PMFOHC.

Experiments are conducted using thirty benchmark functions and five IEEE CEC 2011 real-world problems. The results of the proposed algorithms are compared to several similar algorithms published in the literature. The effectiveness of each algorithm is evaluated by three measures, the best, average, standard deviation of the fitness values. The results illustrated that the PMFOHC version is almost the best optimizer in all test problems and it as a summary, the results for solving the real-world problems showed that the proposed PMFOHC has a promising ability to be very useful in solving the structural design problems with unfamiliar search spaces also overcoming other similar comparative algorithms. The proposed PMFOHC support exploration and exploitation phases to be balanced through, keeping the diversity of the solutions. However, it suffers from a weakness of slow convergence.

In future work, we will enhance the limitation of the proposed methods by using new search techniques such as stochastic hill-climbing and opposition-based learning. Also, we will utilize different optimization problems, as well as multi-objective problems to achieve better results.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflicts of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Abdelmadjid C, Mohamed SA, Boussad B (2013) Cfd analysis of the volute geometry effect on the turbulent air flow through the turbocharger compressor. Energy Proced 36:746–755
2. Abualigah LM, Khader AT, Hanandeh ES (2018) A hybrid strategy for krill herd algorithm with harmony search algorithm to improve the data clustering1. Intell Decis Technol 12(1):3–14
3. Allam D, Yousri D, Eteiba M (2016) Parameters extraction of the three diode model for the multi-crystalline solar cell/module using moth-flame optimization algorithm. Energy Convers Manag 123:535–548
4. Amini S, Homayouni S, Safari A, Darvishsefat AA (2018) Object-based classification of hyperspectral data using random forest algorithm. Geo-spat Inf Sci 21(2):127–138
5. Bäck T (1995) Generalized convergence models for tournament-and (μ, lambda)-selection
6. Bhesdadiya R, Trivedi IN, Jangir P, Kumar A, Jangir N, Totlani R (2017) A novel hybrid approach particle swarm optimizer with moth-flame optimizer algorithm. Advances in computer and computational sciences. Springer, Berlin, pp 569–577
7. Blickle T, Thiele L (1995) A mathematical analysis of tournament selection. ICGA Citeseer 95:9–15
8. Blum C, Li X (2008) Swarm intelligence in optimization. Swarm intelligence. Springer, Berlin, pp 43–85
9. Das S, Suganthan PN (2010) Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur University, Nanyang Technological University, Kolkata, pp 341–359
10. El Aziz MA, Ewees AA, Hassanien AE (2017) Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Expert Syst Appl 83:242–256
11. Elaziz MA, Ewees AA, Ibrahim RA, Lu S (2020) Opposition-based moth-flame optimization improved by differential evolution for feature selection. Math Comput Simul 168:48–75
12. Elsakaan AA, El-Sehiemy RA, Kaddah SS, Elsaid MI (2018) An enhanced moth-flame optimizer for solving non-smooth economic dispatch problems with emissions. Energy 157:1063–1078
13. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
14. Gaston KJ, Bennie J, Davies TW, Hopkins J (2013) The ecological impacts of nighttime light pollution: a mechanistic appraisal. Biol Rev 88(4):912–927
15. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68

