Journal of Bionic Engineering (2022) 19:1522–1543
https://doi.org/10.1007/s42235-022-00207-y

RESEARCH ARTICLE

A Hybrid Moth Flame Optimization Algorithm for Global Optimization

Saroj Kumar Sahoo[1] · Apu Kumar Saha[1]

Received: 20 December 2021 / Revised: 14 April 2022 / Accepted: 19 April 2022 / Published online: 1 July 2022

Abstract
The Moth Flame Optimization (MFO) algorithm shows decent performance results compared to other meta-heuristic algorithms for tackling non-linear constrained global optimization problems. However, it still suffers from obtaining quality solution and slow convergence speed. On the other hand, the Butterfly Optimization Algorithm (BOA) is a comparatively new algorithm which is gaining its popularity due to its simplicity, but it also suffers from poor exploitation ability. In this study, a novel hybrid algorithm, h-MFOBOA, is introduced, which integrates BOA with the MFO algorithm to overcome the shortcomings of both the algorithms and at the same time inherit their advantages. For performance evaluation, the proposed h-MFOBOA algorithm is applied on 23 classical benchmark functions with varied complexity. The tested results of the proposed algorithm are compared with some well-known traditional meta-heuristic algorithms as well as MFO variants. Friedman rank test and Wilcoxon signed rank test are employed to measure the performance of the newly introduced algorithm statistically. The computational complexity has been measured. Moreover, the proposed algorithm has been applied to solve one constrained and one unconstrained real-life problems to examine its problem-solving capability of both type of problems. The comparison results of benchmark functions, statistical analysis, real-world problems confirm that the proposed h-MFOBOA algorithm provides superior results compared to the other conventional optimization algorithms.

Keywords Moth flame optimization algorithm · Butterfly optimization algorithm · Bio-inspired · Benchmark functions · Friedman rank test

1 Introduction

Optimization has the main function in both industrial purposes and the scientific research world. Many numerical and computational processes have been invented to clear up optimization issues in the last twenty years. However, with the aid of numerical methods, it is very complicated to resolve the problems which are non-convex, highly nonlinear, include a giant quantity of variables and constraints. To overcome the drawbacks, such as extra mathematical calculations, initial guess, convergent problems in discrete optimization problems, a set of optimization algorithms known as meta-heuristics algorithms have been proposed in the latest decades. Broadly we divide metaheuristic algorithms into two groups viz., Single Solution-Based (SSB) methods and Population-Based (PB) methods. The SSB methods perform the search by single search representatives, and a group of search representatives is used in PB methods. Depending on single and social information, each solution's position is renovated in PB methods. Moreover, various solutions could easily search the whole search space; hence, better results are produced in PB methods compared to the SSB methods. The PB optimization techniques are mainly grouped into four different types: (i) evolutionary algorithms such as, Genetic Algorithm (GA) [1], Differential Evolution (DE) [2], Biogeography-Based Optimization (BBO) [3], Bird Mating Optimizer (BMO) [4], etc. (ii) Swarm Intelligence (SI) based algorithms, namely Particle Swarm Optimization (PSO) [5], Salp Swarm Algorithm (SSA) [6], Whale Optimization Algorithm (WOA) [7], Symbiotic Organism Search (SOS) [8], Butterfly Optimization Algorithm (BOA) [9], Monarch Butterfly Optimization (MBO) [10], Moth Flame Optimization (MFO) [11], Backtracking Search Algorithm (BSA) [12], JAYA algorithm [13], Slime Mould Algorithm (SMA) [14], Moth Search Algorithm (MSA) [15], Harris Hawks Optimization (HHO) [16], Hunger Games Search (HGS) [17], Colony Predation Algorithm (CPA) [18] etc.

✉ Apu Kumar Saha
apusaha.nita@gmail.com

[1] Department of Mathematics, National Institute of Technology, Agartala, Tripura 799046, India

(iii) physical or chemical law-based algorithms, namely Multi-Verse Optimizer Algorithm (MVO) [19], Gravitational Search Algorithm (GSA) [20] algorithm, Chemical Reaction Optimization (CRO) [21], Atom Search Optimization (ASO) [22], etc. and (iv) human-based algorithms, such as Teaching–Learning Based Optimization (TLBO) [23] algorithm, Cognitive Behavior Optimization Algorithm (COA) [24]. Apart from these above algorithms, several algorithms have been proposed using the mathematics concepts like algebra, geometry etc. Few of them are Sine Cosine Algorithm (SCA), [25], Runge kutta Method (RUN) [26], weIghted meaN oF vectOrs (INFO) [27] etc. Usually, these algorithms start with a randomly taken set of the initial solutions and then run the process until the global optimal solutions of the objective functions are obtained. The optimization process will be stopped when it reaches a maximum number of iterations set by researchers. There is increased awareness and interest nowadays for implementing such metaheuristic algorithms, which are both inexpensive and efficient.

MFO is a SI based algorithm first discovered in 2015 by Mirjalili [11]. MFO's inspiration came from the moths' navigation technique in nature, referred to as transverse orientation. In particular, MFO has two critical strategies, such as spiral flight search and Simple Flame Generation (SFG). The SFG method can create flames from a group of the most powerful moth individuals and fire acquired so far. Moths are given the ability to spiral into the fire to update their place in the iterative process by mimicking the transverse orientation of other moths. Ultimately, MFO can select the most appropriate answer within the search space. If MFO is to succeed, transverse moth orientation is necessary.

MFO has a strong ability to solve numerous challenging constrained and unknown search space problems, which is the main advantage of MFO among all other traditional algorithms. Due to the less parameter and easy algorithm, MFO also has been applied to handle several real-life scientific problems such as optical network unit placement [28], automatic generation control problem [29], image segmentation [30], feature selection [31], medical diagnoses [32, 33], smart grid system [34], and so on.

While MFO may represent a new type of population-based optimization method, the MFO algorithm still needs to be further developed and studied, including the speed of convergence and the capacity to search globally [35]. Various researchers have already proposed some improvements to MFO to overcome the disadvantages of the MFO algorithm. For example, Hongwei et al. [36] proposed a new variant of the MFO algorithm named chaos-enhanced MFO by integrating chaos map into MFO to overcome the demerits of the MFO algorithm. Yueting et al. [37] proposed a series of new variants of the MFO algorithm by integrating MFO with Cauchy mutation, Gaussian mutation, levy mutation, or the combination of three mutations to reduce the disadvantages of MFO algorithm where three modified strategies boost the diversification and intensification capability of the basic MFO algorithm. Xu et al. [38] introduced a new variant of the MFO algorithm by embedding chaotic local search and Gaussian mutation named CLSGMFO to get a more stable balance between diversification and intensification. Kaur et al. [39] presented a modified version of the MFO algorithm, dubbed E-MFO, in which a division of iterations, a Cauchy distribution function, and the influence of the better flame was added to the MFO algorithm to maintain a favorable trade-off between diversification and intensification, as well as increased exploration and exploitation. Tumar et al. [40] embedded a modified MFO algorithm. They proposed an Enhanced Binary MFO algorithm (EBMFO) to predict software faults using adaptive synthetic sampling (ADASYN). Wei Gu and Gan Xiang [41] proposed a new modified MFO algorithm named multi-operator MFO algorithm (MOMFO), which integrates three operators called adaptive control strategy, elite search strategy, and chaos search strategy to make a balance between global and local search capability. The MFO algorithm was updated by Ma et al. [42] to address some of the shortcomings of the basic MFO algorithm, such as slow convergence and convergence to a local minimum. Both the exploration–exploitation and optimization performance optimization methods contain the inertia weight of the diversity feedback control and the small probability mutation component, which are embedded.

In recent times, meta-heuristics and hybrid metaheuristics have played a major role in the research field. Hybridization is used to solve hard optimization problems due to the combination of two to three individual meta-heuristics algorithms. It is also helpful for improving the metaheuristics algorithm with some additional techniques for better improvement of results, run time, or both. Some of the hybrid methods of MFO have been developed by different authors, such as in [43], the author developed an interesting population-based algorithm using a proportional selection scheme to integrate the MFO and Hill Climbing (HC) algorithm named PMFOHC, which helps in (a) quickening the searching process (b) to improve the solution quality. Wu et al. [44] introduced a new PB algorithm known as the HSDE-MFO algorithm by integrating hybrid symbiotic DE and MFO to acquire suitable PV model parameters. In [35] the authors developed a modified algorithm of MFO by the mixture of the Water Cycle Algorithm (WCA), and MFO noted as WCMFO. Here MFO increases the exploitation, and WCA improves the diversification of WCMFO. Also, it has been used in solving constrained optimization problems. Bhesdadiya et al. [45] proposed an algorithm by integrating PSO and MFO which enhance the diversification search during solving high complex design problem and showed superiority in solving unconstrained optimization problems. In [46–48] various hybrid techniques of MFO algorithm have been established to increase the efficiency of MFO algorithm.

