Soft Computing (2023) 27:12155–12180
https://doi.org/10.1007/s00500-023-08416-1

OPTIMIZATION

An adaptive moth flame optimization algorithm with historical flame archive strategy and its application

Zhenyu Wang[1] · Zijian Cao[1] · Haowen Jia[1]

Accepted: 4 May 2023 / Published online: 22 May 2023

Abstract
Moth Flame Optimization (MFO) is a new nature-inspired heuristic algorithm, and has successfully been applied in various fields of practical engineering. To enhance exploitation of MFO and avoid dropping into local optimal solution, an adaptive MFO algorithm with historical flame archive strategy is proposed in this paper, which is termed MFO–HFA to avoid ambiguity. In MFO–HFA, to make full use of population history information, the archive consists of historical optimal individuals, which is utilized to preserve the information of better historical flame. Besides, to make full use of the information of top flame information, a top flame randomly matching mechanism is utilized to improve the convergence ability of population. To demonstrate the advantage of MFO–HFA, it is compared with several well-known variants of MFO and some state-of-the-art intelligence algorithms on both 25 benchmark functions of CEC 2005. The experimental results indicate that MFO–HFA outperforms other compared algorithms and has obtained best accuracy. Furthermore, MFO–HFA is used to generate the rules of IDS by NSL-KDD dataset. The test results demonstrate that MFO–HFA outperforms compared algorithms and has gained 96.5% accuracy.

Keywords Moth flame optimization · Historical flame archive · Top flame randomly matching mechanism

1 Introduction

To search the optimal solution of the nonlinear, non-differentiable and non-separable complex problem, swarm intelligence algorithms were proposed to simulate the foraging behaviors and biological habits of animals, and have received increasing attention in last several decades (Revay and Zelinka 2019; Kaur and Kumar 2020; Mehta and Saxena 2020; Farrag et al. 2019; Kumar 2021, 2019; Dongoran et al. 2018; Zhang et al. 2018; Kanata et al. 2018; Daylamani-Zad et al. 2017). The classical swarm intelligence algorithms and variants have Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995), Artificial Bee Colony (ABC) (Karaboga 2005) and Ant Colony Optimization (ACO) (Dorigo et al. 1991), Cuckoo Search Algorithm and Hill Climbing (CSAHC) (Shehab et al. 2018), Cuckoo Search Algorithm by using Reinforcement Learning (CSARL) (Shehab et al. 2018), Cuckoo Search combined with Bat Algorithm (CSBA) (Shehab et al. 2019), Opposition-based learning in Multi-Verse Optimizer (OMVO) (Shehab and Abualigah 2022). MFO is a novel swarm intelligence algorithm, proposed by Mirjalili in 2015 (2015) and inspired by the transverse orientation mechanism of moths in nature. Meanwhile, MFO has successfully been applied in various fields of the practical engineering, such as breast cancer detection (Ahmed et al. 2019; Sayed and Hassanien 2017), clustering for internet of things (Reddy and Babu 2019; Yang et al. 2017; Bharany et al. 2022), feature selection (Sayed and Hassanien 2017), productivity forecasting (Reddy and Babu 2019), power dispatch problems (Elsakaan et al. 2018; Mei et al. 2018; Trivedi et al. 2018; Anbarasan and Jayabarathi 2017) and image segmentation (Khairuzzaman and Chaudhury 2017; Jia et al. 2019; Abd El Aziz et al. 2017; Said et al. 2017). However, the search performance

✉ Zijian Cao
bosscao@163.com

Zhenyu Wang
tsingke123@163.com

Haowen Jia
603176912@qq.com

[1] School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China

of MFO is extremely influenced by control parameters, global exploration skill and local exploitation ability. To further improve the search performance of MFO, researchers mainly promote the optimization ability of MFO from the following three aspects: enhancing the global exploration and local exploitation ability, adjusting control parameters, and mixing with other algorithms.

In terms of global exploration and local exploitation, Opposition-based Moth Flame Optimization (OMFO) was proposed by Apinantanakon et al. (2017), which introduces the opposite location of moths during the spiral search process. Li et al. (2020) proposed an Improved Moth Flame Optimization (IMFO) to improve global optimization ability of MFO algorithm by using Levy flight mechanism and dimension-by-dimension evaluation method. Zhao et al. (2018) proposed an Ameliorated Moth Flame Optimization (AMFO), which not only improves the solution precision of classic MFO, but also enhances the convergence speed and the stability of MFO. Xu et al. (2019a) designed an enhanced moth flame optimizer with mutation strategy for avoiding premature, and it consists of many mutation strategies, such as Gaussian mutation (GMFO), Cauchy mutation (CMFO) and Levy mutation (LMFO). The Gaussian mutation can improve the exploitation ability of the algorithm, and the Cauchy mutation may guarantee the population to search in the major area and discard local optima readily. The Levy mutation can help population to escape from local optima because of its heavy-tailed distribution.

The optimized moth flame optimizer based on Gaussian mutation and cultural learning was proposed in Xu et al. (2018). Xu et al. (2019b) proposed an improved MFO algorithm based on the chaotic local search and Gaussian mutation. Nadimi-Shahraki et al. (2021b) proposed the migrate-based moth flame optimization (M-MFO) algorithm that uses the migrate operator to improve the position of unlucky moths. The migrate operator promotes the diversity of population, so that population can obtain better search performance. To provide a more efficient tool for optimization purposes, Shan et al. (2021) proposed a double adaptive weight mechanism into MFO algorithm, termed as WEMFO. WEMFO adaptively change the search strategy in different search periods. To fully utilize the information of flame population, an enhanced moth flame optimization with multiple flame guidance mechanism (EMFO) is proposed in Wang et al. (2022). In EMFO, the intersection information of multiple flames is used to guide moth search, which enhances the global diversity of moth population. Nadimi-Shahraki et al. (2021c) proposed a multi-trial vector-based moth flame optimization algorithm, named for MTV-MFO. MTV-MFO uses three different search strategies to enhance the global search ability, and prevents the original MFO's premature convergence during the optimization process. Ma et al. (2021) proposed an improved moth flame optimization algorithm for alleviating the premature convergence problem. An inertia weight of diversity feedback control is utilized to balance the global explore ability and local exploitation ability.

In terms of hybrid algorithm, Shehab et al. (2021) proposed a hybrid moth flame optimization algorithm by using new selection schemes, in which hill climbing (HC) is used to hybrid with moth flame optimization (HC-MFO) for enhancing the global exploration ability. Mohammad et al. (Nadimi-Shahraki et al. 2022) proposed an effective hybridizing of whale optimization algorithm (WOA) and a modified moth flame optimization algorithm, named for WMFO to solve the optimal power flow. In WMFO, WOA and the modified MFO cooperate to effectively discover the promising areas and provide high-quality solutions. Sayed et al. (Sayed and Hassanien 2018) proposed the hybrid MFO and Simulated Annealing (SA), and MFO comminating with PSO were proposed in Bhesdadiya et al. (2017); Anfal and Abdelhafid 2017; Jangir 2017). Gravitational Search Algorithm was hybridized into MFO, which was proposed by Sarma et al. (Sarma et al. 2017). Intelligent facial emotion recognition using the hybrid MFO and Firefly Algorithm (FA) was proposed by Zhang et al. (2016).

Regarding control parameters, Emary et al. (2016) put forward a chaos-based automatic control method on exploration and exploitation rates against the manual parameter control of MFO. Wang et al. (2017) proposed a chaotic strategy to simultaneously perform parameter optimization and feature selection in MFO. To improve the exploitation of moth population, Guvenc et al. (2017) proposed the chaotic moth swarm algorithm, in which ten chaotic maps were incorporated into MFO algorithm for finding the best numbers of moths.

Although those variants of MFO have obtained better performance than classical MFO, they exhibit poor solution accuracy in solving multi-modal optimization problems. To alleviate this problem, an adaptive MFO algorithm with historical flame archive is presented in this paper, inspired by individual' optima guided method in PSO.

The main contributions of this paper can be summarized as follows.

(1) An adaptive historical flame archive strategy is proposed to enhance the solution precision of multimodal problems by storing the better historical flame information.
(2) To accelerate the convergence speed and make full use of flame information, a top flame randomly matching mechanism is constructed by randomly selecting one of top flames to guide moth search.

(3) To systematically verify the superiority of the proposed MFO–HFA algorithm, 25 complex benchmark functions are utilized to estimate the overall performance of MFO–HFA. Besides, MFO–HFA is used to generate the rules of IDS by NSL-KDD dataset.

The rest of this paper is given as follows. In Sect. 2, the review of original MFO algorithm is summarized. The adaptive MFO algorithm with historical flame archive is detailly described in Sect. 3. Section 4 presents the simulation results on the 25 benchmark functions. MFO–HFA is used to generate the rule of IDS by NSL-KDD dataset in Sect. 5, followed by conclusions in Sect. 6.

2 Moth flame optimization

MFO is a novel population-based intelligence algorithm, inspired by the navigation mode of the moths using the moon light in nature. The transverse orientation of moth is shown in Fig. 1. Moth flying at a fixed angle around the moon is an effective method to travel in a straight line for long distance in night. When moth is flying around an artificial light, moth will trap into the spiral path and gradually approach the light. Figure 2 shows that moths gradually approach the flame during the spiral flight process, which is mapped to the spiral search that has the promising performance in solving practical engineering optimization problems. The optimization function and the composition of MFO is described in detail as follows.

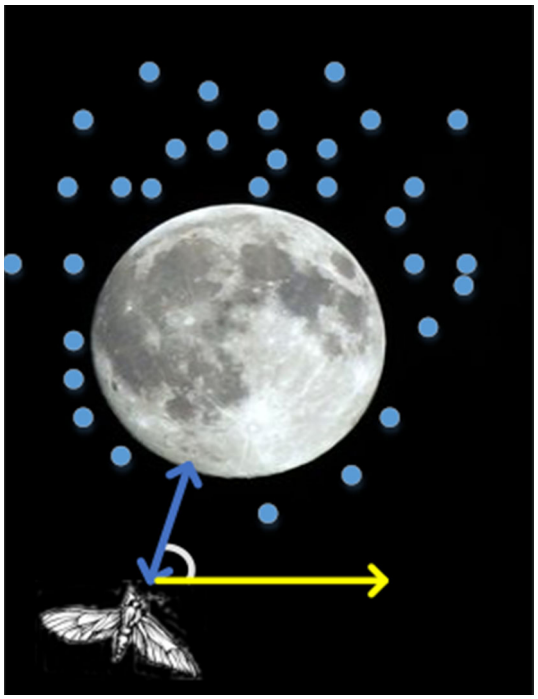


Fig. 1 Transverse orientation of moths

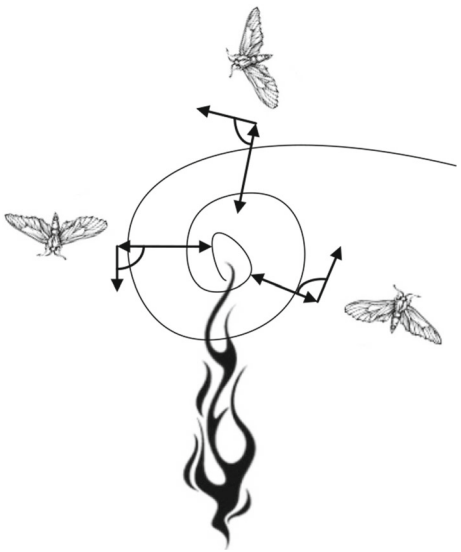


Fig. 2 Spiral flight of moths

2.1 Problem formulation

The problem optimized by MFO algorithm is formulated as follows.

$$f(X^*) = \min f(X) \tag{1}$$

where $X_i = \{x_1, x_2, \ldots, x_D\}$ is a solution of objective problem, and D denotes the number of dimensions. $f(X)$ is fitness value of variable X, and is a minimizing problem. X^* is the global optimal solution.

2.2 Generating the initial population of moths

The moth population of MFO algorithm can be described as follows.

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1D} \\ m_{21} & m_{22} & \cdots & m_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \cdots & m_{ND} \end{bmatrix} \tag{2}$$

where N is the number of moths.

The fitness values of all moths are listed in a matrix as follows.

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_N \end{bmatrix} \tag{3}$$

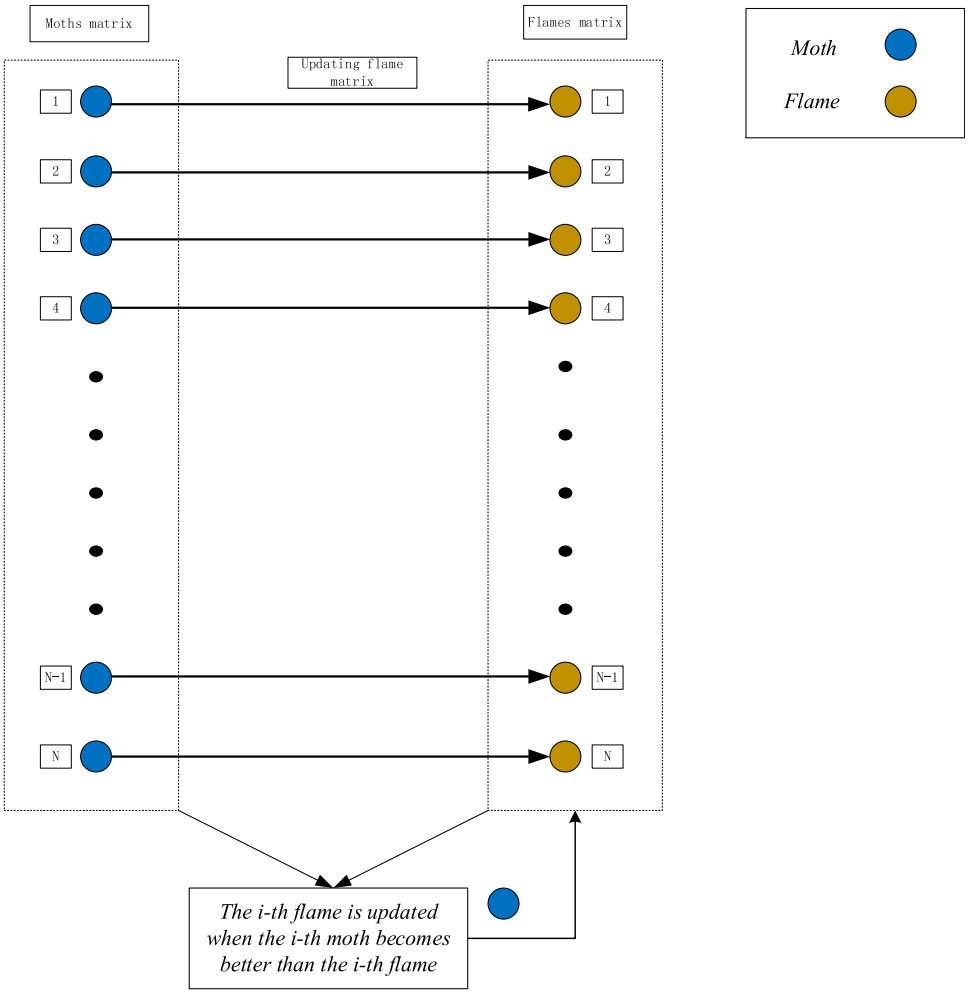


Fig. 3 Updating mechanism of historical flame archive

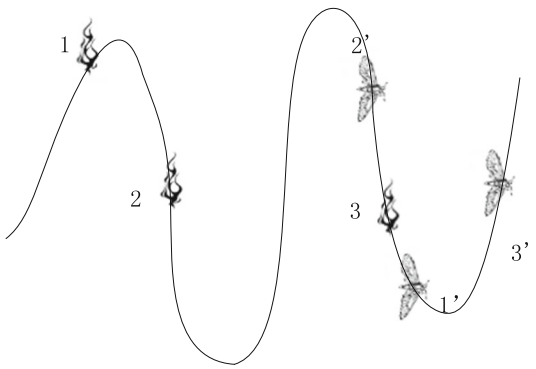


Fig. 4 Schematic diagram of updating flame (The peak value is the worst fitness value of the functions, and the depression value represents the optimal value of fitness value)

Another essential component in the MFO algorithm is flame. An array similar to the moth matrix is given as follows.

$$F = \begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1D} \\ F_{21} & F_{22} & \cdots & F_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ F_{N1} & F_{N2} & \cdots & F_{ND} \end{bmatrix} \tag{4}$$

For the flames, the matrix of fitness values also is key and represented as follows.