16. Glover F (1977) Heuristics for integer programming using surrogate constraints. Decis Sci 8(1):156–166
17. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
18. Hancock PJ (1994) An empirical comparison of selection methods in evolutionary algorithms. AISB workshop on evolutionary computing. Springer, Berlin, pp 80–94
19. Hazir E, Erdinler ES, Koc KH (2018) Optimization of cnc cutting parameters using design of experiment (DOE) and desirability function. J Forest Res 29(5):1423–1434
20. Holland JH et al (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, London
21. Jangir N, Pandya MH, Trivedi IN, Bhesdadiya R, Jangir P, Kumar A (2016) Moth-flame optimization algorithm for solving real challenging constrained engineering optimization problems. In: 2016 IEEE students' conference on electrical, electronics and computer science (SCEECS), IEEE, pp 1–5
22. Karaboga D, Basturk B (2007) Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. International fuzzy systems association world congress. Springer, Berlin, pp 789–798
23. Kennedy J (2010) Particle swarm optimization. Encyclop Mach Learn 12:760–766
24. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, IEEE, vol 4, pp 1942–1948
25. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
26. Koziel S, Yang XS (2011) Computational optimization, methods and algorithms, vol 356. Springer, Berlin
27. Li WK, Wang WL, Li L (2018) Optimization of water resources utilization by multi-objective moth-flame algorithm. Water Resour Manag 32:3303–3316
28. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228–249
29. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073
30. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
31. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
32. Mitchell M (1998) An introduction to genetic algorithms. MIT press, London
33. Oladele R, Sadiku J (2013) Genetic algorithm performance with different selection methods in solving multi-objective network design problem. Int J Comput Appl 70:12
34. Razali NM, Geraghty J et al (2011) Genetic algorithm performance with different selection strategies in solving tsp. Proc World Congress Eng Int Assoc Eng Hong Kong 2:1–6
35. Reddy S, Panwar LK, Panigrahi BK, Kumar R (2018) Solution to unit commitment in power system operation planning using binary coded modified moth flame optimization algorithm (bmmfoa): a flame selection based computational technique. J Comput Sci 25:298–317
36. Sapre S, Mini S (2019) Opposition-based moth flame optimization with cauchy mutation and evolutionary boundary constraint handling for global optimization. Soft Comput 23(15):6023–6041
37. Sarma A, Bhutani A, Goel L (2017) Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality. In: 2017 intelligent systems conference (IntelliSys), IEEE, pp 52–60
38. Savsani V, Tawhid MA (2017) Non-dominated sorting moth flame optimization (ns-mfo) for multi-objective problems. Eng Appl Artif Intell 63:20–32
39. Schlierkamp-Voosen D, Mühlenbein H (1993) Predictive models for the breeder genetic algorithm. Evol Comput 1(1):25–49
40. Sharma A (2014) Bioinformatic analysis revealing association of exosomal MRNAS and proteins in epigenetic inheritance. J Theor Biol 357:143–149
41. Shehab M (2020) Hybridization cuckoo search algorithm for extracting the ODF maxima. Artificial intelligence in diffusion MRI. Springer, Berlin, pp 111–146
42. Shehab M, Khader AT, Al-Betar M (2016) New selection schemes for particle swarm optimization. IEEJ Trans Electro Inf Syst 136(12):1706–1711
43. Shehab M, Khader AT, Al-Betar MA (2017) A survey on applications and variants of the cuckoo search algorithm. Appl Soft Comput 61:1041–1059
44. Shehab M, Khader AT, Al-Betar MA, Abualigah LM (2017) Hybridizing cuckoo search algorithm with hill climbing for numerical optimization problems. In: Information technology (ICIT), 2017 8th international conference on, IEEE, pp 36–43
45. Shehab M, Khader AT, Laouchedi M (2017) Modified cuckoo search algorithm for solving global optimization problems. International conference of reliable information and communication technology. Springer, Berlin, pp 561–570
46. Shehab M, Abualigah L, Al Hamad H, Alabool H, Alshinwan M, Khasawneh AM (2019) Moth-flame optimization algorithm: variants and applications. Neural Comput Appl 20:1–26
47. Shehab M, Khader AT, Alia MA (2019) Enhancing cuckoo search algorithm by using reinforcement learning for constrained engineering optimization problems. In: 2019 IEEE Jordan international joint conference on electrical engineering and information technology (JEEIT), IEEE, pp 812–816
48. Smith T, Villet M (2001) Parasitoids associated with the diamondback moth, *Plutella xylostella* (l.), in the eastern Cape, south Africa. In: The management of diamondback moth and other crucifer pests. Proceedings of the fourth international workshop, pp 249–253
49. Sodeifian G, Ardestani NS, Sajadian SA (2019) Extraction of seed oil from *Diospyros lotus* optimized using response surface methodology. J Forest Res 30(2):709–719
50. Tang Z, Gong M (2019) Adaptive multifactorial particle swarm optimisation. CAAI Trans Intell Technol 4(1):37–46
51. Trivedi I, Kumar A, Ranpariya AH, Jangir P (2016) Economic load dispatch problem with ramp rate limits and prohibited operating zones solve using levy flight moth-flame optimizer. In: 2016 international conference on energy efficient technologies for sustainability (ICEETS), IEEE, pp 442–447
52. Volkovs M, Chiang F, Szlichta J, Miller RJ (2014) Continuous data cleaning. In: 2014 IEEE 30th international conference on data engineering, IEEE, pp 244–255
53. Wang M, Chen H, Yang B, Zhao X, Hu L, Cai Z, Huang H, Tong C (2017) Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. Neurocomputing 267:69–84
54. Xu Y, Chen H, Heidari AA, Luo J, Zhang Q, Zhao X, Li C (2019) An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. Expert Syst Appl 129:135–155
55. Xu Y, Chen H, Luo J, Zhang Q, Jiao S, Zhang X (2019) Enhanced moth-flame optimizer with mutation strategy for global optimization. Inf Sci 492:181–203
56. Yang XS (2010) A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin, pp 65–74
57. Yousri D, AbdelAty AM, Said LA, AboBakr A, Radwan AG (2017) Biological inspired optimization algorithms for

cole-impedance parameters identification. AEU Int J Electron Commun 78:79–89
58. Zawbaa HM, Emary E, Parv B, Sharawi M (2016) Feature selection approach based on moth-flame optimization algorithm. In: 2016 IEEE congress on evolutionary computation (CEC), IEEE, pp 4612–4617