Like the MFO algorithm, the BOA algorithm is a relatively new PB metaheuristic algorithm that mimics the searching of food and mating pair behaviour of butterflies for global optimization. The approach is based mainly on butterflies' foraging strategies that use their smell to determine where the nectar or the pairing partner is. The BOA is a highly powerful and versatile algorithm to solve complicated real-world problems where the search areas are relatively complex. For example, Arora and Singh [49] introduced a novel improved BOA (IBOA) using a dynamic and adaptive strategy to modify the sensor modality instead of a constant value. The authors of [50] embedded a novel enhanced BOA algorithm called Bidirectional BOA (BBOA) by applying bidirectional search in BOA, which assisted the local search in both forward and backward direction. While selecting the direction for local search, the greedy selection technique was used. In [51], an improved BOA (WPBOA) was proposed, which incorporated guiding weights and a population restart strategy. With the addition of guiding weight into the global search phase, the algorithm's convergence rate and precision were increased. Dhanya and Kanmani [52] introduced a novel algorithm (BOA-C) with the help of Cauchy mutation operator to enhance the global search ability of BOA and tested on both low and high-dimensional optimization problems. Li et al. [53] introduced an enhanced version of BOA algorithm, namely FPSBOA to balance the exploration and exploitation of BOA. The authors have used nineteen 2000-dimensional and twenty 1000-dimensional functions to verify FPSBOA for complex large-scale optimization problems. In [54], the evidence of bias of BOA was demonstrated for the problems whose optimal value was near the origin, and an unbiased BOA (UBOA) was suggested to eliminate this problem. Lohar et al. [55] used BOA and some other algorithms to optimize the geotechnical parameters used in slope stability analysis. Again, in [56], Arora and Singh proposed another hybrid method by the ensemble of BOA and artificial bee colony (ABC) algorithm. In 2019, Arora and Anand introduced binary versions of BOA (bBOA) [57] where two approaches of binary BOA, namely bBOA-S and bBOA-V were proposed and applied the same for feature selection problem in wrapper mode. Recently, Sharma et al. [58] presented a novel hybrid MPBOA algorithm, which combines the BOA's parasitism and mutualism phases with the SOS algorithm's search phrases to improve the search behaviour of the BOA, which allows for better trade-offs between global and local searches in the MPBOA algorithm. Sharma et al. [59] created a new hybrid metaheuristic algorithm called h-BOASOS integrating BOA and SOS algorithms, and then applied it to find the cost and weight of the cantilever retaining wall. Sharma and Saha [60] introduced a powerful hybrid algorithm named BOSCA by combining SCA with BOA, which helps in stabilizing the global exploration and local exploitation ability of the proposed algorithm. Sharma and Saha [61] introduced a new efficient hybrid algorithm, m-MBOA. They utilized the mutualism step in the exploration section of BOA to decorate the overall performance of the original BOA algorithm. Liu et al. [62] introduced an upgraded version of the BOA called LBOLBOA by integrating orthogonal learning, Lévy flight, and Broyden-Fletcher-Goldfarb Shanno (BFGS) into the original BOA. The main goal of the proposed LBOLBOA is to reduce the shortcoming of the BOA such as slow convergence speed and quickly fall into the local optima solution. The effectiveness of the suggested LBOLBOA has been tested on IEEE CEC’2017 benchmark problems and the parameter optimization of the Kernel Extreme Learning Machine (KELM) for prediction of cervical hyperextension injury. Yu et al. [63] developed an improved BOA-optimized KELM model (in short SBOA-KELM) by integrating SSA into the original BOA algorithm and applied it to bearing fault diagnosis. First, the energy entropy features are extracted from the raw vibration signals by complete ensemble empirical mode decomposition based on adaptive noise (CEEMDAN). The original vibration signals were decomposed into multiple Intrinsic Mode Function (IMF) components by CEEMDAN. The energy entropy of the IMFs was calculated to construct an energy feature vector. Second, to avoid data redundancy caused by smaller energy features and increase calculation, a random forest was used to evaluate feature’s importance and select informative features as new feature vectors. Third, the proposed SBOA-KELM method was used for fault feature classification. Finally, the proposed SBOA has been tested on IEEE CEC’2017 benchmark functions and SBOA-KELM applied diagnosing the fault diagnosis of rolling bearings.

Apart from the above modifications on MFO and BOA algorithms, various researchers introduced other efficient hybrid algorithms for solving various global optimization problems. For example, Saka et al. [64] introduced hybrid Taguchi-Vortex Search (VS) algorithm (in short HTVS) by combining VS algorithm and Taguchi orthogonal approximation. The aim of the proposed algorithm is to develop a better trade-off between diversification and intensification. Chakraborty et al. [65] introduced a new hybrid method by integrating modified WOA with Success History-based Adaptive DE (SHADE). The main goal of this hybrid method is to reduce the shortcomings of both algorithms and guide both algorithms to explore and exploit in the search space, and helps obtain good quality of solutions. Singh and Singh [66] introduced a hybrid algorithm HPSOGWO with the help of PSO and Grey Wolf Optimizer (GWO) to enhance the exploration and exploitation ability of both the algorithms. Wang et al. [67] introduced hybrid VS by merging Artificial Bee Colony (ABC) algorithm and VS algorithm to enhance the effectiveness of the component algorithms. Nama and Saha [68] introduced an efficient hybrid approach, namely HBSA by combining BSA and SQI. The motto of this hybrid approach is to deal with the unconstrained, nonlinear and non-differentiable optimization problems. Yildiz [69] introduced hybrid Taguchi-Harmony Search (HS) algorithm and the robustness and effectiveness of the suggested approach has been measured by applying it into the engineering design and manufacturing optimization problems. Nama et al. [70] proposed the hybrid SOS (HSOS) by integrating SOS algorithm with Simple Quadratic Interpolation (SQI), which helps in enhancing the robustness of the algorithm. Chakraborty et al. [71] introduced an efficient hybrid method called HSWOA by hybridizing the HGS algorithm into the WOA algorithm and applied it to solve different engineering design problems. Sharma et al. [72] introduced a different type of modification in BOA named mLBOA in which Lagrange interpolation and SQI are used in exploration and exploitation phase respectively to improve the original BOA algorithm.

Motivating by the efficiency of MFO and BOA algorithms and the effectiveness of different hybrid techniques, in this article, we have proposed a hybrid algorithm, namely h-MFOBOA, by an intelligent ensemble of BOA in the MFO algorithm to alleviate the inherent drawbacks of the MFO algorithm. As far our knowledge is concerned, no work on the hybridization of MFO and BOA is present in the literature. The salient features of BOA and MFO are hybridized to create a new approach, where BOA is used to improve the efficacy of the MFO algorithm by updating the flame positions during its operation. The following are the main contributions of the work:

(i) Local and global phases of BOA are applied after the updating positions of flames of MFO to further enhance the performance of MFO.
(ii) The proposed algorithm is evaluated and compared to six popular state-of-the-art algorithms and five variants of the MFO algorithm on a diverse set of twenty-three benchmark functions.
(iii) Friedman rank test and Wilcoxon signed-rank test are used to analyze the performance of the proposed h-MFOBOA algorithm.
(iv) The complexity of the proposed algorithm has been obtained and some of the convergence graphs are plotted to check its convergence competence.
(v) To see its problem-solving capability, the proposed algorithm is applied to solve a constrained and an unconstrained problem and compared with a wide variety of algorithms.

The rest of the present article is designed as follows: A summary of the MFO and BOA algorithm is shown in Sect. 2 and Sect. 3 respectively. The proposed h-MFOBOA algorithm is shown in Sect. 4. Computational complexity of the proposed h-MFOBOA is introduced in Sect. 5. In Sect. 6, experimental setup, simulation results, statistical analyses, and convergence analysis have been presented. The application of real-world problems is shown in Sect. 7. Finally, conclusions with future enhancements are discussed in Sect. 8.

2 Classical MFO Algorithm

This section presents the origin of the MFO algorithm and its working process with the mathematical formulation in Subsect. 2.1 And 2.2, respectively.

2.1 Inspiration

Moths are insects and belong to the class of Arthropoda. The navigation techniques of moths are unique, which attracts researchers to think about it. Moths travel at night with the moonlight's help, and for navigation, moths utilize the transverse orientation mechanism, shown in Fig. 1. They fly using moonlight through crosswise inclination by keeping a fixed tendency towards the moon for a long journey in a straight path. The efficiency of preference depends on the distance of flame, i.e., when the distance between them decreases, the moth moves in a helix path around the flame, connecting the moth to the flame. Using these behaviours of moth and mathematical modelling, the MFO algorithm is developed by Mirjalili in 2015.

2.2 MFO Algorithm

In basic MFO, all moths are expressed as a set of candidate's solutions. The positions of all moths are expressed as a vector of decision variables. Let us consider the following matrix for moths

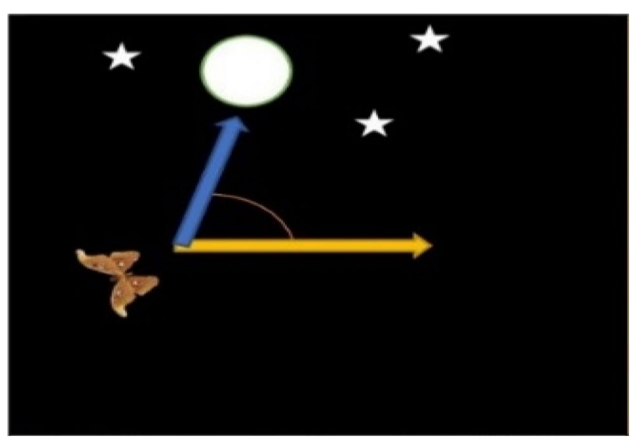

Fig. 1 Transverse orientation of moth

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \ddots & \cdots & \cdots & x_{2,n} \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ x_{N-1,1} & \cdots & \cdots & \ddots & x_{N-1,n} \\ x_{N,1} & x_{N,2} & \cdots & x_{N,n-1} & x_{N,n} \end{bmatrix}, \quad (1)$$

where $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}], i \in \{1, 2, \ldots, N\}$.