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_N \end{bmatrix} \tag{5}$$

2.3 Operators of MFO

MFO has three main operators described detailly as follows.

$$MFO = (I, P, T) \tag{6}$$

where I is the initialization function that generates a group unfirmly random solution (moths) in optimization space

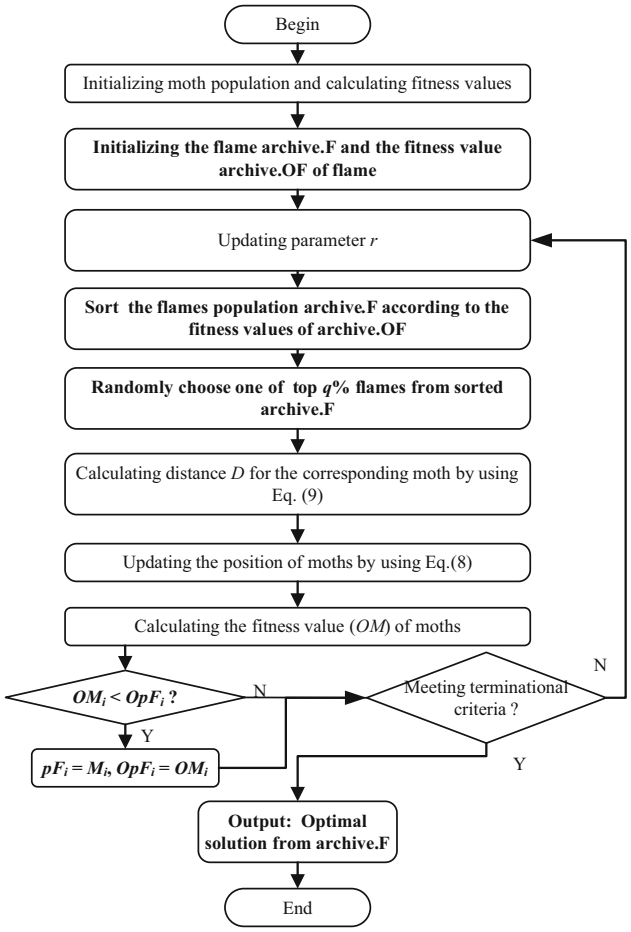


Fig. 5 Flowchart of MFO–HFA

and corresponding to fitness values ($I : \emptyset \rightarrow \{M, OM\}$). P stands for spiral search function that is the main operator ($P : M \rightarrow M$), and moves the moths around the flames for searching optimal solution. T refers to whether to satisfy the optimization process ($T : M \rightarrow true, false$).

The following equation represents I operator, which is used to generate initial moth population.

$$M_{i,j} = rand * \left(ub_j - lb_j\right) + lb_j \tag{7}$$

where $M_{i,j}$ denotes j th dimension of i th moth, ub_j and lb_j represent the upper bounds and lower bounds, respectively. *rand* is the random in range [0,1]. The moths fly in the search space by using the transverse orientation mechanism. There are three conditions that should be complied with when utilizing a spiral search. Firstly, the position of the moth should be the starting point of the spiral search. Subsequently, the position of the flame should be the ending point of the spiral search. Finally, the scope of spiral search should be in the search space.

Thus, the logarithmic spiral search of the MFO algorithm can be represented as follows.

$$moth_{i,t+1} = D_{ij} * e^{b\beta} * \cos(2\pi\beta) + F_j \tag{8}$$

$$D_{ij} = |F_j - moth_{i,t}| \tag{9}$$

where F_j is jth flame, and $moth_{i,t}$ denotes ith moth at t generation. Meanwhile, D_{ij} stands for the distance between jth flame and ith moth, and b is the spiral constant and defines the shape of the logarithmic spiral. In addition, β represents the random in interval $[r, 1]$, where r is linearly decreased from -1 to -2.

As it can be seen in Eq. (9), the next position of a moth is determined by the corresponding flame, and it is not necessarily in the space between them. Therefore, the exploration ability and the exploitation capacity of the population can be guaranteed.

2.4 Updating the number of flames

The balance between local search ability and global exploration ability is achieved by the original time-varying mechanism (i.e., the number of flames gradually decrease) that is defined as follows.

$$Flame\_no = round(N - (N-1) * gen/genMax) \tag{10}$$

where *Flame_no* is the number of flames. *gen* denotes *gen*th iteration, and *genMax* is the maximal generation. Besides, *round* stands for the rounding function.

2.5 Flame matching mechanism

How to choose the flame for moths is a crucial problem that decides the performance of the algorithm. In MFO algorithm, the flames are sorted based on their fitness values after updating the matrix of flames in each iteration. Then, the flame select method is executed to improve the convergence ability of the population, listed as follows.

$$MF_i = \begin{cases} F_i, & if\ i \leq Flame\_no \\ F_{Flame\_no}, & otherwise \end{cases} \tag{11}$$

where ith moth fly around MF_ith flame of sorted flame matrix.

2.6 Process of spiral search of MFO

To sum up, the search process of MFO can be described as follows. Moths are randomly generated in the search space, and the fitness value of each moth individual is calculated. Some top positions found so far are viewed as flames and added to flame population. Subsequently, control parameter flame_no and decrease factor r are updated according to the time-varying mechanism, respectively. The positions of moths are renewed by the spiral search function to find

Table 1 Results of solution accuracy obtained by six compared algorithms

Func	Results	MFO–HFA	AMFO	GMFO	CMFO	LMFO	OMFO
F1	Mean	**2.11E−17**	8.22E+03≠	7.26E−11≠	1.53E−10≠	9.80E−11≠	8.39E−11≠
	Std	**3.76E−17**	1.30E+03	1.05E−10	3.29E−10	1.11E−10	7.62E−11
	P value	–	8.35E−34	1.15E−03	2.39E−02	6.06E−05	1.41E−06
	H value	–	1	1	1	1	1
F2	Mean	**1.27E+00**	1.96E+04≠	3.52E+01≠	2.88E+01≠	3.79E+01≠	3.62E+01≠
	Std	**1.87E+00**	3.56E+03	2.44E+01	1.65E+01	2.90E+01	2.21E+01
	P value	–	4.45E−31	9.52E−09	8.58E−11	8.43E−08	3.22E−10
	H value	–	1	1	1	1	1
F3	Mean	**2.31E+06**	1.30E+08≠	3.44E+06≠	4.12E+06≠	3.92E+06≠	3.92E+06≠
	Std	**1.18E+06**	2.03E+07	1.46E+06	1.79E+06	1.56E+06	1.41E+06
	P value	–	1.41E−33	4.13E−03	1.17E−04	1.52E−04	6.90E−05
	H value	–	1	1	1	1	1
F4	Mean	**9.69E+01**	2.54E+04≠	3.18E+03≠	2.70E+03≠	2.78E+03≠	2.77E+03≠
	Std	**1.41E+02**	3.66E+03	2.08E+03	1.78E+03	1.63E+03	1.57E+03
	P value	–	1.47E−35	1.70E−09	2.86E−09	1.09E−10	4.79E−11
	H value	–	1	1	1	1	1
F5	Mean	2.90E+04	2.83E+04 ~	2.72E+04ξ	**2.28E+04ξ**	**2.28E+04ξ**	2.74E+04ξ
	Std	3.53E+01	2.84E+03	2.77E+03	**4.84E+02**	**4.31E+02**	2.64E+03
	P value	–	2.10E−01	1.97E−03	**4.56E−48**	**1.41E−50**	4.39E−03
	H value	–	0	1	**1**	**1**	1
F6	Mean	1.02E+02	6.22E+08≠	**9.52E+01 ~**	2.32E+02≠	2.32E+02≠	1.54E+02≠
	Std	1.79E+02	1.52E+08	**1.09E+02**	4.16E+02	5.14E+02	2.75E+02
	P value	–	2.01E−25	**8.64E−01**	1.58E−02	2.40E−03	4.37E−05
	H value	–	1	**0**	1	1	1
F7	Mean	4.70E+03	4.57E+03ξ	4.67E+03 ~	**3.91E+03ξ**	3.93E+03ξ	4.68E+03 ~
	Std	2.53E−12	1.43E+02	9.60E+01	**1.88E+02**	1.50E+02	1.02E+02
	P value	–	7.97E−05	1.98E−01	**7.73E−26**	1.86E−29	3.22E−01
	H value	–	1	0	**1**	1	0
F8	Mean	2.10E+01	**2.09E+01 ~**	2.10E+01 ~	2.10E+01 ~	**2.09E+01ξ**	**2.09E+01 ~**
	Std	4.58E−02	**5.82E−02**	4.27E−02	4.29E−02	**5.89E−02**	**5.06E−02**
	P value	–	**9.63E−02**	9.81E−01	8.71E−01	**3.25E−02**	**5.59E−01**
	H value	–	**0**	0	0	**1**	**0**
F9	Mean	**2.42E+01**	2.32E+02≠	5.02E+01≠	5.19E+01≠	5.92E+01≠	4.50E+01≠
	Std	**6.66E+00**	1.28E+01	1.76E+01	1.88E+01	1.48E+01	9.30E+00
	P value	–	1.57E−50	1.08E−08	9.31E−09	1.91E−14	5.77E−12
	H value	–	1	1	1	1	1
F10	Mean	**6.73E+01**	2.97E+02≠	1.27E+02≠	1.12E+02≠	1.13E+02≠	1.07E+02≠
	Std	**5.40E+01**	1.25E+01	4.38E+01	2.90E+01	3.32E+01	2.71E+01
	P value	–	1.28E−25	8.69E−05	6.95E−04	7.66E−04	2.13E−03
	H value	–	1	1	1	1	1
F11	Mean	**1.10E+01**	3.88E+01≠	2.62E+01≠	3.99E+01≠	2.87E+01≠	2.50E+01≠
	Std	**1.99E+00**	1.12E+00	4.80E+00	6.92E−01	8.37E+00	3.04E+00
	P value	–	4.32E−47	2.77E−19	1.59E−49	9.26E−14	2.99E−24
	H value	–	1	1	1	1	1
F12	Mean	9.83E+05	1.02E+06 ~	1.34E+04ξ	9.17E+05 ~	3.41E+04ξ	**1.12E+04ξ**
	Std	1.30E+05	1.09E+05	9.83E+03	2.08E+05	6.13E+04	**6.25E+03**
	P value	–	2.49E−01	4.17E−37	1.82E−01	1.05E−34	**3.46E−37**
	H value	–	0	1	0	1	**1**

Table 1 (continued)

Func	Results	MFO–HFA	AMFO	GMFO	CMFO	LMFO	OMFO
F13	Mean	**2.85E+00**	3.15E+01≠	7.34E+00≠	6.57E+00≠	7.15E+00≠	6.50E+00≠
	Std	**4.11E−01**	2.52E+00	2.23E+00	1.68E+00	2.31E+00	2.23E+00
	P value	–	1.91E−45	3.61E−13	2.33E−14	3.82E−12	1.85E−10
	H value	–	1	1	1	1	1
F14	Mean	1.31E+01	1.34E+01≠	**1.27E+01ξ**	1.32E+01≠	1.28E+01ξ	1.31E+01 ~
	Std	2.12E−01	1.31E−01	**3.00E−01**	1.84E−01	3.53E−01	2.21E−01
	P value	–	4.29E−09	**1.21E−06**	3.47E−02	7.10E−04	9.20E−01
	H value	–	1	**1**	1	1	0
F15	Mean	3.85E+02	5.71E+02≠	3.78E+02 ~	**2.62E+02ξ**	3.38E+02 ~	3.84E+02 ~
	Std	6.28E+01	3.84E+01	6.61E+01	**8.10E+01**	1.15E+02	8.53E+01
	P value	–	6.39E−17	6.99E−01	**2.64E−07**	8.06E−02	9.75E−01
	H value	–	1	0	**1**	0	0
F16	Mean	**1.01E+02**	3.28E+02≠	1.67E+02≠	1.65E+02≠	1.95E+02≠	1.47E+02≠
	Std	**8.00E+01**	2.61E+01	6.18E+01	8.26E+01	8.81E+01	3.46E+01
	P value	–	5.46E−18	1.82E−03	7.11E−03	2.51E−04	1.07E−02
	H value	–	1	1	1	1	1
F17	Mean	**1.16E+02**	3.62E+02≠	1.59E+02≠	1.92E+02≠	1.75E+02≠	1.56E+02≠
	Std	**8.12E+01**	1.46E+01	9.15E+01	1.20E+02	9.22E+01	7.45E+01
	P value	–	1.20E−19	8.86E−04	1.21E−02	1.99E−02	2.38E−02
	H value	–	1	1	1	1	1
F18	Mean	**9.05E+02**	1.01E+03≠	9.12E+02≠	9.12E+02≠	9.11E+02≠	9.12E+02≠
	Std	**1.21E+00**	8.37E+00	3.72E+00	3.11E+00	3.60E+00	3.48E+00
	P value	–	7.26E−47	6.38E−12	1.08E−15	8.05E−12	6.47E−14
	H value	–	1	1	1	1	1
F19	Mean	**9.04E+02**	1.01E+03≠	9.12E+02≠	9.11E+02≠	9.11E+02≠	9.12E+02≠
	Std	**6.22E−01**	7.99E+00	3.56E+00	3.52E+00	2.91E+00	4.15E+00
	P value	–	6.37E−48	1.04E−13	1.91E−13	1.23E−15	4.99E−12
	H value	–	1	1	1	1	1
F20	Mean	**9.04E+02**	1.00E+03≠	9.13E+02≠	9.11E+02≠	9.11E+02≠	9.14E+02≠
	Std	**3.53E−01**	8.72E+00	4.29E+00	3.81E+00	2.59E+00	4.45E+00
	P value	–	7.96E−46	1.66E−13	2.98E−12	1.14E−17	1.97E−14
	H value	–	1	1	1	1	1
F21	Mean	**5.00E+02**	1.13E+03≠	5.24E+02≠	5.00E+02 ~	5.89E+02≠	5.39E+02≠
	Std	**2.24E−13**	2.14E+01	8.31E+01	1.65E−11	1.99E+02	1.46E+02
	P value	–	1.57E−65	1.55E−06	2.55E−05	2.93E−02	1.86E−03
	H value	–	1	1	1	1	1
F22	Mean	**8.94E+02**	1.07E+03≠	9.64E+02≠	9.52E+02≠	9.65E+02≠	9.57E+02≠
	Std	**1.34E+01**	2.02E+01	4.42E+01	3.14E+01	3.76E+01	4.03E+01
	P value	–	1.91E−36	8.10E−10	3.82E−11	9.63E−12	1.31E−09
	H value	–	1	1	1	1	1
F23	Mean	5.50E+02	1.13E+03≠	5.50E+02 ~	5.76E+02 ~	5.36E+02 ~	**5.34E+02 ~**
	Std	8.07E+01	3.01E+01	8.08E+01	1.48E+02	6.07E+00	**1.32E−02**
	P value	–	4.56E−35	9.99E−01	4.54E−01	3.79E−01	**3.23E−01**
	H value	–	1	0	0	0	**0**
F24	Mean	**2.00E+02**	1.12E+03≠	**2.00E+02 ~**	**2.00E+02 ~**	**2.00E+02 ~**	**2.00E+02 ~**
	Std	**8.67E−13**	2.84E+01	**3.48E−10**	**6.24E−11**	**2.51E−10**	**6.07E−11**
	P value	**–**	1.77E−67	**1.05E−02**	**5.66E−05**	**5.56E−02**	**8.54E−05**
	H value	**–**	1	**1**	**1**	**0**	**1**