N indicates moths' number at initial population and n as variable numbers. The fitness vector of moth is shown below:

$$Fit[\boldsymbol{X}] = \begin{bmatrix} Fit[X_1] \\ Fit[X_2] \\ \vdots \\ Fit[X_n] \end{bmatrix}. \quad (2)$$

Flame matrix is the second key point of the MFO algorithm. Here the size of both moth matrix ***(X)*** and flame matrix ***(FM)*** are same as each moth flies around the corresponding flame.

$$\boldsymbol{FM} = \begin{bmatrix} FM_1 \\ FM_2 \\ \vdots \\ FM_N \end{bmatrix} = \begin{bmatrix} Fm_{1,1} & Fm_{1,2} & \cdots & Fm_{1,n-1} & Fm_{1,n} \\ Fm_{2,1} & \ddots & \cdots & \cdots & Fm_{2,n} \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ Fm_{N-1,1} & \cdots & \cdots & \ddots & Fm_{N-1,n} \\ Fm_{N,1} & Fm_{N,2} & \cdots & Fm_{N-1} & Fm_{N,n} \end{bmatrix}. \quad (3)$$

Also, the fitness vector of flame matrix is store in the following matrix i.e.

$$Fit[\boldsymbol{FM}] = \begin{bmatrix} Fit[FM_1] \\ Fit[FM_2] \\ \vdots \\ Fit[FM_n] \end{bmatrix}, \quad (4)$$

Here $Fit\ [*]$ is a candidate solution's fitness function. MFO has two important components one is moth and other is flame where, moth moves through the respective flame to achieve suitable outcomes and the best outcomes acquired by the moth is known as flame. As the moth moves in a spiral manner, therefore, the author of MFO has defined a spiral function which is represented in the following equation:

$$x_i^{K+1} = \begin{cases} \delta_i \bullet e^{bt} \bullet cos(2\pi t) + Fm_i(k), i \leq N.FM \\ \delta_i \bullet e^{bt} \bullet cos(2\pi t) + Fm_{N.FM}(k), i \geq N.FM \end{cases}, \quad (5)$$

where $\delta_i = \left| x_i^K - Fm_i \right|$ represents distance of moth at i^{th} place and its specific flame (Fm_i) The distance between the i^{th} moth M_i and its specific flame further, b is a constant used to recognize the shape of the search for spiral flight shape and t be any random number between − 1 and 1 referring to how much closer the moth is to its specific flame. Figure 2 represents that a moth flies towards its flame in a helix manner, with a distinct value of t in a 1-dimensional manner.

$$r = -1 + \text{current}_{\text{iter}} \left(\frac{-1}{\text{Maximum}_{\text{iter}}} \right), \quad (6)$$

$$t = (r - 1) \times rand(0, 1) + 1, \quad (7)$$

where $\text{maximum}_{\text{iter}}$ represents the number of maximum iterations, r be the convergence constant decreases from (− 1) to (− 2) linearly proving that both diversification and intensification occur in MFO algorithm.

In every iteration, flame position for the current and last iterations are collected and arranged as per the fitness value for the global and local search. Only the best $N.FM$ flames are preserved, and other flames are wiped away, leading to the one imperfection briefly described in [73]. The following formula can obtain the number of flames $(N.FM)$ that has been reduced over the iteration. The flowchart of the MFO algorithm is presented in Fig. 3.

$$\text{N.FM} = \text{round}\left(\text{N.FM}_{\text{Lastiter}} - \text{Current}_{\text{iter}} \frac{(\text{N.FM}_{\text{Lastiter}} - 1)}{\text{Maximum}_{\text{iter}}} \right). \quad (8)$$

3 Butterfly Optimization Algorithm

A new population-based meta-heuristics approach named the Butterfly Optimization Algorithm (BOA) was created in 2018 by Arora and Singh, based on the food-gathering and mating behaviour of butterflies. In BOA, it is assumed that all butterflies generates an aroma with certain strengths and the aroma of each butterfly has been connected with the location of the search agents. The aroma produced by an individual butterfly is circulated over the entire search region and reach all butterflies and detected by each and every butterfly which form a strong social information network system in the search space. In implementation phase, BOA basically has two phases: global phase and local phase. Butterflies' routine frequency depends on two key concepts: stimulus

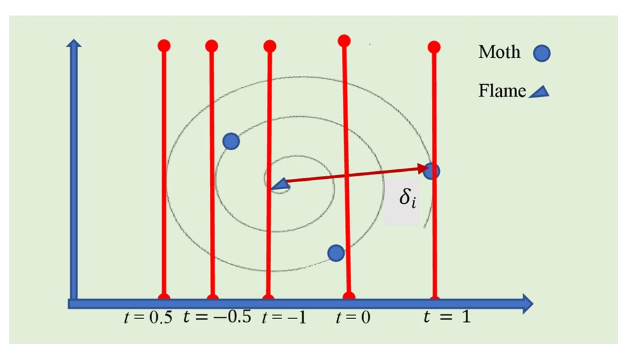

Fig. 2 Logarithmic spiral position w.r.t 't' and space around a flame

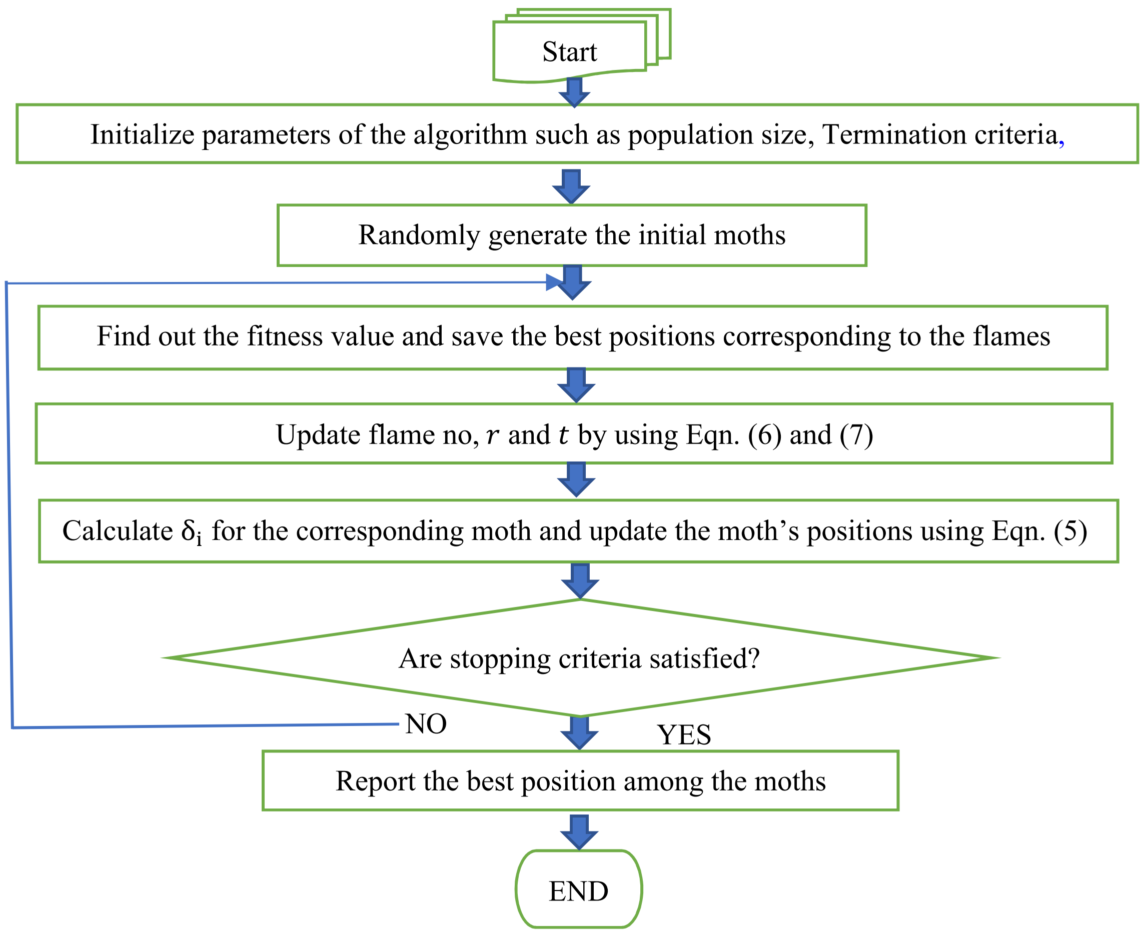

Fig. 3 Flow chart of the MFO algorithm

intensity (I), which is linked to the butterflies' fitness, and scent formulation (f), which is subjective and experienced by other butterflies. BOA depicts the scent as follows:

$$f_i = c \times I^a, \tag{9}$$

where f_i is the amount of aroma originated by i^{th} butterfly, c is the sensory modality, I is the strength of the stimulus and a is called power exponent.

The benefit of f will grow faster than the value of I because the inferior butterfly in BOA move to a better butterfly, as measured by fitness. So, f should be allowed to vary depending on the power exponent and the degree of amalgamation that can be reached (a). In the basic BOA, the values of *"a"* was set to be increase linearly over iterations from 0.1 to 0.3, while $\varepsilon c\varepsilon$ was set at 0.01. The BOA considers a switch probability to carry out its search process, which controls the algorithm's strategy between global and local searches. In basic BOA, it was taken as 0.8. The global as well as local phases of BOA are mathematically represented by the following two equations

$$B\_F_i^{t+1} = B\_F_i^t + \left(r^2 \times B_{F_{best}}^{\ t} - B_{F_i}^{\ t}\right) \times f_i, \tag{10}$$

$$B\_F_i^{t+1} = B\_F_i^t + \left(r^2 \times B_{F_j}^{\ t} - B_{F_k}^{\ t}\right) \times f_i, \tag{11}$$

where $B\_F_{best}^t$ is the location of the best butterfly in the search space at the t^{th} iteration, r is arandom number in (0, 1) and f_i represents the aroma released by the i^{th} butterfly. $B\_F_j^t$ and $B\_F_k^t$ represent the j^{th} and k^{th} butterflies from population in t^{th} iteration. The flowchart of the BOA is presented in Fig. 4.

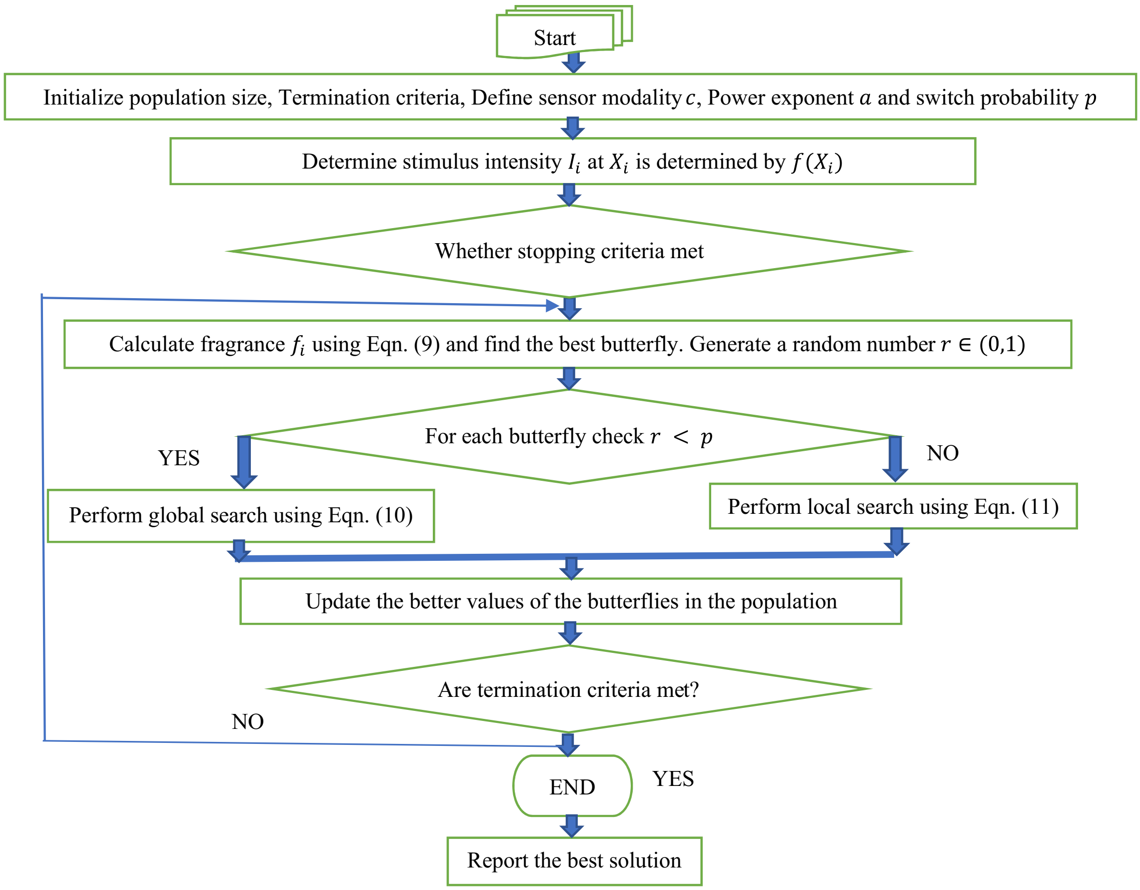

Fig. 4 Flow chart of the BOA algorithm

4 The Proposed Algorithm

The main motto of any metaheuristic algorithm is to handle the balancing phase, i.e., exploration and exploitation. We know that excessive exploration is the reason for losing optimal solutions because it spends more time searching the uninteresting regions. On the other hand, extreme exploitation is also the main reason for premature convergence as the population rapidly lacks diversity. So, better performance of any algorithm is achieved when it maintains stability between diversification and intensification.

In MFO, exploration, and exploitation are obtained from the spiral movement of moths around the flame. The power of the exponent factor 't' gives a better clarification about exploration and exploitation. We know that the next position of the moth is obtained from the spiral Eq. (3). The spiral equation parameter 't' is responsible for how close the moth is to the flame in the next position (with $t = -1$ being the most intimate and $t = 1$ being the farthest). When the next part is out of the space between the moth and the flame, its exploration; when it's in the area, its exploitation.

The MFO features good exploitation ability because individuals in the MFO algorithm follow its flames by a spiral trajectory according to Eq. (5). MFO updates its flames with a 'survival of the fittest' mechanism, which means the flames with better fitness value will survive from the flame selection. This mechanism makes the MFO algorithm features a fast convergence speed but also raises a problem of diversity loss of moths. On the other hand, the literature study of BOA argues that BOA has good exploration ability and poor exploitation ability. It is due to high switch probability value (80%) most of the butterfly performs better in exploration phase than exploitation. Therefore, to avoid conflicts between these two methods and to developed a novel

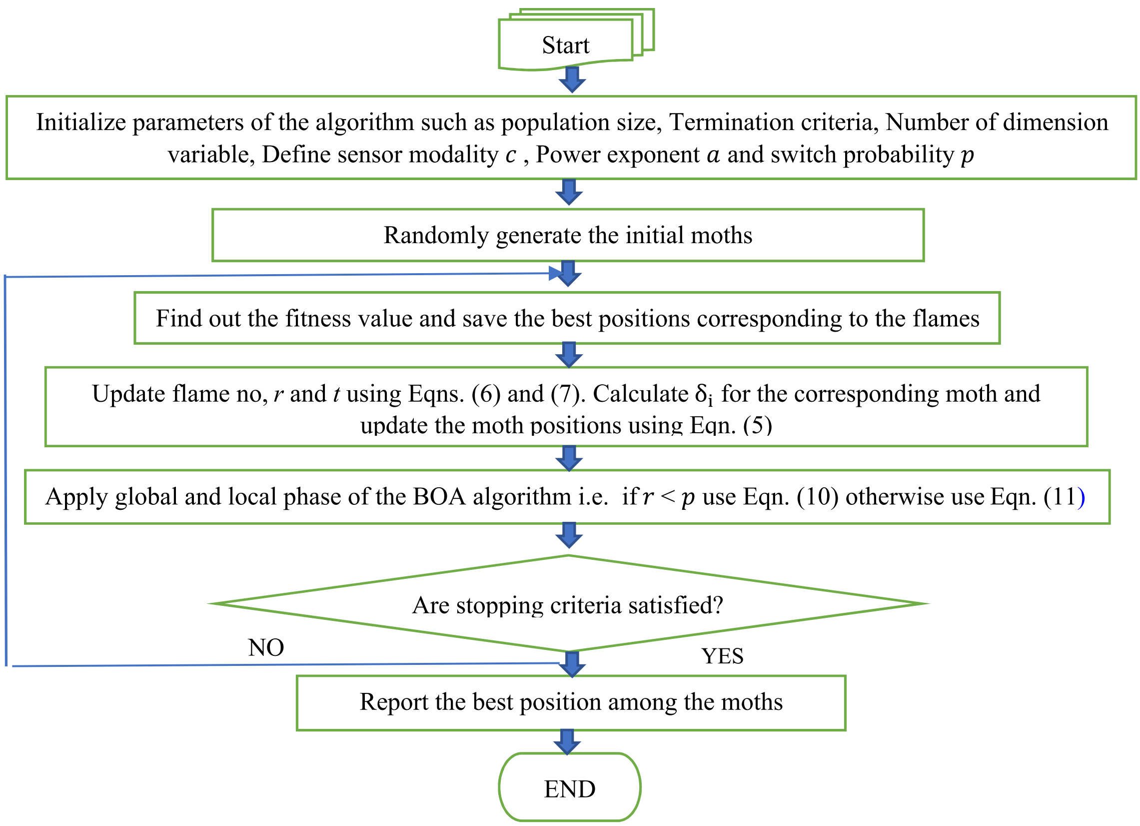

Fig. 5 Flow chart of the h-MFOBOA algorithm

well-balanced metaheuristic algorithm we have embedded BOA into the MFO algorithm.

This paper presents a hybrid moth flame optimization algorithm to increase population diversity and expedite convergence (h-MFOBOA). This strategy also makes it easier to balance the capability of the MFO to discover and exploit new opportunities. We similarly start the algorithm like MFO, and then we apply the global and local phase of the BOA algorithm [Eq. (10) and Eq. (11)] for position updating. The flowchart of the suggested h-MFOBOA is presented in Fig. 5. The major steps of h-MFOBOA can be shown in Algorithm 1 and summarized below.