Table 1 (continued)

Func	Results	MFO–HFA	AMFO	GMFO	CMFO	LMFO	OMFO
F25	Mean	**9.84E+02**	1.27E+03≠	9.88E+02≠	9.86E+02 ∼	9.88E+02 ∼	9.86E+02 ∼
	Std	**5.46E+00**	8.56E+00	9.60E+00	1.16E+01	1.04E+01	9.28E+00
	P value	–	2.27E−64	3.55E−02	3.86E−01	5.11E−02	1.95E−01
	H value	–	1	1	0	0	0
≠/ξ/ ∼		–	21/1/3	16/3/6	16/3/6	16/5/4	16/2/7

better solution. The above process will be repeated until the termination criteria are met.

3 Adaptive moth flame optimization with historical flame archive strategy

The main advantage of original MFO algorithm is that the spiral search mechanism is simple and efficient to optimize some practical problems. However, for some complex problems, especially the multi-modal and high dimensional problems, it may be premature and the obtained solution precision will be poor. The flame population in MFO algorithm increases the risk of premature. Besides, the classical flame matching mechanism in MFO is inefficient and does not make use of information of top flame. To solve above problems, this paper proposes two effective mechanisms (adaptive historical flame archive strategy and new flame number updating mechanism) described in detail as follows.

3.1 Adaptive historical flame archive strategy

The effect of the flame population that is composed of the best flame found so far, is crucial to guarantee the search ability of MFO. However, in the later stage of evolutionary search, the diversity of flame population is very poor so that moth population easy to fall into the trap of local optimal solution. In order to enhance the diversity of flame population, an adaptive historical flame archive strategy is designed to avoid moth population premature. This flame archive is described as follows.

$$\mathrm{F} = \begin{bmatrix} pF_1 \\ pF_2 \\ \vdots \\ pF_N \end{bmatrix} \tag{12}$$

where pF_i denotes the personal historical optimal solution of the ith moth.

The updating strategy of historical flame archive can be described by Fig. 3, and it can be formulated as follows.

$$pF_i = \begin{cases} M_i, & if\ OM_i < OpF_i \\ pF_i, & otherwise \end{cases} \tag{13}$$

where OpF_i is the fitness value of the historical optimal solution of ith moth.

The advantage of the historical flame archive will be clearly illustrated in Fig. 4. The labels 1, 2 and 3 in the Fig. 4 denote the positions of index 1, 2 and 3 flames, respectively. Meanwhile, labels 1′, 2′ and 3′ are the positions of index 1, 2 and 3 moths after an update, respectively. Based on the definition of MFO algorithm, flames are the best position found so far. Therefore, the new flames will be the positions labeled 3′, 3 and 1′, which will make the population lose the information of the global optimal solution and trap into the local optima. The adaptive historical flame archive strategy described above is used to ensure that the information of the global optima can be maintained in the process of search, because the ith flame will be replaced by the ith moth only when the fitness value of the ith moth is better than that of the ith flame. In the above example, the positions of index 2, 3 and 1′ will be viewed as the new flames so that index 2 with the information of global optima is kept.

3.2 Top flame randomly matching mechanism

The flame matching mechanism in the original MFO is inefficient and does not make use of information of top flame, because many moths of the population in the middle and later stage of search are searching around the *Flame_no*th flame and the *Flame_no*th flame not provide the best direction information (i.e., the *Flame_no*th flame is not the best individual of flame population). To solve the problem, each moth randomly chooses one of top $q\%$ flames for searching the solution space, which is defined as follows.

$$SF = sort(F) \tag{14}$$

$$F_i = SF_{rand*q\%*N} \tag{15}$$

Table 2 Results of solution accuracy obtained by six compared algorithms

Func	Results	MFO–HFA	MFO	PSO	CLPSO	DE	ACoDE
F1	Mean	2.11E−17	1.71E−23ξ	3.40E+03≠	1.55E−10≠	**8.08E−30ξ**	2.12E−10≠
	Std	3.76E−17	8.03E−23	1.69E+03	4.92E−11	**4.04E−29**	6.68E−11
	P value	–	7.19E−03	2.28E−13	1.23E−20	**7.19E−03**	1.05E−20
	H value	–	1	1	1	**1**	1
F2	Mean	1.27E+00	6.84E−02ξ	7.66E+02≠	3.95E+03≠	**2.15E−05ξ**	4.73E−02ξ
	Std	1.87E+00	7.24E−02	8.28E+02	6.77E+02	**1.61E−05**	3.78E−02
	P value	–	2.33E−03	2.91E−05	3.46E−32	**1.36E−03**	1.97E−03
	H value	–	1	1	1	**1**	1
F3	Mean	2.31E+06	2.36E+06 ~	8.37E+06≠	2.21E+07≠	4.01E+05ξ	**9.99E+04ξ**
	Std	1.18E+06	9.53E+05	1.00E+07	6.40E+06	2.38E+05	**1.10E+05**
	P value	–	8.80E−01	4.14E−03	6.04E−20	3.08E−10	**2.65E−12**
	H value	–	0	1	1	1	**1**
F4	Mean	9.69E+01	1.15E+03≠	1.48E+03≠	1.08E+04≠	**7.74E−03ξ**	3.10E+00ξ
	Std	1.41E+02	1.13E+03	1.45E+03	1.84E+03	**6.51E−03**	2.33E+00
	P value	–	2.60E−05	1.94E−05	4.75E−32	**1.22E−03**	1.69E−03
	H value	–	1	1	1	**1**	1
F5	Mean	2.90E+04	**2.68E+04ξ**	3.11E+04≠	2.91E+04≠	2.90E+04 ~	2.90E+04 ~
	Std	3.53E+01	**2.88E+03**	1.83E+03	2.73E+01	1.98E−09	9.78E−01
	P value	–	**4.27E−04**	6.80E−07	3.05E−13	1.52E−06	1.11E−05
	H value	–	**1**	1	1	1	1
F6	Mean	1.02E+02	1.22E+02≠	2.73E+08≠	2.85E+01ξ	**3.69E−02ξ**	2.40E+01ξ
	Std	1.79E+02	2.31E+02	3.82E+08	1.14E+01	**5.07E−02**	1.48E+00
	P value	–	7.34E−06	7.92E−04	4.46E−02	**6.19E−03**	3.32E−02
	H value	–	1	1	1	**1**	1
F7	Mean	4.70E+03	**4.65E+03 ~**	4.85E+03≠	4.70E+03 ~	4.70E+03 ~	4.70E+03 ~
	Std	2.53E−12	**1.42E+02**	2.62E+02	3.56E−08	1.34E−12	2.13E−10
	P value	–	**1.48E−01**	4.64E−03	1.74E−22	1.00E+00	2.27E−31
	H value	–	**0**	1	1	0	1
F8	Mean	2.10E+01	**2.02E+01ξ**	2.09E+01ξ	2.09E+01ξ	2.09E+01 ~	2.09E+01 ~
	Std	4.58E−02	**7.14E−02**	5.77E−02	5.58E−02	5.35E−02	6.74E−02
	P value	–	**8.15E−42**	1.92E−03	4.22E−02	2.89E−01	4.11E−01
	H value	–	**1**	1	1	0	0
F9	Mean	2.42E+01	4.92E+01≠	5.87E+01≠	**3.08E−04ξ**	1.10E+02≠	6.13E−04ξ
	Std	6.66E+00	1.74E+01	1.86E+01	**1.24E−04**	2.62E+01	2.77E−04
	P value	–	2.12E−08	1.73E−11	**3.42E−23**	8.74E−21	3.42E−23
	H value	–	1	1	**1**	1	1
F10	Mean	**6.73E+01**	1.11E+02≠	1.00E+02≠	1.34E+02≠	1.77E+02≠	1.62E+02≠
	Std	**5.40E+01**	3.69E+01	2.44E+01	1.12E+01	7.38E+00	1.56E+01
	P value	–	1.48E−03	7.91E−03	2.23E−07	2.11E−13	5.50E−11
	H value	–	1	1	1	1	1
F11	Mean	**1.10E+01**	2.53E+01≠	2.07E+01≠	2.71E+01≠	3.96E+01≠	3.30E+01≠
	Std	**1.99E+00**	4.07E+00	2.98E+00	1.30E+00	8.24E−01	1.68E+00
	P value	–	1.16E−20	5.89E−18	4.10E−35	7.00E−49	1.47E−39
	H value	–	1	1	1	1	1
F12	Mean	9.83E+05	**5.40E+03ξ**	7.46E+03ξ	1.18E+05ξ	9.32E+05 ~	1.55E+05ξ
	Std	1.30E+05	**4.66E+03**	8.40E+03	1.50E+04	1.08E+05	1.87E+04
	P value	–	**2.56E−37**	3.02E−37	9.72E−35	1.38E−01	8.46E−34
	H value	–	**1**	1	1	0	1

Table 2 (continued)

Func	Results	MFO–HFA	MFO	PSO	CLPSO	DE	ACoDE
F13	Mean	**2.85E+00**	7.10E+00≠	2.97E+00≠	3.14E+00≠	1.51E+01≠	7.72E+00≠
	Std	**4.11E−01**	2.04E+00	7.30E−01	3.22E−01	1.06E+00	5.58E−01
	P value	–	1.40E−13	4.90E−02	8.71E−03	1.33E−44	6.64E−36
	H value	–	1	1	1	1	1
F14	Mean	1.31E+01	1.34E+01≠	**1.20E+01ξ**	1.29E+01ξ	1.32E+01≠	1.33E+01≠
	Std	2.12E−01	2.46E−01	**6.12E−01**	1.51E−01	2.41E−01	1.07E−01
	P value	–	2.06E−05	**6.16E−11**	1.17E−02	1.47E−02	3.62E−04
	H value	–	1	**1**	1	0	1
F15	Mean	3.85E+02	3.72E+02 ~	5.10E+02≠	**1.02E+02ξ**	3.80E+02 ~	4.08E+02≠
	Std	6.28E+01	1.00E+02	1.06E+02	**2.84E+01**	1.04E+02	2.77E+01
	P value	–	5.92E−01	5.91E−06	**2.01E−25**	8.51E−01	9.46E−12
	H value	–	0	1	**1**	0	0
F16	Mean	**1.01E+02**	1.60E+02≠	3.49E+02≠	1.91E+02≠	2.05E+02≠	1.83E+02≠
	Std	**8.00E+01**	6.67E+01	1.89E+02	2.64E+01	2.97E+01	2.02E+01
	P value	–	6.83E−03	2.02E−07	2.22E−06	1.55E−07	8.55E−06
	H value	–	1	1	1	1	1
F17	Mean	**1.16E+02**	1.78E+02≠	2.84E+02≠	2.62E+02≠	2.30E+02≠	2.30E+02≠
	Std	**8.12E+01**	1.04E+02	1.64E+02	2.65E+01	4.67E+01	8.83E+00
	P value	–	2.26E−02	3.26E−05	3.35E−11	1.76E−07	8.78E−09
	H value	–	1	1	1	1	1
F18	Mean	9.05E+02	9.11E+02≠	9.40E+02≠	9.11E+02≠	**9.03E+02ξ**	9.06E+02≠
	Std	1.21E+00	3.74E+00	2.59E+01	9.33E−01	**1.27E−01**	1.56E−01
	P value	–	4.29E−11	1.72E−08	2.05E−24	**1.08E−05**	2.17E−07
	H value	–	1	1	1	**1**	1
F19	Mean	9.04E+02	9.10E+02≠	9.42E+02≠	9.11E+02≠	**9.03E+02ξ**	9.06E+02≠
	Std	6.22E−01	2.30E+00	2.34E+01	9.25E−01	**2.50E−01**	2.71E−01
	P value	–	3.25E−15	2.34E−10	5.15E−32	**6.69E−07**	4.66E−18
	H value	–	1	1	1	**1**	1
F20	Mean	9.04E+02	9.08E+02≠	9.46E+02≠	9.11E+02≠	**9.03E+02ξ**	9.06E+02≠
	Std	3.53E−01	2.27E+01	3.04E+01	7.55E−01	**1.57E−01**	3.13E−01
	P value	–	4.01E−02	1.24E−08	4.66E−39	**8.48E−11**	4.48E−26
	H value	–	1	1	1	**1**	1
F21	Mean	**5.00E+02**	5.12E+02≠	1.06E+03≠	**5.00E+02 ~**	**5.00E+02 ~**	**5.00E+02 ~**
	Std	**2.24E−13**	6.00E+01	1.01E+02	**3.24E−08**	**9.91E−14**	**4.56E−11**
	P value	–	3.22E−02	3.43E−31	**2.22E−03**	**2.72E−05**	**1.07E−11**
	H value	–	1	1	**1**	**1**	**1**
F22	Mean	8.94E+02	9.58E+02≠	9.14E+02≠	9.54E+02≠	**8.73E+02ξ**	9.03E+02≠
	Std	1.34E+01	3.42E+01	4.99E+01	9.87E+00	**1.41E+01**	1.03E+01
	P value	–	1.53E−11	5.18E−07	5.20E−23	**2.51E−06**	7.90E−03
	H value	–	1	1	1	**1**	1
F23	Mean	5.50E+02	5.67E+02≠	1.10E+03≠	**5.34E+02 ~**	**5.34E+02 ~**	**5.34E+02 ~**
	Std	8.07E+01	1.14E+02	5.73E+01	**1.45E−04**	**3.52E−04**	**2.02E−04**
	P value	–	5.51E−03	3.66E−31	**3.22E−01**	**3.22E−01**	**3.22E−01**
	H value	–	1	1	**0**	**0**	**0**
F24	Mean	**2.00E+02**	**2.00E+02 ~**	9.52E+02≠	**2.00E+02 ~**	**2.00E+02 ~**	**2.00E+02 ~**
	Std	**8.67E−13**	**1.41E−12**	4.20E+01	**2.33E−03**	**2.90E−14**	**1.81E−10**
	P value	**–**	**1.10E−01**	4.56E−55	**1.53E−02**	**5.47E−22**	**5.31E−11**
	H value	**–**	**0**	1	**1**	**1**	**1**

Table 2 (continued)

Func	Results	MFO–HFA	MFO	PSO	CLPSO	DE	ACoDE
F25	Mean	9.84E+02	**9.82E+02 ~**	1.17E+03≠	1.01E+03≠	9.89E+02≠	1.00E+03≠
	Std	5.46E+00	**7.51E+00**	6.23E+01	5.17E+00	3.61E+00	3.93E+00
	P value	–	**5.30E−01**	2.92E−19	2.33E−24	2.00E−04	1.77E−19
	H value	–	**0**	1	1	1	1
≠/ξ/ ~		–	15/5/5	22/3/0	15/4/6	8/9/8	10/6/9

where $sort(F)$ refers that the flames of flame population are ranked from small to large according to their fitness value, and F_i denotes the flame corresponding to the ith moth. *rand* is a random in range [0, 1], and q is a control parameter that defines the number of top flames. N stands for the size of flame population.

The pseudo-code of the MFO–HFA is summarized in Algorithm 1, where the modification of MFO–HFA is bold to show clearly.

As previously analyzed, the complete flowchart of the MFO–HFA algorithm is given in Fig. 5, where the modification of MFO–HFA is bold for clarity.

3.3 Flowchart and pseudo-code of MFO–HFA

Algorithm1. Pseudo-code of the MFO-HFA

```
Begin
  Initializing the parameters;
  Initializing uniformly random population;
  Calculating the fitness values of moths;
  archive.F = M; archive.OF = OM;
  gen = 1;
  while not meet termination condition
    for i = 1 to N do
      Updating the parameter r and t;
      Selecting the flame F_i from archive.F according to Eq. (14) and (15);
      Updating the moth's position using Eq. (8) and (9);
    end for
    Boundary constraint;
    Updating the archive.F according to Eq. (13);
    gen = gen + 1;
  end while
  output: optimal solution from archive.F;
end
```

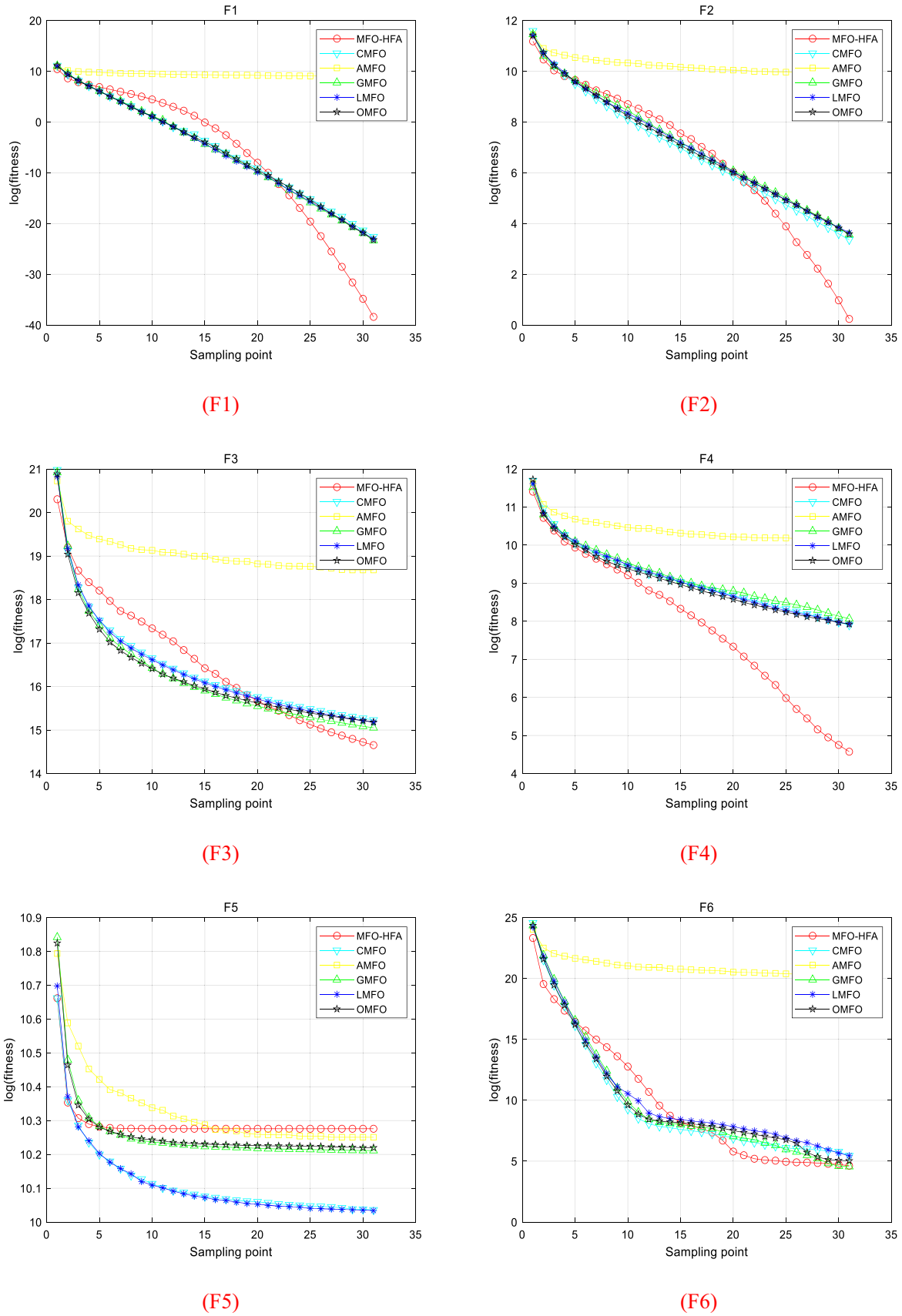
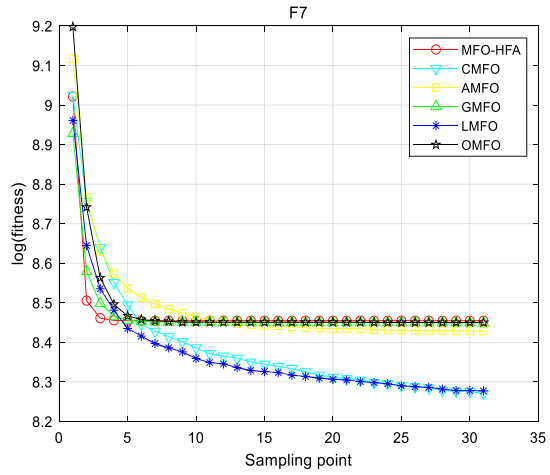
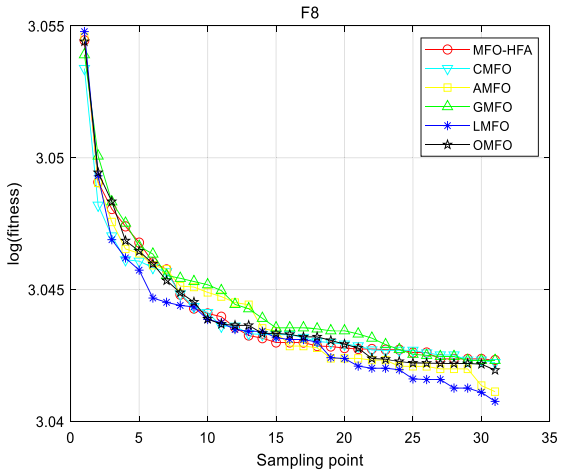


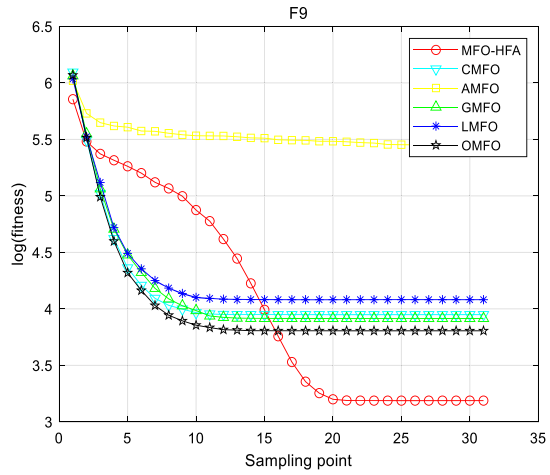
Fig. 6 Convergence performance of the six compared algorithms (i.e., MFO–HFA, CMFO, GMFO, LMFO, AMFO and OMFO) on 25 functions



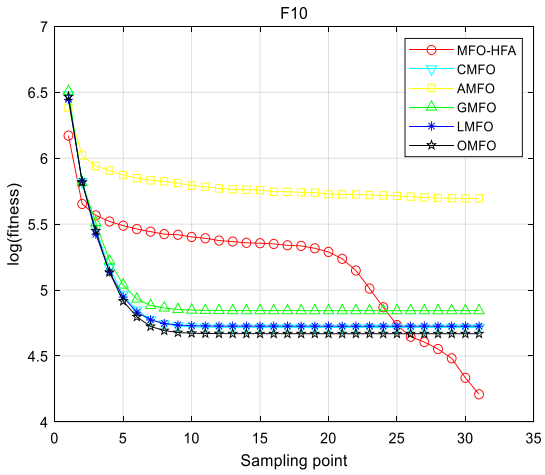
(F7)



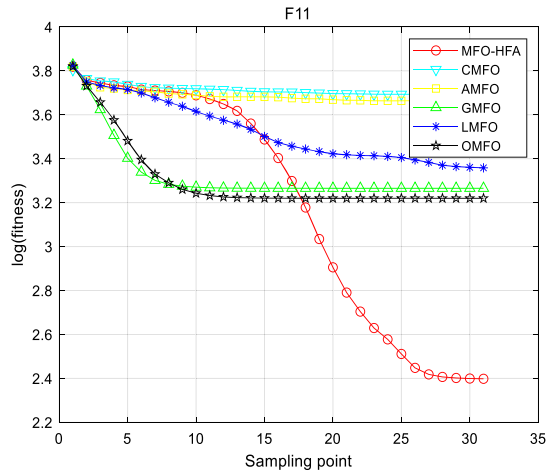
(F8)



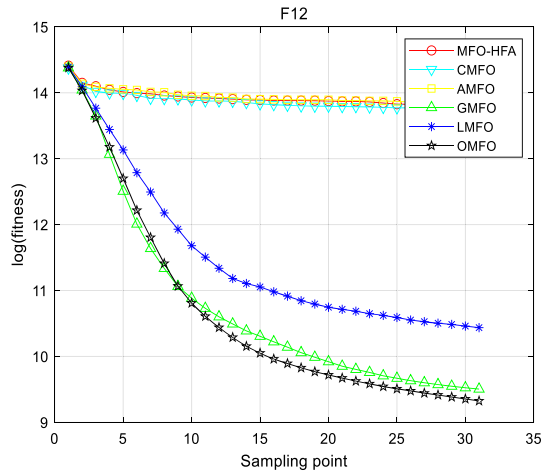
(F9)



(F10)

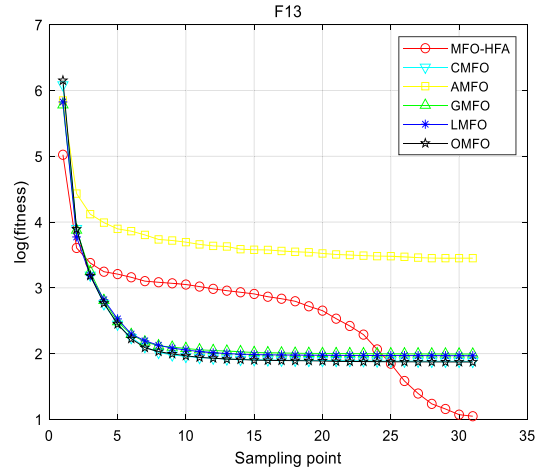


(F11)

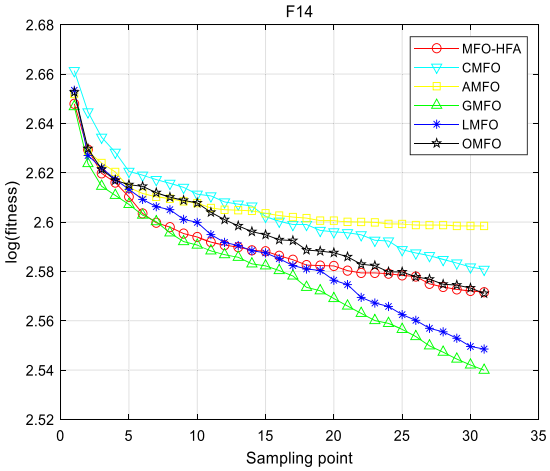


(F12)