1st step: initialize all parameters such as the number of populations, maximum iteration, and function evaluation randomly.
2nd Step: apply the sorting procedure to both the moth matrix and flame matrix w.r.t the fitness value and update the number of flames using the Eq. 8.
3rd step: update r and t using Eq. 6 and Eq. 7. Also, Update moths position w.r.t corresponding flame using Eq. 5.
4th Step: update the new solution using Eq. 10 and Eq. 11 and then find the fitness value of the latest solutions. Best fitness gives the optimum value.
5th Step: if it does not satisfy the stopping criteria, go to the 2nd step to get the best fitness value.

Algorithm 1: Pseudo-code of the h-MFOBOA algorithm.

```
Input: Objective function f(X), X = (X_1 X_2 ......... X_N);
  Maximum iteration (Maximum_iter), Number of moths in the population (N), dimension (d), Sensor
  modality (c), power exponent (a) and switch probability (p).
  for i = 1: N                          %% N= no. of population
     for j = 1: d                       %% d = dimension
          Generate N solutions X_i(i = 1,2,....,N) with dimension d using following equation
           X (i,j) = LB(i) + (UB (i) −LB(i)) ∗ rand
      end for
  end for
  While Current_iter < Maximum_iter
          if   Iteration==1
                N.FM = N
              else
                 Employ Eqn. (8).
        end if
       FM = Fitness function f(X);
    if Iteration==1
              Arrange the moths according to FM
              Update the Flames
              Iteration=0;
          else
               arranged moth based on FM from Last iteration
                Update the Flames
     end if
            Reduce convergence constant
          for j  = 1: N
              for k = 1: d
                      Find r and t using Eqn. (6) & Eqn. (7)
                      Update moths position as to their particular flame
              end for
          end for
      select one solution randomly (i ≠ j);
      Update the solution by global and local phase BOA by Eqn. (10) and Eqn. (11);
      Find new solutions best value;
     Current_iter = Current_iter + 1;
  end while
  Output: Report the best flame position
```

5 Computational Complexity of h-MFOBOA

Complexity of any algorithm is a function which provides the running time or space with respect to input size. This is of two kinds: one is complexity of space and other is time complexity. The process of finding a formula for total space will be required towards execution of the algorithm is referred as space complexity. Also, process of finding a formula for total time required for successful execution of algorithm is known as time complexity. A big-O notation is used to analysis the computational complexity of the proposed h-MFOBOA algorithm. The Complexity of h-MFOBOA also depends on initialization of moth position (T_{IMP}), evaluation of moth position (T_{EMP}), searching of spiral flight (T_{SSF}), flame generation (T_{FG} and global and local phase of the BOA (T_{BOA}). Let maximum iterate number, variable number and moths' number are denoted by I, D and N respectively. Here we will use time complexity for the comparison of both h-MFOBOA and MFO algorithm.

According to the quicksort algorithm, Computational complexity for sorting N-flame and N-moth are lying between 3Nlog3NI and (3 N)2I towards worst and best case

$$T_{h-MFOBOA} = T_{TMP} + T_{EMP} + T_{SSF} + T_{FG} + T_{BOA},$$

$$= O(ND) + O(NDI) + O(NDI) + O((3N)2I) + O(NDI),$$

$$= O(ND + 3NDI + 9N2I).$$

Hence, time complexity for h-MFOBOA with respect to worst case is O [NI (D + N)]. Also, from [9], the time complexity of MFO for the worst case is O [NI (D + N)]. Therefore, both MFO and h-MFOBOA has same complexity.

6 Simulation Results and Discussions

In this section, the experimental setup of our proposed method is presented in Sect. 6.1, a comparison of h-MFOBOA with basic MFO and other evolutionary algorithms and statistical performance are presented in Sect. 6.2 and Sect. 6.3 respectively.

6.1 Experimental Setup

The algorithm is coded and run on a Windows computer with an Intel i5 processor, 8 GB of RAM, and a MATLAB R2015a compiler. At most 1000 iterations are in use as a basis to stop our proposed algorithm. There are different ways to stop the algorithm such as maximum number of iterations achieved, a fix error tolerance value, Maximum use of CPU time, maximum number of iterations having zero improvement, etc. Each function was repeated for 30 runs and rounded up to two numbers after the decimal to produce less statistical errors and a statistically significant output. We put down the Average (A), Standard Deviation (SD), 'Best' and 'Worst' of h-MFOBOA with other algorithms for collation. To fulfill this this criteria, one particular union of variables used for h-MFOBOA in the copy of both unimodal, multimodal and fixed dimensional benchmark functions which are taken from literature. The powers exponent constant b is equal to 1 and t varies from −1 to 1 and size of the population is thirty (30).

6.2 Discussion on Basic Benchmark Functions

Our proposed h-MFOBOA optimization model is tested against six meta-heuristics (DE, PSO, JAYA, BOA, BSA, and MFO) which have previously demonstrated their superiority in various global optimization problems and can produce satisfying results on different unimodal, multimodal and fixed-dimension problem instances. The parameter setting of all the algorithms employed for comparison is given in Table 1. The results of each algorithm were calculated and presented in Table 1.

In Table 2, F1–F7 has been investigated under unimodal functions. Out of seven parts, h-MFOBOA achieves superior results for F5, F6, and F7 operations and achieves the best global optimum value for other functions. So, we can conclude that h-MFOBOA is good for diversification and reaches more than 90% best optimum value among different traditional optimization algorithms.

In Table 2, F8–F17 has been investigated under multimodal benchmark functions. Our proposed algorithm possesses superior results for F11, F12, F13, and F14 benchmark functions, and for other parts, it achieves the second and third highest optimum value. From Table 2, it can be clear that h-MFOBOA provides more than 85% good global solutions among other state-of-the-art algorithms.

In Table 2, F18–F23 has been investigated under fixed dimensional multimodal benchmark functions. For F18, F19, F20, and F21, h-MFOBOA achieved the best optimal value, and for others, it provides the second and third highest global optimum value. Therefore, we can conclude that our proposed h-MFOBOA achieved the best quality optimum value among other traditional optimization algorithms.

As shown in Table 3, the average performance of h-MFOBOA is greater than, similar to, or worse than the other six

Table 1 Parameter setting of the considered algorithms

Algorithm	Parameter values
DE	No. of population = 30, Maximum iteration = 1000, Scaling Factor (F) = 0.5 = Crossover probability
PSO	No. of population = 30, Maximum iteration = 1000, $w = 0.9$ to $0.4, c_1 = c_2 = 0.2$
JAYA	No. of population = 30, Maximum iteration = 1000, $r = \text{rand}(0, 1)$
BOA	No. of population = 30, Maximum iteration = 1000, Switch probability (p) = 0.8, Sensor modality (c) = 0.01, Power exponent (a) = 0.1 to 0.3
BSA	No. of population = 30, Maximum iteration = 1000, Two parameters 'a' and 'b' are uniformly random numbers between 0 and 1, Mix rate = 1
MFO	No. of population = 30, Maximum iteration = 1000, Convergence constant decreases linearly from (−1) to (−2)

Table 2 Experimental results of h-MFOBOA with other basic algorithms on 23 benchmark functions