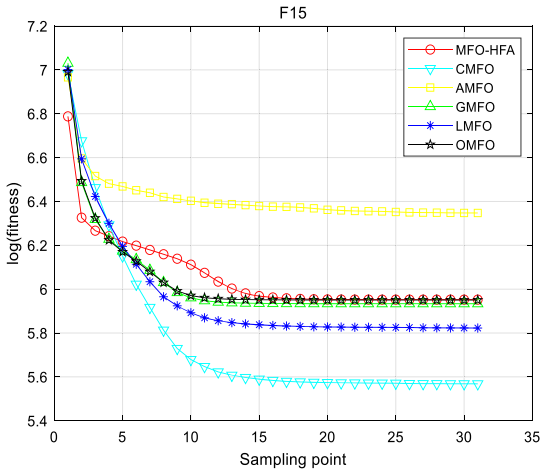
◀**Fig. 6** continued



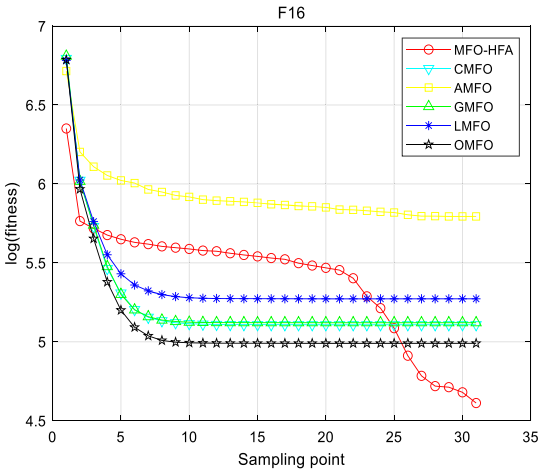
(F13)



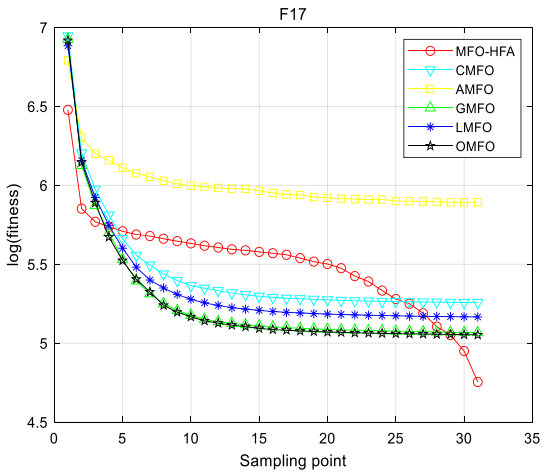
(F14)



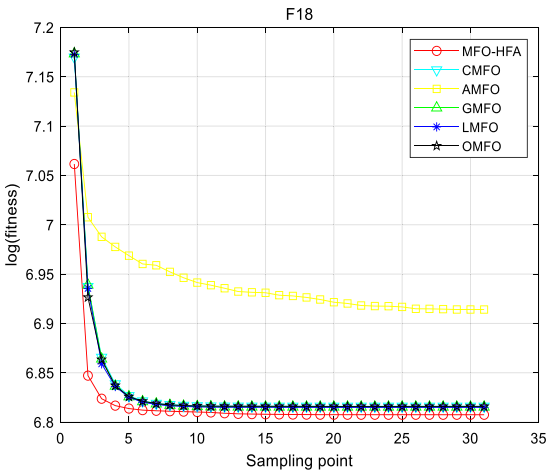
(F15)



(F16)

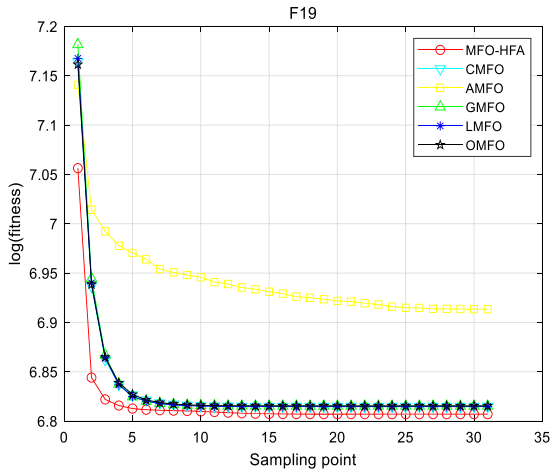


(F17)

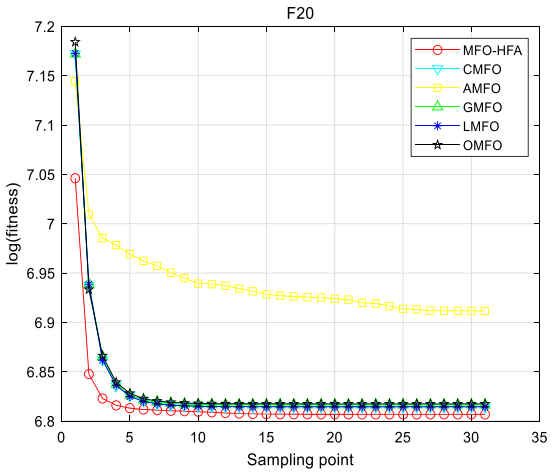


(F18)

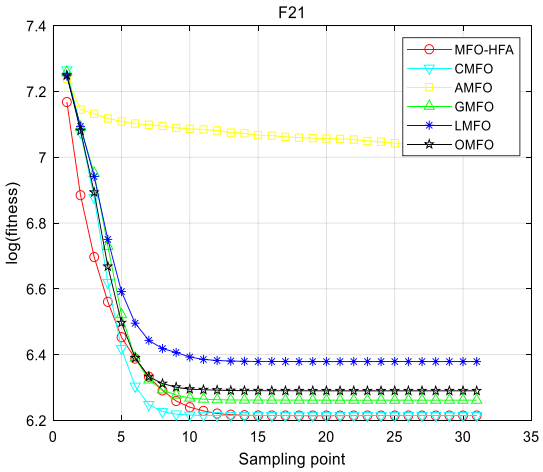
◀**Fig. 6** continued



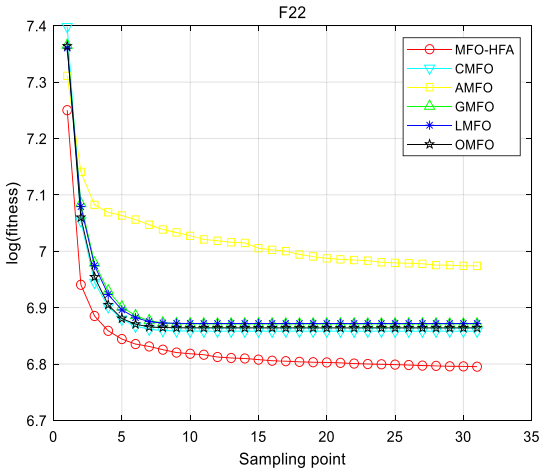
(F19)



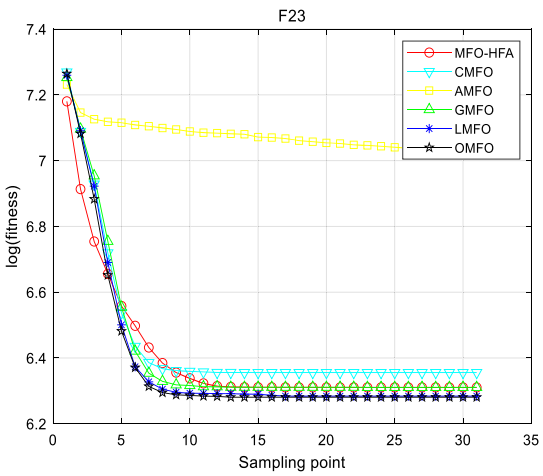
(F20)



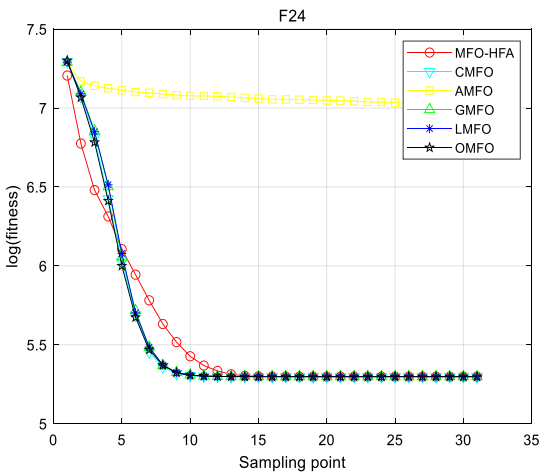
(F21)



(F22)

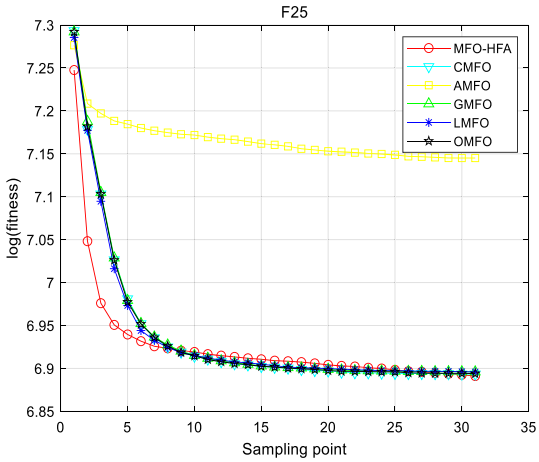


(F23)



(F24)

◀**Fig. 6** continued



(F25)

◀**Fig. 6** continued

4 Benchmark function optimization problems

The numerical benchmark functions of CEC 2005 (Suganthan et al. 2005) are used to test the performance of MFO–HFA compared with classic MFO algorithm, other variants of MFO and some state-of-the-art optimization algorithms (i.e., DE, adaptive CoDE (ACoDE) (Wang et al. 2011), PSO and comprehensive learning particle swarm optimizer (CLPSO) (Liang et al. 2006)). For a fair comparison, all simulations are carried out on the same physical environment with MATLAB 2018b, and each algorithm is independently run 25 times with D*10,000 function evaluations (FES) for reducing statistical errors. The Wilcoxon's rank sum test at a 5% significance level was used to calculate statistically reliable results.

4.1 Benchmark functions

The 25 benchmark functions are used to test the performance of the MFO–HFA proposed by this paper, proposed in the CEC2005 (Suganthan et al. 2005) special session on real-parameter optimization. F1–F5 of CEC 2005 are continuous unimodal functions while F6–F14 are multimodal and have a significant number of local minima. Besides, F15–F25 are hybrid composition functions.

The dimension of the problems (i.e., decision variables) is set to 30 for all the 25 functions. In this experiment, the mean value and standard deviation of the function error value $(f(gbest) - f(X^*))$ are recorded for testing the performance of each algorithm, where $gbest$ is the best solution found by the algorithm in a run and X^* is the theoretical global optimum of the benchmark functions.

4.2 Parameter settings of comparative algorithms

MFO–HFA is compared with five other variants of MFO algorithm, i.e., AMFO (Zhao et al. 2018), GMFO (Xu et al. 2019a), CMFO (Xu et al. 2019a), LMFO (Xu et al. 2019a), OMFO (Apinantanakon and Sunat 2017). Besides, classic MFO (Mirjalili 2015), PSO (Kennedy and Eberhart 1995), ACoDE (Wang et al. 2011), CLPSO (Liang et al. 2006) and DE (Storn and Price 1997) are used as a comparison algorithm to evaluate the effect of MFO–HFA algorithm. The parameters of the MFO–HFA algorithm are set as follows. The size of moth population is 100, and the size of the historical flame archive equal to the size of population. The parameter q is set as 0.2, according to the sensitivity analysis of the parameter q in Sect. 4.5. The population size of other algorithms also is 100, and other parameters of comparison algorithms are the same with their original papers.

4.3 Experimental results

4.3.1 Comparisons on solution accuracy

The results of solution accuracy are shown in both Tables 1 and 2 in terms of the mean optimal solutions and the standard deviation of the solutions, which are obtained by each algorithm with 25 independent runs and 300,000 times fitness evaluation on 25 benchmark functions.

In each row of Tables 1 and 2, the average values over 25 independent runs are listed in the first line, and the standard deviations are given in the second line. The P value and H value of nonparametric statistical test with a significance level $\alpha = 0.05$ are presented in the third and fourth lines. The symbol '+' is tagged in the back of the mean value yielded by the algorithm that is significantly worse than MFO–HFA algorithm. If MFO–HFA is worse than other algorithms, a 'ξ' is added in the back of the mean value of corresponding algorithm. The symbol ' ∼ ' indicates that there is no significant difference between MFO–HFA and the compared algorithm. At the last row of the table, a summary of total number of '+', 'ξ' and ' ∼ ' is presented. In addition, the best results are bold to show clearly.

It can be seen from the Table 1 that MFO–HFA obtains the best performance on 17 test functions, and is poor on 9 functions (F5–F8, F12, F14, F15 and F23). AMFO yields the best results on function F8, and GMFO obtains the best results on 3 functions (F6, F14 and F24). CMFO gains the best results on 4 functions (F5, F7, F15 and F24), and LMFO obtains the best results on 3 functions (F5, F8 and