Sl. No		DE	PSO	Jaya	BOA	BSA	MFO	h-MFOBOA
F1	A	1.45E+03	2.64E − 06	9.93E − 05	0	3.00 E+05	3.59E − 110	0
	SD	2.91E+02	2.30E − 06	5.25E − 05	0	0	1.97E − 109	0
	Best	1.05E+01	1.72E − 07	3.26E − 06	0	3.00E+05	4.25E − 122	0
	Worst	6.55E+03	8.11E − 06	4.05E − 04	0	3.00E+05	3.79E − 107	0
F2	A	7.19E − 04	3.99E − 06	0	4.73E − 01	1.81E+05	2.66E − 01	0
	SD	1.27E − 03	1.05E − 05	0	4.54E − 01	0	6.29E − 01	0
	Best	3.69E − 12	2.40E − 10	0	9.53E − 05	1.81E+05	5.18E − 04	0
	Worst	6.05E − 02	5.16E − 05	0	1.98	1.81E+05	7.10E − 01	0
F3	A	7.50E − 05	2.91E − 07	0	1.65E − 01	3.00E+04	1.88	0
	SD	6.98E − 05	4.84E − 07	0	2.42E − 01	0	2.08	0
	Best	4.87E − 09	5.98E − 11	0	8.45E − 06	3.00E+04	1.02	0
	Worst	2.15E − 04	1.84E − 06	0	1.66	3.00E+04	1.26	0
F4	A	1.17E − 08	1.22E − 102	1.55E − 218	0	0	6.82E − 11	0
	SD	2.74E − 08	3.88E − 102	0	0	0	3.73E − 11	0
	Best	3.78E − 14	1.17E − 114	3.68E − 233	0	0	1.43E − 30	0
	Worst	5.10E − 06	1.63E − 101	1.04E − 216	0	0	1.35E − 10	0
F5	A	− 3.79E − 03	− 3.79E − 03	− 3.79E − 03	− 3.34 E − 03	8.71	− 3.56E − 03	− 3.79E − 03
	SD	6.23 E − 06	9.46E − 07	2.27E − 09	7.69 E − 04	5.42E − 15	3.41E − 04	4.26E − 17
	Best	− 3.79E − 03	− 3.79E − 03	− 3.79E − 03	− 3.79E − 03	3.21E − 16	− 3.79E − 03	− 3.21E − 01
	Worst	− 3.79E − 03	− 3.78E − 03	− 3.79E − 03	0	1.14E − 12	− 9.04E − 04	− 3.21E − 01
F6	A	2.24E − 02	3.82E − 03	2.14 E − 12	1.08 E − 01	1.61E+03	1.84 E − 01	0
	SD	2.19E − 02	4.62E − 03	4.68 E − 12	1.84 E − 01	9.25 E − 13	2.75 E − 01	0
	Best	3.75E − 07	1.69E − 07	2.87E − 16	9.55E − 08	4.61E+05	5.75E − 04	0
	Worst	1.85E − 01	1.53E − 02	8.47E − 12	7.70E − 01	2.61E+02	1.66E − 01	0
F7	A	4.09E − 21	2.05E − 06	1.50 E − 32	2.67 E − 01	1.01E − 06	6.36 E − 02	0
	SD	7.57E − 21	6.45E − 06	1.11 E − 47	3.24 E − 01	4.02E − 10	1.03 E − 01	0
	Best	7.11E − 35	8.47E − 10	1.49E − 32	7.22E − 11	7.01E − 12	8.21E − 05	0
	Worst	4.69E − 15	3.56E − 05	1.49E − 32	7.15E − 01	1.31E − 05	5.32E − 01	0
F8	A	3.01	2.34	3	1.18 E+01	1.60E+03	1.61E+01	0
	SD	1.62E − 02	2.39	2.03E − 04	9.68	0	2.01E+01	0
	Best	3.01	9.98E − 01	3	1.05E+01	1.60E+03	1.55E+01	0
	Worst	3.01	1.26E+01	3	1.65E+01	1.60E+03	1.88E+01	0
F9	A	1.19E − 05	3.75E − 04	1.70E − 07	0	8.75E+05	3.13 E − 51	0
	SD	1.35E − 05	9.05E − 04	3.08 E − 07	0	0	1.71 E − 50	0
	Best	3.12E − 14	1.37E − 07	4.26E − 09	0	8.75E+05	4.06E − 131	0
	Worst	1.11E − 04	3.96E − 03	4.48E − 07	0	8.75E+05	2.55E − 48	0
F10	A	3.69E − 05	6.05E − 07	5.17E − 30	0	3.00 E+03	4.18E − 29	0
	SD	2.58E − 05	1.02E − 06	9.74E − 30	0	0	2.29E − 28	0
	Best	6.88E − 11	3.45E − 11	3.74E − 31	0	3.00E+03	6.11E − 68	0
	Worst	1.11E − 04	3.91E − 06	2.27E − 29	0	3.00E+03	4.36E − 27	0
F11	A	− 1.13E − 10	2.21E − 03	− 1.13E − 10	− 9.52 E − 11	1.00E+02	− 1.13E − 10	− 1.13E − 10
	SD	9.61E − 14	6.34E − 03	1.66E − 25	2.07E − 11	0	1.33E − 14	1.28E − 25
	Best	− 1.13E − 10	7.89E − 08	− 1.12E − 10	− 1.12E − 10	1.00E+02	− 1.13E − 10	− 1 0.10E − 10
	Worst	− 1.12E − 10	2.93E − 02	− 1.12E − 10	− 2.56E − 11	1.00E+02	1.11E − 10	− 1.08E − 10
F12	A	1.93E − 01	6.39E − 01	2.48E − 02	6.11E − 01	8.62E+02	1.19E − 01	1.86E − 02
	SD	1.88E − 01	0	3.08E − 02	5.94E − 01	1.15E − 13	1.17E − 01	2.06E − 02
	Best	8.65E − 04	6.37E − 01	2.26E − 04	9.75E − 07	8.15E+02	5.89E − 07	7.31E − 06
	Worst	1.50E − 01	6.37E − 01	7.20E − 02	7.68E − 01	4.19E+04	3.47E − 01	3.54E − 01

Table 2 (continued)

Sl. No		DE	PSO	JAYA	BOA	BSA	MFO	h-MFOBOA
F13	A	1.71E − 03	1.56	3.16E − 04	6.13E − 03	4.65E+26	1.16E − 02	− 3.62
	SD	9.34E − 04	1.4	2.19E − 05	8.33E − 03	2.54E+27	1.66E − 02	1.05E − 01
	Best	5.34E − 09	1.21E − 02	3.07E − 04	4.56E − 09	1.21E+26	4.96E − 05	− 3.62
	Worst	2.10E − 02	4.57	4.14E − 04	7.66E − 02	3.59E+28	1.56E − 01	− 3.62
F14	A	6.45E − 05	1.21E − 02	0	1.47 E − 04	− 2.71E − 94	0	− 5.11E − 03
	SD	9.79E − 05	1.67E − 02	0	8.06 E − 04	0	0	0
	Best	5.11E − 10	2.95E − 05	0	4.07E − 05	− 2.70E − 94	0	− 5.11E − 03
	Worst	1.67E − 04	6.02E − 02	0	1.33E − 02	− 2.70E − 94	0	− 5.11E − 03
F15	A	− 3.04	− 2.03E − 01	− 3.02	− 2.53	− 3.77E − 02	− 2.91	− 3.04
	SD	1.10E − 03	3.52E − 02	2.83 E − 02	1.74 E − 01	1.41E − 17	4.73 E − 02	6.88 E − 12
	Best	− 3.04	− 2.69E − 01	− 3.32	− 2.94	− 4.65E − 19	− 3.65	− 3.78
	Worst	− 3.04	− 1.13E − 01	− 3.2	− 1.83	− 1.22E − 01	− 2.41	− 3.78
F16	A	6.45E − 05	1.13E − 03	0	1.47 E − 04	7.50 E+03	0	0
	SD	9.79E − 05	2.69E − 03	0	8.06 E − 04	0	0	0
	Best	4.47E − 14	3.13E − 04	0	9.45E − 11	7.49E+03	0	0
	Worst	3.30E − 04	1.53E − 02	0	7.31E − 03	7.49E+03	0	0
F17	A	9.94E − 15	3.53 E+01	1.24 E − 113	0	5.10 E+04	7.47 E − 98	0
	SD	3.88E − 14	0	4.71 E − 113	0	0	4.09 E − 97	0
	Best	6.60E − 35	3.15E+01	9.58E − 122	0	5.08E+04	6.11E − 155	0
	Worst	2.16E − 12	3.15E+01	9.03E − 113	0	5.08E+04	3.55E − 91	0
F18	A	2.14 E − 02	1.15 E+04	6.13 E − 05	6.32 E − 01	3.97 E+05	7.27 E − 01	− 2.92
	SD	1.94E − 02	0	9.92 E − 05	1.42 E − 01	0	1.75 E − 01	5.30 E − 02
	Best	8.38E − 04	1.12 E+04	6.07E − 05	4.28E − 03	3.67E+04	3.09E − 04	− 2.92E − 01
	Worst	5.90E − 02	1.12 E+04	4.14E − 05	2.71	3.67E+04	1.06E+01	− 2.92E − 01
F19	A	− 3.04	− 1.46	− 3.02	− 2.53	− 1.32	− 2.91	− 1.65 E+03
	SD	1.10E − 03	0	2.83 E − 02	1.74 E − 01	4.51E − 16	4.73 E − 02	1.53 E+02
	Best	− 3.04	− 1.41	− 3.32	− 2.82	5.01E − 18	− 3.12	− 1.87E+03
	Worst	− 3.04	− 1.41	− 3.17	− 1.94	− 1.66	− 2.85	1.64E+01
F20	A	− 1.92E+03	− 3.25	− 1.48E+03	− 1.52E+03	7.50E+03	− 1.37 E+03	3.49 E − 12
	SD	5.07E+01	6.04E − 02	4.98 E+01	7.97 E+01	0	1.11 E+02	9.23 E − 12
	Best	− 2.95E+03	− 3.32	− 1.57E+03	− 1.75E+03	8.12E+03	− 1.42E+03	6.89E − 16
	Worst	− 1.35E+03	− 3.2	− 1.23E+03	− 1.33E+03	6.12E+04	− 1.81E − 01	3.54E − 11
F21	A	1.92E − 02	− 4.98	1.56E − 06	1.82E − 05	2.55E+03	0	− 1.54 E+01
	SD	1.25E − 02	3.27	4.40E − 06	1.59E − 06	1.38E − 12	0	1.24
	Best	5.40E − 04	− 1.01E+01	5.35E − 08	6.76E − 08	1.39E+03	0	− 1.54E+01
	Worst	2.05E − 01	− 2.63	4.81E − 06	3.17E − 04	2.85E+03	0	1.34E+01
F22	A	1.59	1.04 E+01	3.74E − 01	0	3.60E+01	0	0
	SD	1.85E − 01	0	2.04E − 01	0	0	0	0
	Best	1.13E − 02	1.03 E+01	5.19E − 07	0	2.78E+01	0	0
	Worst	3.31	1.03 E+01	2.84E − 01	0	3.65E+01	0	0
F23	A	7.1	2.87 E+01	8.56E − 01	2.32 E − 01	5.66E+01	6.76E − 58	0
	SD	5.14E − 01	0	1.85E − 01	1.27	6.70E − 02	3.70E − 57	0
	Best	5.25	2.85 E+01	6.45E − 04	5.91E − 06	5.66E+01	8.31E − 105	0
	Worst	1.17E+01	2.85 E+01	2.65E − 01	1.67E+01	7.12E+03	2.95E − 51	0

Table 3 Performance assessment of h-MFOBOA and other basic algorithms on 23 benchmark functions

	DE	PSO	JAYA	BOA	BSA	MFO
Superior to	22	22	16	14	22	17
Similar to	0	0	6	6	1	2
Inferior to	1	1	2	3	0	4

Table 4 Friedman rank test of h-MFOBOA and other basic algorithms on 23 benchmark functions

Algorithm	Mean rank	Rank
h-MFOBOA	**2.15**	**1**
BOA	3.39	2
MFO	4.24	4
DE	4.41	6
PSO	4.39	5
JAYA	4.15	3
BSA	5.26	7

Table 5 Wilcoxon's test for h-MFOBOA and other basic algorithms on 23 benchmark functions ($\alpha = 0.05$)

h-MFOBOA vs. Algorithm	*p* value	R +	R −	Winner
BOA	< 0.001	383	52	h-MFOBOA
MFO	0.001	266	14	h-MFOBOA
DE	0.008	207	28	h-MFOBOA
PSO	0.010	404	30	h-MFOBOA
JAYA	< 0.001	399	7	h-MFOBOA
BSA	< 0.001	347	59	h-MFOBOA

algorithms in a range of circumstances. From Table 3, we noticed that h-MFOBOA works better than DE, PSO, JAYA, BOA, BSA, and MFO in 22, 22, 16, 14, 22, and 17 benchmark functions, respectively, similar results can be seen in 0, 0, 6, 6, 1 and 2 occasions, respectively, and worse values are achieved in 1, 1, 2, 3, 0 and 4 benchmark functions respectively. The mathematical formulation of the 23 (twenty-three) benchmark functions with dimension, range of the variables, and optimum value are shown in Appendix-1.

6.3 Statistical Analysis

Friedman and Wilcoxon signed rank test are used to analyze the performance of proposed h-MFOBOA algorithm. In this paper, for each benchmark function Friedman test is used from the average performance of algorithms. we use IBM-SPSS software for finding the average rank. The outcomes of the Friedman rank test between h-MFOBOA, DE, PSO, JAYA, BOA, BSA and MFO for twenty-three benchmark functions is presented in Table 4. From Tables 4, it is clearly visible that h-MFOBOA obtain least rank among other algorithms at 1% relevant.

The outcomes of Wilcoxon rank test are demonstrated at the 5% relevant point between h-MFOBOA, DE, PSO, JAYA, BOA, BSA and MFO for twenty-three benchmark functions is presented in Table 5. From Table 5, all the R + (positive rank) values higher than R − (negative) values which demonstrate the superiority of h-MFOBOA among other competitors.

For contrast, some of the convergence graphs of the h-MFOBOA method with other techniques, including DE, PSO, JAYA, BOA, BSA, and MFO, were compared on certain benchmark functions such as Beale, Levy, Matyas, and Power-Sum in Fig. 3. In these figures, both the function evaluation and objective function value are presented in the horizontal and vertical axis, respectively. It can be clear that h-MFOBOA has rapid convergence as compared to the other methods. About search accuracy, robustness, convergence speed, and escaping local optima, h-MFOBOA has greater performance and competitive advantage over different algorithms.

6.4 Discussion on Variants of the MFO Algorithm

In this subsection, comparison evaluation has been done in with six MFO variants such as OMFO [74], LMFO [75], WCMFO [35], WEMFO [76], and SMFO [77]. The simulation outcomes of h-MFOBOA together with five MFO variants for twenty-three benchmark functions including unimodal and multimodal and fixed dimensional multimodal benchmark functions are presented in Table 6. These benchmark functions are taken from Appendix-1. The parameters of all the variants are taken same as in their original algorithm. All the results are evaluated using Matlab 2015(a). The Average (A), Standard Deviation (SD), 'Best' and 'Worst' values of h-MFOBOA with other variants of the MFO algorithm are presented in Table 6.

From Table 6, it can be observed that, our proposed h-MFOBOA algorithm achieved more than 82% best results for all groups of benchmark problems as compared to the variants of MFO algorithms but it provides more than seventy percent best results when compared with WCMFO and WEMFO algorithm. Also, the number of occasions of superiority, similarity and inferiority are presented in Table 7. From Table 7, we noticed that h-MFOBOA works better than OMFO, LMFO, WCMFO, WEMFO and SMFO in 17,

Table 6 Experimental results of h-MFOBOA with MFO variants on 23 benchmark functions

Sl. No		OMFO	LMFO	WCMFO	WEMFO	SMFO	h-MFOBOA
F1	A	1.81E − 01	5.22E − 03	2.54E − 01	1.65E − 01	1.32E − 02	0
	SD	2.68E − 01	4.85E − 03	3.65E − 01	1.80E − 01	1.79E − 02	0
	Best	4.56E − 03	1.04E − 04	1.41E − 13	1.42E − 03	1.47E − 03	0
	Worst	9.55E − 01	1.63E − 02	7.62E − 01	6.78E − 01	5.78E − 01	0
F2	A	1.25	2.20E − 02	7.70E − 08	3.03E − 01	1.16E − 01	0
	SD	1.77	2.48E − 02	5.50E − 08	3.47E − 01	1.68E − 01	0
	Best	5.95E − 03	3.06E − 04	9.67E − 09	3.92E − 03	4.92E − 03	0
	Worst	6.99	9.95E − 02	2.33E − 07	1.25	1.25	0
F3	A	1.10E − 93	3.29E − 08	2.70E − 22	7.04E − 198	4.23E − 14	0
	SD	6.05E − 93	3.09E − 08	3.27E − 22	0	1.94E − 13	0
	Best	1.57E − 209	7.62E − 10	1.52E − 24	3.91E − 264	1.91E − 26	0
	Worst	3.31E − 92	1.27E − 07	1.16E − 21	2.11E − 196	2.51E − 10	0
F4	A	1.11E − 125	7.77E − 10	0	2.96E − 204	2.60E − 12	0
	SD	6.11E − 125	1.52E − 09	0	0	1.43E − 11	0
	Best	1.71E − 258	9.45E − 14	0	4.49E − 250	3.29E − 18	0
	Worst	3.34E − 124	7.89E − 09	0	8.54E − 203	5.54E − 09	0
F5	A	− 3.35E − 03	− 3.79E − 03	− 3.79E − 03	− 3.78E − 03	− 3.78E − 03	− 3.79E − 03
	SD	6.97E − 04	3.39E − 07	2.75E − 10	1.33E − 05	1.43E − 05	4.26E − 17
	Best	− 3.79E − 03	− 3.79E − 03	− 3.79E − 03	− 3.79E − 03	− 3.79E − 03	− 3.21E − 01
	Worst	− 1.28E − 03	− 3.78E − 03	− 3.78E − 03	− 3.73E − 03	− 3.73E − 03	− 3.21E − 01
F6	A	1.51E − 01	7.34E − 03	8.43E − 10	1.41E − 01	2.78E − 02	0
	SD	2.51E − 01	9.62E − 03	1.13E − 09	1.31E − 01	3.01E − 02	0
	Best	1.27E − 04	2.96E − 05	2.19E − 11	3.85E − 03	5.85E − 03	0
	Worst	9.99E − 01	3.59E − 02	4.47E − 09	5.19E − 01	4.19E − 01	0
Sl. No		OMFO	LMFO	WCMFO	WEMFO	SMFO	h-MFOBOA
F7	A	6.95E − 110	3.09E − 07	6.74E − 20	1.61E − 206	2.80E − 11	0
	SD	3.81E − 109	2.38E − 07	1.09E − 19	0	1.21E − 10	0
	Best	1.18E − 246	1.52E − 08	7.27E − 22	1.76E − 248	3.76E − 16	0
	Worst	2.08E − 108	8.33E − 07	5.87E − 19	4.71E − 205	4.51E − 09	0
F8	A	0	4.8513E − 06	0	0	2.89E − 09	0
	SD	0	5.31E − 06	0	0	1.43E − 08	0
	Best	0	2.06E − 08	0	0	2.06E − 08	0
	Worst	0	2.07E − 05	0	0	2.07E − 05	0
F9	A	0	1.59E − 06	6.75E − 15	0	5.00E − 13	0
	SD	0	1.96E − 06	3.41E − 15	0	1.59E − 12	0
	Best	0	4.95E − 08	1.88E − 15	0	1.45E − 25	0
	Worst	0	9.56E − 06	1.66E − 14	0	5.16E − 07	0
F10	A	7.19E − 02	6.59E − 04	9.04E − 12	8.77E − 03	3.44E − 03	0
	SD	1.10E − 01	9.23E − 04	1.35E − 11	1.15E − 02	5.47E − 03	0
	Best	4.48E − 04	1.54E − 05	5.69E − 14	1.45E − 05	8.45E − 05	0
	Worst	4.13E − 01	3.61E − 03	5.30E − 11	5.16E − 02	3.16E − 02	0
F11	A	− 1.12E − 10	− 1.12E − 10	− 1.12E − 10	− 1.12E − 10	− 1.13E − 10	− 1.13E − 10 1.28E − 25
	SD	6.89E − 15	1.82E − 14	1.05E − 19	4.08E − 14	5.92E − 13	− 1.10E − 10
	Best	− 1.12E − 10	− 1.12E − 10	− 1.12E − 10	− 1.12E − 10	− 1 0.12E − 10	− 1.08E − 10
	Worst	− 1.11E − 10	− 1.11E − 10	− 1.10E − 10	− 1.11E − 10	− 1.11E − 10	
F12	A	4.99E − 02	2.92E − 02	6.48E − 02	5.51E − 02	3.44E − 10	1.86 E − 02
	SD	2.73E − 02	2.68E − 02	3.00E − 02	3.02E − 01	1.31E − 09	2.06 E − 02
	Best	1.63E − 06	1.11E − 05	2.24E − 06	3.18E − 02	4.18E − 14	7.31E − 06
	Worst	1.49E − 01	9.63E − 02	1.62E − 01	1.65E − 01	2.65E − 05	3.54E − 01