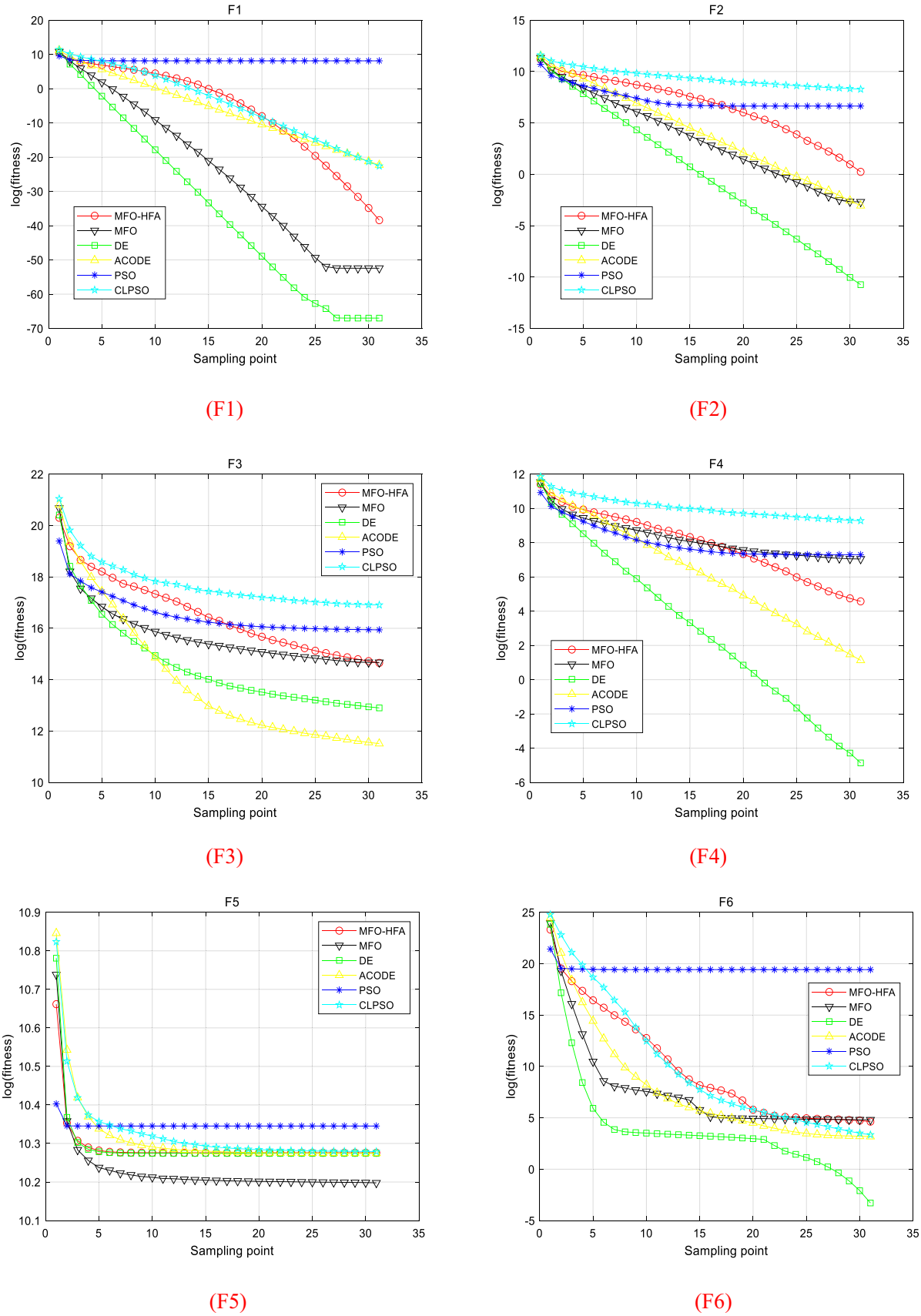
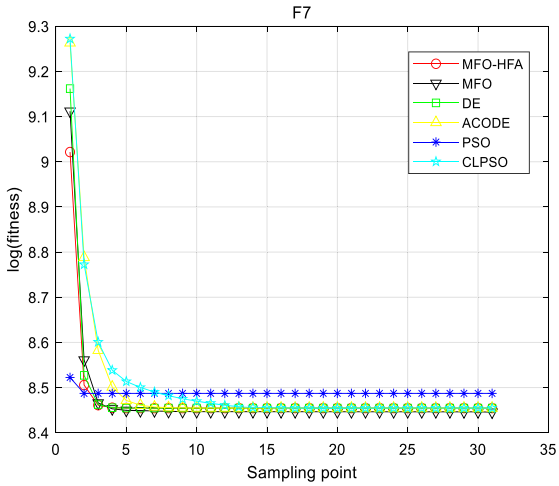
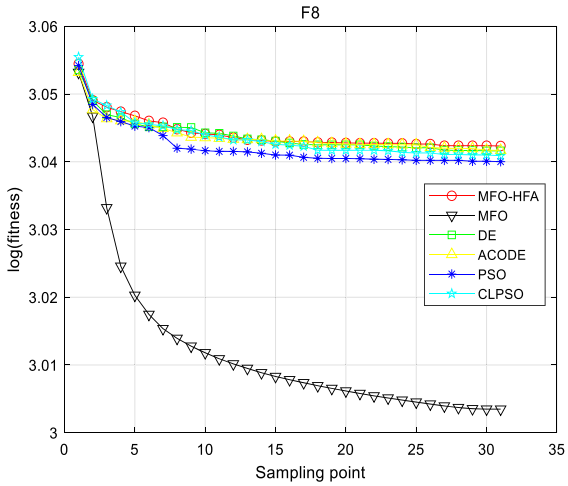


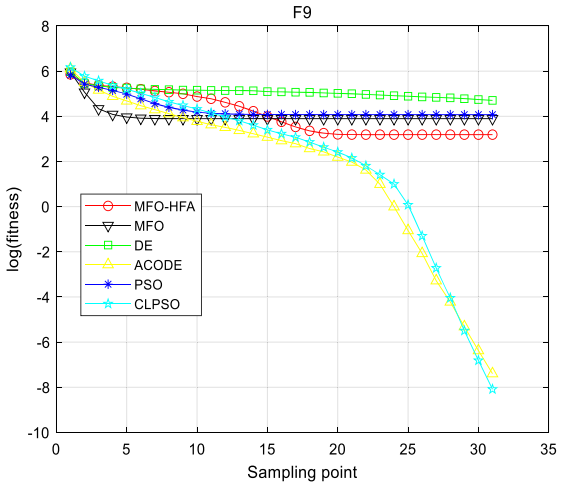
Fig. 7 Convergence performance of the six compared algorithms (i.e., MFO–HFA, MFO, ACoDE, DE, CLPSO and PSO) on 25 functions



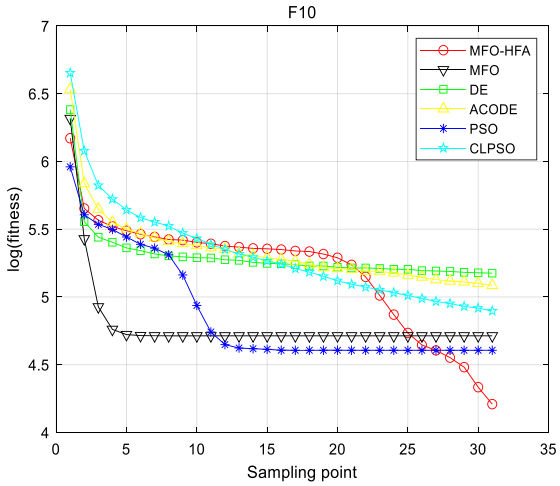
(F7)



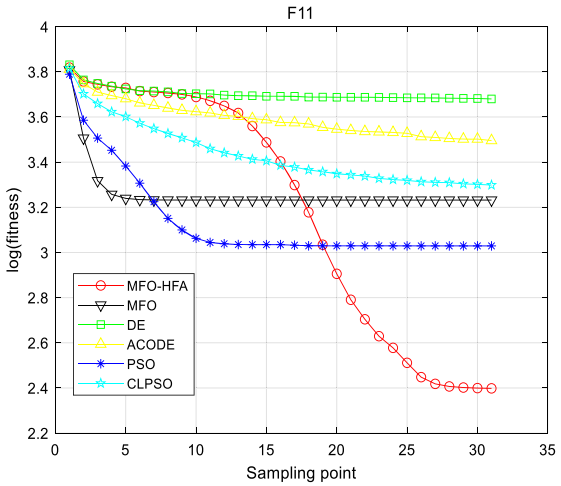
(F8)



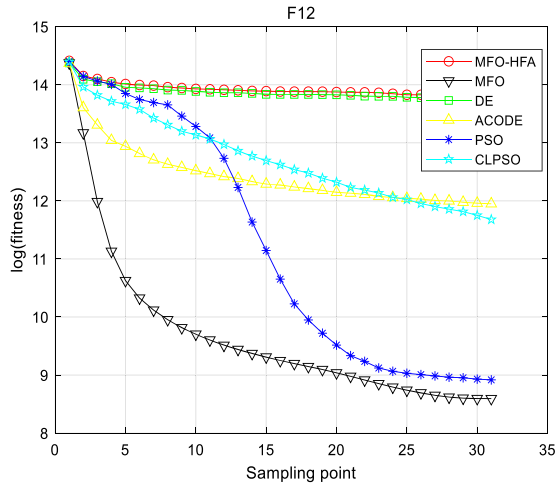
(F9)



(F10)

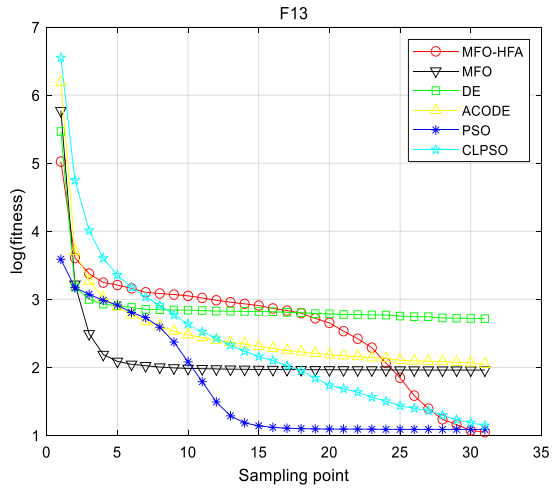


(F11)

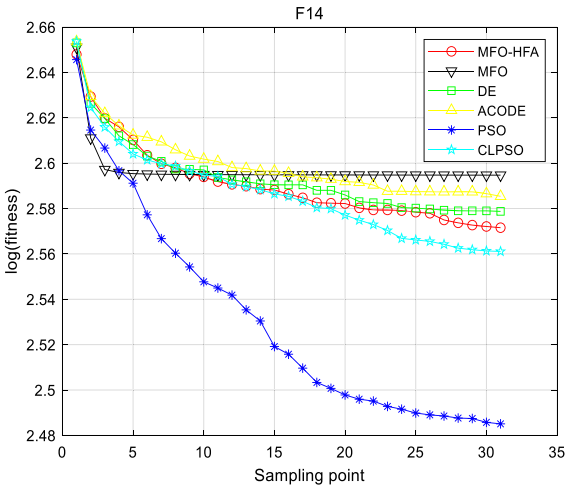


(F12)

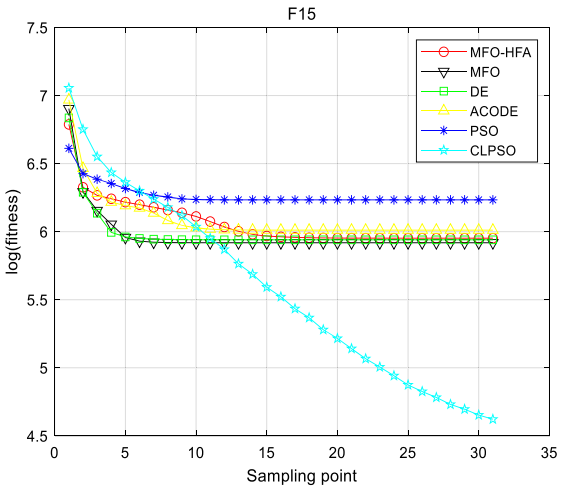
◀**Fig. 7** continued



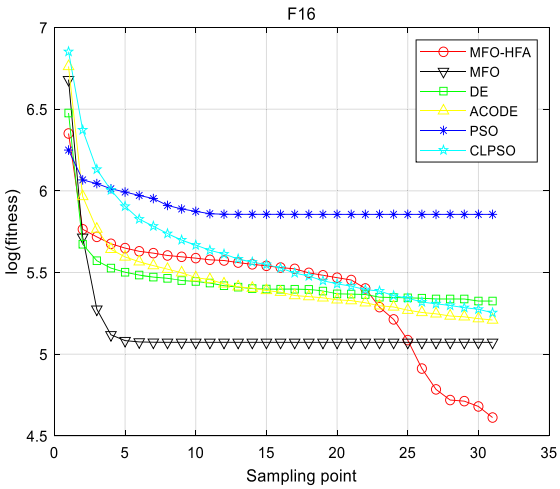
(F13)



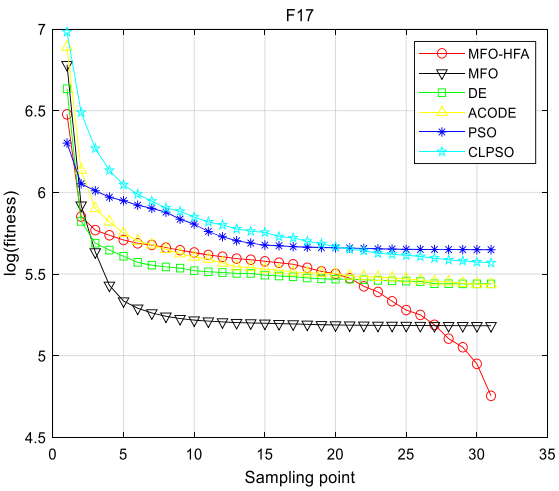
(F14)



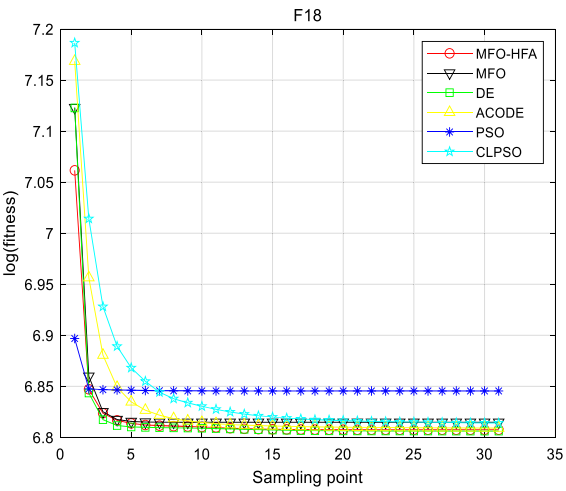
(F15)



(F16)

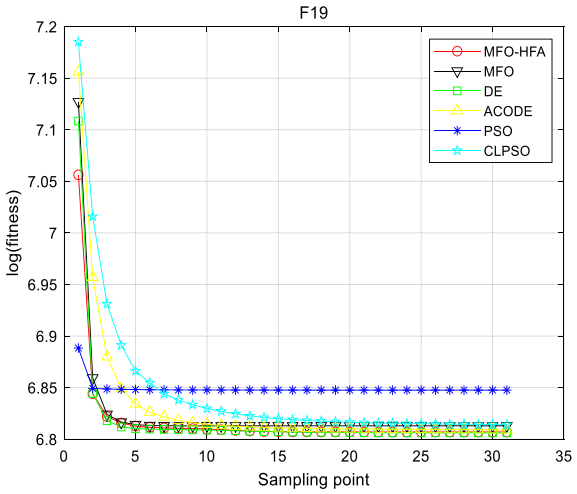


(F17)

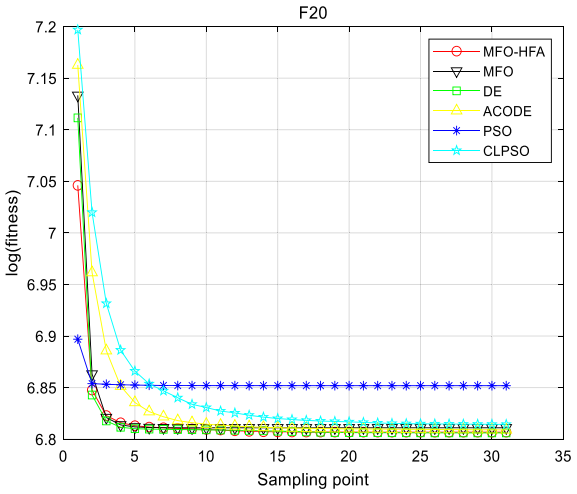


(F18)

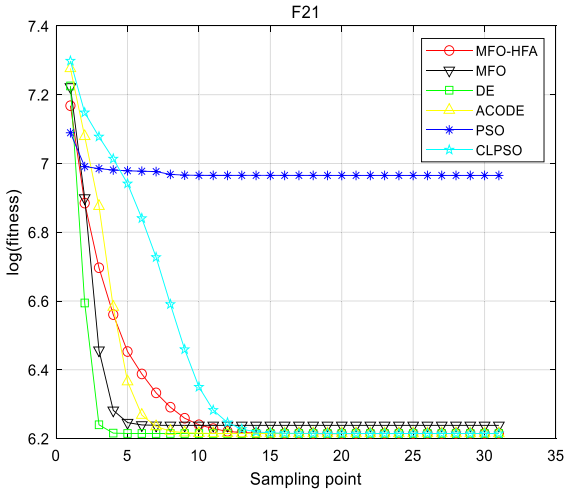
◀**Fig. 7** continued



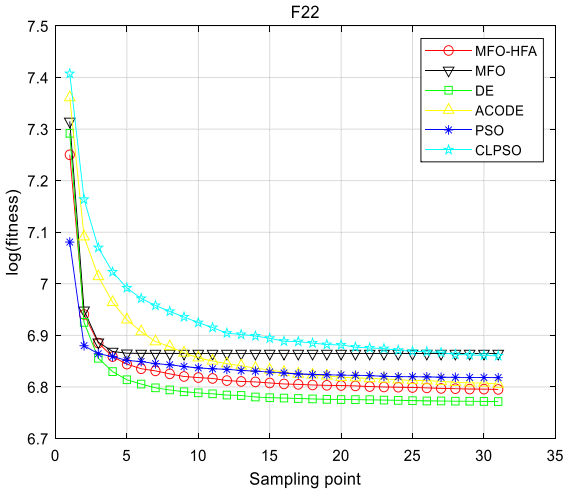
(F19)



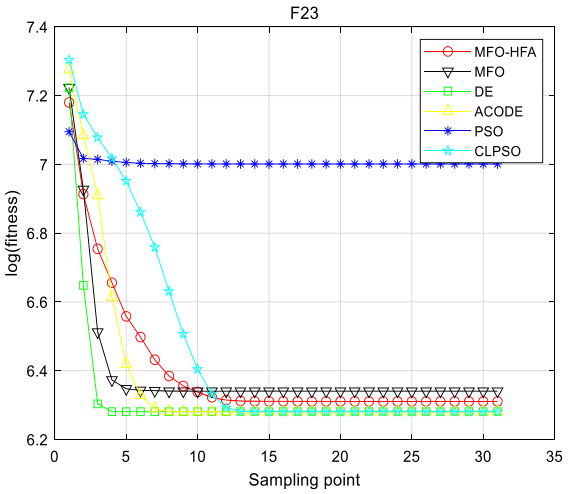
(F20)



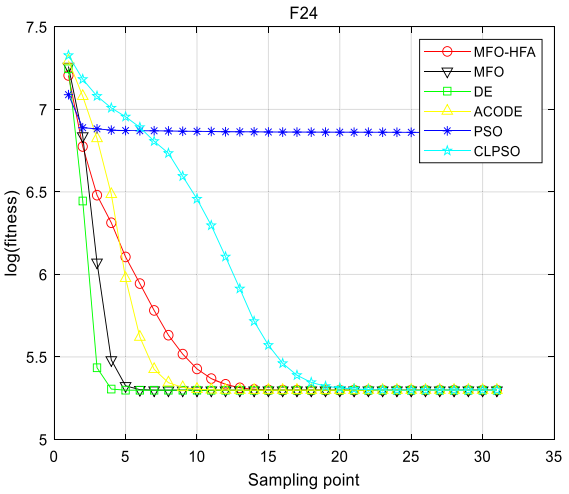
(F21)



(F22)

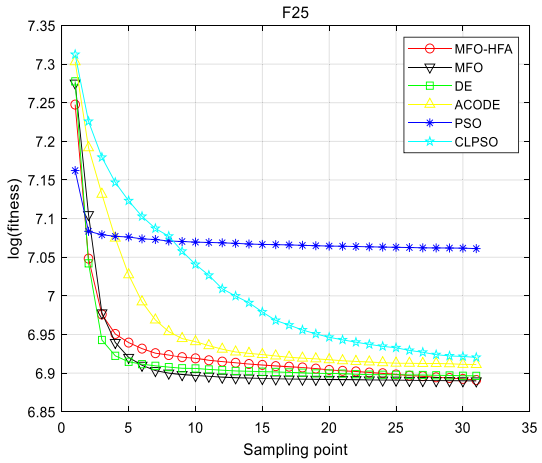


(F23)



(F24)

◀**Fig. 7** continued



◀**Fig. 7** continued

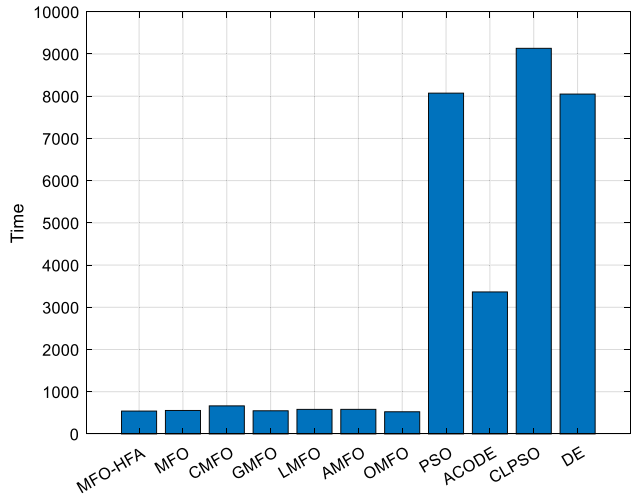


Fig. 8 Mean CPU time of compared algorithms on 25 benchmark functions of CEC2005

Table 3 Average rankings obtained by four algorithms on 25 benchmark functions of CEC2005

Algorithm	Ranking
MFO	3.14
MFO-A	2.24
MFO-T	2.44
MFO–HFA	**2.18**

Table 4 Average rankings obtained by the MFO–HFA with different q value on 25 benchmark functions of CEC2005

Algorithm	Ranking
q = 0.1	3.26
***q* = 0.2**	**2.82**
q = 0.3	3.7
q = 0.4	4.66
q = 0.5	4.66
q = 0.6	5.4
q = 0.7	6.2
q = 0.8	7.64
q = 0.9	7.9
q = 1.0	8.76

F24). OMFO gets best performance on 4 functions (F8, F12, F23 and F24).

It can be clearly observed from Table 2 that the original MFO algorithm has the best performance on 6 functions (F5, F7, F8, F12, F24 and F25), compared with MFO–HFA, PSO, CLPSO, DE and ACoDE. Furthermore, CLPSO algorithm obtains the best performance on five functions, i.e., F9, F15, F21, F23 and F24. The state-of-the-art DE algorithm outperformances other algorithms on 9 benchmark functions which are 3 unimodal functions (F1, F2 and F4), 1 multimodal function (F6) and 5 hybrid

Table 5 Evaluation of all algorithms on NSL-KDD dataset

Algorithms	Precision	Recall	F1-Score	Accuracy
MFO–HFA	0.97553	0.947999	0.961561	0.964725
OMFO	0.969173	0.943619	0.955977	0.959657
GMFO	0.972536	0.93225	0.951889	0.956203
LMFO	0.972799	0.930962	0.951364	0.955743
CMFO	0.972099	0.928318	0.94966	0.954228
MFO	0.970228	0.929192	0.949151	0.953675
AMFO	0.806304	0.938296	0.851649	0.823173

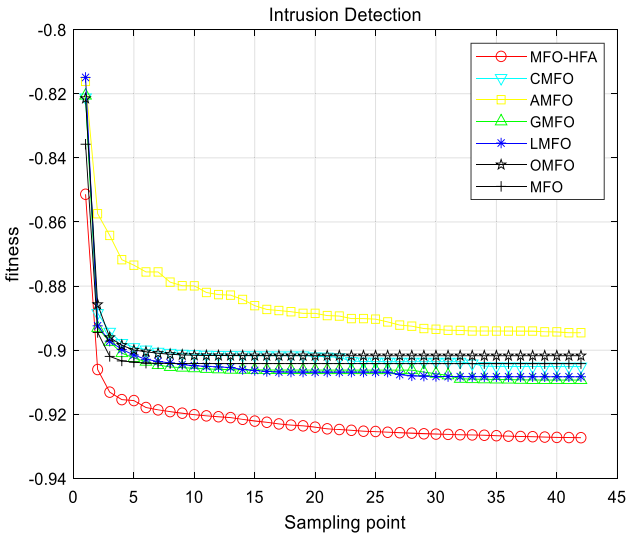


Fig. 9 Convergence performance of the seven algorithms on network intrusion detection

composition functions (F18–F24). The ACoDE gains the best results on 3 standard functions (F3, F21 and F24). The MFO–HFA obtains the best performance on 7 benchmark functions that are 3 multimodal functions (F10, F11 and F13) and 4 hybrid composition functions (F16, F27, F21 and F24).

Compared with MFO, PSO and CLPSO, MFO–HFA has achieved an overwhelming advantage. Meanwhile, the performance of MFO–HFA is similar to that of DE on 25 benchmark functions. Based on the above analysis, MFO–HFA significantly improves the solution accuracy and exploration ability of MFO. The main reason is the adaptive historical flame archive strategy of MFO–HFA has stronger ability to jump out of local optimum compared with PSO, CLPSO, DE and ACoDE.

4.3.2 The comparison results of convergence speed

In Figs. 6 and 7, the vertical axis is the natural logarithm of the mean value over independent 25 runs, and the horizontal axis is the sampling point where 31 sampling points are taken from FES = 1000 and mod (FES, 10,000) = 0.

It can be clearly seen from the Fig. 6 that MFO–HFA obtains better convergence speed and solution accuracy than other 5 variants of MFO on four unimodal functions (F1–F4), three multimodal functions (F9, F10, F11 and F13) and seven hybrid composition functions (F16–F20, F22 and F24). It proves that the methods proposed by this paper improve the convergence speed of original MFO and MFO–HFA has strong convergence ability. In addition, in early stage of evolution search, although MFO–HFA has a slower convergence speed than other five algorithms on some functions, it has strong exploration ability and has achieved better solution accuracy. It indicates that the historical flame archive strategy can improve the exploration capacity of MFO and make the moth population escapes the local optimal trap. MFO–HFA does not have a best convergence speed on five functions (F5, F7, F8, F12, F15 and F23), this may be because the local exploration ability of MFO–HFA is slightly poor.

It can be seen from Fig. 7 that MFO–HFA gains the highest convergence speed on four functions (F10, F11, F16 and F17), and MFO obtains the highest convergence speed on three functions (F5, F8 and F12). DE gets the highest convergence speed on eight benchmark functions (F1, F2, F4, F6 and F21-F24). Meanwhile, PSO only has the best convergence speed on test function F14, and CLPSO has obtained the best convergence speed on two functions (F9 and F15). These above analyses show that MFO–HFA has promising convergence speed in solving some complex problems.

4.4 Component analysis of MFO–HFA

To verify the effectiveness of the component of MFO–HFA, adaptive historical flame archive strategy is embedded in MFO (named for MFO-A), and MFO with top flame randomly matching mechanism is named for MFO-T. Meanwhile, MFO is utilized to optimize 25 benchmark functions of CEC2005, compared with MFO–HFA, MFO-A and MFO-T. The average ranking of the Friedman test of above algorithms is shown in Table 3. Friedman test (Das et al. 2011) is a non-parametric statistical test for comparison of more than two algorithms, utilized to show the differences in compared algorithms. All compared algorithms are ranked according to their average performance for each test function. It is worth noting that the KEEL software (Dukic and Dobrosavljevic 1990) is used to calculate the rankings of compared algorithms for all problems.

Table 3 clearly shows that MFO–HFA gains first performance, and its average rankings is 2.18. MFO-A obtains the second performance, and its average rankings is 2.24. Meanwhile, MFO-A is better than MFO, which indicate that adaptive historical flame archive strategy is effective. In addition, MFO-T is also better than MFO, and gains third performance. Those show that top flame randomly matching mechanism can fully use the information of top flames for improving the search ability of population. Finally, MFO–HFA is better than MFO-A and MFO-T, which demonstrate that adaptive historical flame archive strategy and top flame randomly matching mechanism are effectively integrated into MFO.

4.5 Sensitivity of the parameter q

The parameter q is a threshold to determine whether the flame is top flame. The choice of parameter q will influence the diversity of top flames, and then affect the performance of algorithms. To find out a good choice of the parameter q, MFO–HFA with different parameter $q = \{0.1, 0.2, \ldots, 1\}$ is used to optimize 25 benchmark problems of CEC2005. The other parameter settings of the algorithm are same as the settings described earlier.

Table 4 summarizes the results of average rankings of Friedman test. It clearly shows that 0.2 is best and a boundary. In the range [0.1, 0.2], the larger the q value, the better the ranking of the algorithm. In the range [0.2, 1.0], the larger the q value, almost, the worse the ranking of the algorithm. Therefore, we suggest $q = 0.2$ for the MFO–HFA.

4.6 Comparison results of time complexity

The total comparisons of average time complexity of 25 functions in one iteration about compared algorithms are shown in Fig. 8 in the form of bar plot. In Fig. 8, it is clearly shown that the mean CPU time of both MFO–HFA and MFO is similar. Additionally, MFO–HFA is further better than PSO, ACODE, CLPSO and DE, which indicate that the time complexity of MFO–HFA is acceptable.

5 Rule-based network intrusion detection problem

Network Intrusion Detection System (NIDS) is designed to identify and prevent the misuse of the computer networks. Most of the current IDSs are rule-based, and their performance significantly depends on sets of pre-defined rule that are provided by experts or automatically created by system. Therefore, the update of rule is critical to rule-based IDS. However, the update of rule is a nonlinear, non-differentiable and non-separable complex problem. To verify the performance of MFO–HFA on real-world engineer optimization problems, MFO–HFA is utilized to optimize the rule updating of network intrusion detection. In this section, NSL-KDD dataset (Meena and Choudhary 2017; NSL-Kdd dataset) is used to train individuals and test individuals of MFO–HFA and the compared intelligence algorithms.

5.1 Rule-based network intrusion detection method

The intelligent algorithm-based network intrusion detection is firstly proposed by Chittur et al. (Chittur 2001), which is classical rule-based intrusion detection method. There are two crucial points when using intelligent algorithms to solve the problem of rule updating.

Firstly, the rule provided by intelligent algorithms is based on data analysis for the network intrusion detection problem. Each attribute in the rule is designed to preserve a randomization parameter for each data, and this parameter multiplied by the data would obtain a weight value for the determinacy of whether a certain data is an attack or not. The determinacy formula, C_i, of whether record R is classified as an attack by rule X_i, is described as follows.

$$C_i(X_i) = \sum_{j=1}^{n} (R_j \times X_{i,j}) \tag{16}$$

where $X_{i,j}$ denotes the random parameter for attribute R_j, and n represents the number of attributes. Furthermore, the arbitrary threshold value is established, and any determinacy value which exceeds this threshold value is regarded as a malicious attack.

Secondly, the fitness of an individual is dependent upon how many attacks are correctly detected and how many normal data are viewed as malicious attack. The false positives are expressed as a positive ratio of total normal data while correct detections are expressed as a negative ratio of total attacks. In this experiment, the fitness function F of specific individual X_i is given as follows.

$$F(X_i) = \frac{\beta}{B} - \frac{\alpha}{A} \tag{17}$$

where α is the number of correctly detected attacks, and A stands for the number of total attacks. Besides, β is the number of false positive, and B denotes the total number of normal data. The fitness value is over the closed interval $[-1,1]$ with -1 being the worst fitness and 1 being the best fitness.

5.2 NSL-KDD dataset

The KDD dataset was generated via a simulated U.S. Air Force local-area network set up at Lincoln Labs, which was operated similarly to a standard Air Force network, excepting for planned and recorded attacks. In this paper, an improved KDD dataset (NSK-KDD) is used to test the performance of MFO–HFA. Although NSL-KDD dataset may not be a perfect representative of practical networks because of the lack of public data sets for network-based IDSs, it still can be used as an effective benchmark dataset to compare different network intrusion detection methods. The NSL-KDD dataset does not include redundant records of the classic KDD99 dataset in the train set, so that the classifiers will not have a preference on frequent records. Meanwhile, there are no duplicate records in the test sets, and the number of selected records from each difficulty level group is inversely proportional to the rate of records in the original KDD99 dataset.

The NSL-KDD dataset was split into twin sections, i.e., training dataset and testing set. The training set consisted of 125,973 network connections, and test set compose of 22,544 network connections. Each one of network connection records compose of 41 attributions and a label attribution. In addition, every string type attribute of original records of NSL-KDD are digitized. For example, the numerical value of the 2th attribute will be 0 when this attribute is "TCP" type. If the "protocol_-type" is "ICMP", 1 is treated as the value of the 2th attribute.

5.3 Parameter settings of compared algorithms

The parameters of six intelligent algorithms (i.e., MFO–HFA, MFO (Shehab et al. 2018), AMFO (Zhao et al.

2018), GMFO (Xu et al. 2019a), CMFO (Xu et al. 2019a), LMFO (Xu et al. 2019a) and OMFO (Apinantanakon and Sunat 2017)) are set as follows. The size of moth population is 100, and the dimension D of individuals is equal to 41. In addition, other parameters of comparison algorithms are the same with their original paper, and the threshold value is set to 1. The initial upper boundary of all dimensions of individuals is 1, and the initial lower boundary is -1. It is worth noted that there is no boundary in decision space, and the initial boundary is only used to initialize the population. Each algorithm is independently run 25 times with 410,000 function evaluations.

5.4 The results of experiment simulation

The standard metrics of precision, recall, F1-Score and accuracy are used to evaluate the performance of MFO–HFA algorithms on NSL-KDD dataset. These metrics can be realized in terms of TP (True Positive), FP (False Positive), TN (True Negative) and FN (False Negative). TP denotes the number of data is classified as an attack, and which are actual attack. FP implies the number of data that are detected as an attack, but which are actual normal data. TN is the number of data that are classified as normal data, and which are actual normal. FN indicates the number of data that are classified as normal data, but which are actual attack. Precision is defined as the proportion of positive identifications that are actual correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{18}$$

Recall denotes the TP rate, i.e., the proportion of correct predictions to that of actual attack.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{19}$$

F1-Score represents the harmonic mean of precision and recall.

$$\text{F1} - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

Accuracy is defined as ratio of correct prediction to that of total amount of data.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{21}$$

The evaluation of all algorithms on NSL-KDD dataset is listed in Table 5. It can be clearly seen from the Table 5 that MFO–HFA gains 96.47% accuracy and obtains the highest score on Precision, Recall and F1-Score, compared with other six algorithms, which indicates that MFO–HFA exhibits promising performance on the rule-updating of network intrusion detection.

In the Fig. 9, the vertical axis is the mean value of fitness over independent 10 runs, and the horizontal axis is the sampling point where 42 sampling points were taken from FES = 1000 and mod (FES, 10,000) = 0). It can be seen from Fig. 8 that MFO–HFA obtains the highest convergence speed and the best solution accuracy on first sampling point. Based on above analysis, it can be concluded that MFO–HFA is effective for rule-based network intrusion detection and has great potential in solving practical engineering problems.

6 Conclusions

This paper proposes a variant of MFO algorithm, which is named MFO with historical flame archive, MFO–HFA for short. An adaptive historical flame archive strategy and a top flame randomly matching mechanism are integrated into MFO. In MFO–HFA, a new flame matrix (historical flame archive) is applied instead of the old flame matrix in MFO. The adaptive historical flame archive strategy adaptively updates flame so that population can keep the historical optimal solution information. In addition, the top flame randomly matching mechanism is utilized to accelerate the convergence speed and make full use of flame information. The performance of the MFO–HFA is compared with other variant of MFO, the original MFO and other state-of-the-art swarm intelligence algorithms on CEC 2005 benchmark functions and intrusion detection problem. Although no algorithm is optimal for every problem according to the theory of no free lunch, MFO–HFA has really obtained promising performance in solving complex shifted and rotated multi-modal problems, and real-world intrusion detection optimization problems.

Funding This research was partially funded by the Shaanxi Natural Science Basic Research Project (Grant No. 2020JM-565).

Data availability Data sharing not applicable.

Declarations

Conflict of interest The authors have received research grants from Xi'an Technological University, and declares that they have no conflict of interest.

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

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Abd El Aziz M, Ewees AA, Hassanien AE (2017) Whale optimization algorithm and moth flame optimization for multilevel thresholding image segmentation. Expert Syst Appl 83:242–256

Ahmed A, Ali M, Selim M (2019) Bio-inspired based techniques for thermogram breast cancer classification. Int J Intell Eng Syst 12:114–124

Anbarasan P, Jayabarathi T (2017) Optimal reactive power dispatch using moth flame optimization algorithm. Int J Appl Eng Res 12:3690–3701

Anfal M, Abdelhafid H (2017) Optimal placement of PMUs in algerian network using a hybrid particle swarm–moth flame optimizer (PSO-MFO), Electroteh Electron Autom 65

Apinantanakon W, Sunat K (2017) OMFO: a new opposition-based moth flame optimization algorithm for solving unconstrained optimization problems. In: International conference on computing and information technology, pp 22–31

Bharany S, Sharma S, Bhatia S et al (2022) Energy efficient clustering protocol for FANETS using moth flame optimization. Sustainability 14(10):6159

Bhesdadiya RH, Trivedi IN, Jangir P et al. (2017) A novel hybrid approach particle swarm optimizer with moth flame optimizer algorithm. In: Advances in computer and computational sciences, pp 569–577

Chittur A (2001) Model generation for an intrusion detection system using genetic algorithms, High School Honors Thesis, Ossining High School, In Cooperation with Columbia University

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

Daylamani-Zad D, Graham LB, Paraskevopoulos IT (2017) Swarm intelligence for autonomous cooperative agents in battles for real-time strategy games. In: 2017 9th international conference on virtual worlds and games for serious applications, pp. 39–46

Dongoran A, Rahmadani S, Zarlis M (2018) Feature weighting using particle swarm optimization for learning vector quantization classifier. J Phys Conf Ser 978:012032

Dorigo M, Maniezzo V, Colorni A (1991) Positive feedback as a search strategy

Dukic ML, Dobrosavljevic ZS (1990) A method of a spread-spectrum radar polyphase code design. IEEE J Sel Areas Commun 8(5):743–749

Elsakaan AA, El-Sehiemy RAA, Kaddah SS et al (2018) Economic power dispatch with emission constraint and valve point loading effect using moth flame optimization algorithm. Adv Eng Forum 28:139–149

Elsheikh AH, Panchal H, Ahmadein M et al (2021) Productivity forecasting of solar distiller integrated with evacuated tubes and external condenser using artificial intelligence model and moth-flame optimizer. Case Stud Therm Eng 28:101671

Emary E, Zawbaa HM (2016) Impact of chaos functions on modern swarm optimizers. PLoS ONE 11:e0158738

Farrag AAS, Mohamad SA, Sayed ME (2019) Swarm intelligent algorithms for solving load balancing in cloud computing. Egypt Comput Sci J 43:45–57

Guvenc U, Duman S, Hınıslıoglu Y (2017) Chaotic moth swarm algorithm. In: 2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications, pp 90–95

Jangir P (2017) Optimal power flow using a hybrid particle swarm optimizer with moth flame optimizer. Glob J Res Eng

Jia H, Ma J, Song W (2019) Multilevel thresholding segmentation for color image using modified moth flame optimization. IEEE Access 7:44097–44134

Kanata S, Sianipar GH, Maulidevi NU (2018) Optimization of reactive power and voltage control in power system using hybrid artificial neural network and particle swarm optimization. In: 2018 2nd international conference on applied electromagnetic technology, pp 67–72

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department

Kaur K, Kumar Y (2020) Swarm intelligence and its applications towards various computing: a systematic review. In: 2020 International conference on intelligent engineering and management, pp 57–62

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4, pp 1942–1948

Khairuzzaman AKM, Chaudhury S (2017) Moth flame optimization algorithm based multilevel thresholding for image segmentation. Int J Appl Metaheuristic Comput 8:58–83

Kumar N (2019) A modified particle swarm optimization for task scheduling in cloud computing. In: Proceedings of 2nd international conference on advanced computing and software engineering

Kumar N (2021) Effect of acceleration coefficient on particle swarm optimization for task scheduling in cloud computing. In: EAI endorsed transactions on cloud systems

Li Y, Zhu X, Liu J (2020) An improved moth flame optimization algorithm for engineering problems. Symmetry 12:1234

Liang JJ, Qin AK, Suganthan PN et al (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10:281–295

Ma L, Wang C, Xie N et al (2021) Moth-flame optimization algorithm based on diversity and mutation strategy. Appl Intell 51:5836–5872

Meena G, Choudhary RR (2017) A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In: 2017 International conference on computer, communications and electronics, pp 553–558

Mehta D, Saxena S (2020) Swarm intelligence based hierarchical routing protocols study in WSNs. In: 2020 Sixth international conference on parallel, distributed and grid computing, pp 272–277

Mei RNS, Sulaiman MH, Daniyal H et al (2018) Application of moth flame optimizer and ant lion optimizer to solve optimal reactive power dispatch problems. J Telecommun Electron Comput Eng 10:105–110

Mirjalili S (2015) Moth flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249

Nadimi-Shahraki MH, Banaie-Dezfouli M, Zamani H et al (2021a) B-MFO: a binary moth-flame optimization for feature selection from medical datasets. Computers 10(11):136

Nadimi-Shahraki MH, Fatahi A, Zamani H et al (2021b) Migration-based moth-flame optimization algorithm. Processes 9(12):2276

Nadimi-Shahraki MH, Taghian S, Mirjalili S et al (2021c) Mtv-mfo: multi-trial vector-based moth-flame optimization algorithm. Symmetry 13(12):2388

Nadimi-Shahraki MH, Fatahi A, Zamani H et al (2022) Hybridizing of whale and moth-flame optimization algorithms to solve diverse scales of optimal power flow problem. Electronics 11(5):831

NSL-Kdd dataset. [Online], Available: https://www.unb.ca/cic/datasets/nsl.html

Reddy MPK, Babu MR (2019) A hybrid cluster head selection model for internet of things. Clust Comput 22:13095–13107

Revay L, Zelinka I (2019) Swarm intelligence in virtual environment. J Adv Eng Comput 3:415–424

Said S, Mostafa A, Houssein EH et al. (2017) Moth flame optimization based segmentation for MRI liver images. In: International conference on advanced intelligent systems and informatics, pp 320–330

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, pp 52–60

Sayed GI, Hassanien AE (2017) Moth flame swarm optimization with neutrosophic sets for automatic mitosis detection in breast cancer histology images. Appl Intell 47:397–408

Sayed GI, Hassanien AE (2018) A hybrid SA-MFO algorithm for function optimization and engineering design problems. Complex Intell Syst 4:195–212

Shan W, Qiao Z, Heidari AA et al (2021) Double adaptive weights for stabilization of moth flame optimizer: balance analysis, engineering cases, and medical diagnosis. Knowl-Based Syst 214:106728

Shehab M, Abualigah L (2022) Opposition-based learning multi-verse optimizer with disruption operator for optimization problems. Soft Comput 26(21):11669–11693

Shehab M, Khader AT, Laouchedi M (2018) A hybrid method based on cuckoo search algorithm for global optimization problems. J Inf Commun Technol 17(3):469–491

Shehab M, Alshawabkah H, Abualigah L et al (2021) Enhanced a hybrid moth-flame optimization algorithm using new selection schemes. Eng Comput 37:2931–2956

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

Storn R, Price K (1997) Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. J Glob Optim 11:341–359

Suganthan PN, Hansen N, Liang JJ et al. (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGAL report, 2005005

Trivedi IN, Parmar SA, Pandya MH et al. (2018) Optimal active and reactive power dispatch problem solution using moth flame optimizer

Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15:55–66

Wang M, Chen H, Yang B et al (2017) Toward an optimal kernel extreme learning machine using a chaotic moth flame optimization strategy with applications in medical diagnoses. Neurocomputing 267:69–84

Wang Z, Cao Z, Liu C et al (2022) An enhanced moth-flame optimization with multiple flame guidance mechanism for parameter extraction of photovoltaic models. Math Probl Eng 2022:8398768

Xu L, Li Y, Li K et al (2018) Enhanced moth flame optimization based on cultural learning and Gaussian mutation. J Bionic Eng 15:751–763

Xu Y, Chen H, Luo J et al (2019a) Enhanced moth flame optimizer with mutation strategy for global optimization. Inf Sci 492:181–203

Xu YT, Chen HL, Heidari AA et al (2019b) An efficient chaotic mutative mode-flame-inspired optimizer for global optimization tasks. Expert Syst Appl 129:135–155

Yang X, Luo Q, Zhang J et al. (2017) Moth swarm algorithm for clustering analysis. In: International conference on intelligent computing, pp 503–514

Zhang L, Mistry K, Neoh SC et al (2016) Intelligent facial emotion recognition using moth firefly optimization. Knowl-Based Syst 111:248–267

Zhang X, Wang Z, Ye YF (2018) Optimization of adaptive cycle engine performance based on improved particle swarm optimization. In: 2018 Joint propulsion conference

Zhao XD, Fang YM, Ma Z et al. (2018) An ameliorated moth flame optimization algorithm. In: 2018 37th Chinese control conference, pp 2372–2377