Table 6 (continued)

Sl. No		OMFO	LMFO	WCMFO	WEMFO	SMFO	h-MFOBOA
F13	A	6.73E − 30	2.50E − 01	1.27E − 05	1.45E − 52	1.39E − 02	− 3.62
	SD	3.62E − 29	7.50E − 02	6.17E − 06	3.22E − 52	5.50E − 02	1.05 E − 01
	Best	1.56E − 66	1.06E − 01	4.24E − 06	7.00E − 67	5.00E − 07	− 3.62
	Worst	1.98E − 28	4.00E − 01	2.42E − 05	9.91E − 52	8.91E − 01	− 3.62
F14	A	1.05E − 01	1.11E − 01	1.06	1.27E − 01	4.72E − 01	− 5.11E − 03
	SD	6.72E − 02	1.08E − 01	1.06	8.83E − 02	3.72E − 01	0
	Best	1.11E − 02	6.13E − 03	1.13E − 01	1.03E − 02	3.03E − 05	− 5.11E − 03
	Worst	2.83E − 01	5.54E − 01	4.62	3.30E − 01	1.30E − 01	− 5.11E − 03
F15	A	7.15E − 51	9.97E − 02	1.58	7.89E − 103	3.97E − 05	− 3.04
	SD	3.91E − 50	2.96E − 04	2.22E − 01	3.58E − 102	2.12E − 04	6.88 E − 12
	Best	2.35E − 123	9.86E − 02	1.19	1.65E − 139	5.65E − 08	− 3.78
	Worst	2.14E − 49	9.98E − 02	2.09	1.93E − 101	2.93E − 02	− 3.78
F16	A	8.80E − 03	4.17E − 04	1.67E − 03	2.25E − 03	1.18E − 03	0
	SD	9.91E − 03	7.50E − 05	4.79E − 03	1.40E − 03	6.61E − 04	0
	Best	6.23E − 04	3.21E − 04	3.07E − 04	4.06E − 04	3.06E − 07	0
	Worst	3.55E − 02	6.19E − 04	2.03E − 02	5.17E − 03	2.17E − 02	0
F17	A	7.1	4.39	5.68	4.55	1.87	0
	SD	4.01	3.09	5.02	2.65	8.92E − 01	0
	Best	1.99	9.98E − 01	9.98E − 01	9.98E − 01	4.98E − 02	0
	Worst	1.26E+01	1.18E+01	1.64E+01	1.01E+01	2.01E+01	0
F18	A	1.51E+01	3.02	1.11E+01	3.56	3.83	− 2.92
	SD	1.94E+01	3.62E − 02	1.75E+01	8.54E − 01	2.48	5.30 E − 02
	Best	3	3	3	3	3	− 2.92E − 01
	Worst	9.27E+01	3.16	8.40E+01	6.57	5.57	− 2.92E − 01
F19	A	− 2.35E − 01	− 2.34E − 01	− 3.00E − 01	− 2.47E − 01	− 2.98	− 1.65 E+ 03
	SD	2.25E − 02	1.76E − 02	2.25E − 16	1.98E − 02	4.61E − 01	1.53 E+02
	Best	− 2.72E − 01	− 2.67E − 01	− 3.00E − 01	− 2.75E − 01	− 2.75E − 01	− 1.87E+03
	Worst	− 1.89E − 01	− 1.86E − 01	− 3.00E − 01	− 2.01E − 01	− 2.01E − 01	1.64E+01
F20	A	− 3.06	− 3.06	− 3.27	− 3.07	− 3.02	3.49 E − 12
	SD	6.16E − 02	8.09E − 02	5.92E − 02	6.68E − 02	8.66E − 02	9.23 E − 12
	Best	− 3.22	− 3.22	− 3.32	− 3.22	− 3.22	6.89E − 16
	Worst	− 2.95	− 2.91	− 3.2	− 2.95	− 2.99	3.54E − 11
F21	A	− 4.82	− 4.77	− 6.3	− 4.97	− 4.27	− 1.54 E+ 01
	SD	9.99E − 01	8.95E − 01	3.33	8.54E − 01	1.03	1.24
	Best	− 7.42	− 8.02	− 1.01E+01	− 8.18	− 8.59	− 1.54E+01
	Worst	− 3.34	− 3.3	− 2.63	− 3.93	− 3.63	1.34E+01
F22	A	− 5.15	− 4.84	− 5.77	− 5.37	− 4.14	0
	SD	1.47	8.41E − 01	3.4	1.1	8.62E − 01	0
	Best	− 9.17	− 7.76	− 1.04E+01	− 8.73	− 7.78	0
	Worst	− 3.78	− 3.68	− 1.83	− 4.3	− 3.39	0
F23	A	− 5.06	− 4.84	− 4.5	− 5.56	− 4.34	0
	SD	1.18	8.98E − 01	2.89	1.23	1.08	0
	Best	− 8.23	− 7.34	− 1.05E+01	− 9.28	− 8.38	0
	Worst	− 3.36	− 3.64	− 1.85	− 4.23	− 3.53	0

19, 17, 17, and 17 benchmark functions, respectively, similar results can be seen in 2, 0, 2, 2 and 2 occasions, respectively, and worse values are achieved in 4, 4, 4, 4 and 4 benchmark functions respectively.

Friedman and Wilcoxon signed rank test are used to analyze the performance of proposed h-MFOBOA algorithm. In this paper, for each benchmark function Friedman test is used from the average performance of algorithms. The IBM-SPSS software has been used for finding the average rank.

Table 7 Performance assessment of h-MFOBOA and MFO variants on 23 benchmark functions

	OMFO	LMFO	WCMFO	WEMFO	SMFO
Superior to	17	19	17	17	22
Similar to	2	0	2	2	2
Inferior to	4	4	4	4	0

Table 8 Friedman rank test of h-MFOBOA and MFO variants on 23 benchmark functions

Algorithm	Mean rank	Rank
h-MFOBOA	**2.48**	**1**
OMFO	4.22	5
LMFO	4.13	4
WCMFO	3.70	3
WEMFO	3.24	2
SMFO	3.24	2

The boldface represents the best value

Table 9 Wilcoxon's test for h-MFOBOA and MFO variants on 23 benchmark functions ($\alpha = 0.05$)

h-MFOBOA vs. Algorithm	*p* value	R+	R-	Winner
OMFO	0.322	144	87	h-MFOBOA
LMFO	0.301	172	104	h-MFOBOA
WCMFO	0.375	141	90	h-MFOBOA
WEMFO	0.455	137	94	h-MFOBOA
SMFO	0.455	137	94	h-MFOBOA

The outcomes of the Friedman rank test between h-MFOBOA, OMFO, LMFO, WCMFO, WEMFO and SMFO for benchmark functions is presented in Table 8. From Table 8, it is clearly visible that h-MFOBOA obtains least rank among other algorithms at 1% relevant.

In Table 9, the outcome of Wilcoxon rank test is demonstrated at the 5% relevant point between h-MFOBOA, OMFO, LMFO, WCMFO, WEMFO and SMFO for twenty-three benchmark functions in Table 9. From Table 9, all the R + (positive rank) values higher than R − (negative) values which demonstrate the superiority of h-FOBOA among other competitors. Moreover, to examine the convergence speed of the proposed algorithm, convergence graphs of some of the randomly chosen functions have been presented in Fig. 6, which clearly indicate that the suggested h-MFOBOA has a superior convergence speed than the compared algorithms.

7 Real-World Applications

To assess the efficiency of the h-MFOBOA proposed, two Real-World Problems (RWP) were resolved, such as optimal gas production capacity problem and three-bar truss design problem.

7.1 RWP-1: Optimal Capacity of Gas Production Facilities

This challenging problem has been adapted from [69] and is presented in Appendix-2 with its Mathematical representation. These results are presented in Table 10. In this table, the results of DE, GSA and DE-GSA, BOA are taken from [78] and few variants of the MFO algorithm namely WEMFO, LMFO and OMFO. It has been noted that our approach is more efficient than the other methods.

7.2 RWP-2: Three-bar Truss Design Problem

The above problem is popular in civil engineering field. It is used due to its complex constrained search space [79, 80]. To achieve minimum weight, two parameters of this design problem have been manipulated with respect to the constraints namely buckling, stress and deflection. The mathematical formulation and various components of the three-bar truss design problem are presented in Appendix 3 and Fig. 7 respectively.

Our developed h-MFOBOA method is used to evaluate this design problem and it is compared with existing algorithms in the literature [11], including DEDS, MBA, Tsa, PSO-DE, and CS with few variants of the MFO algorithm such as WEMFO, LMFO and OMFO. Table 11 paraphrases the text by summarising the comparison results. Our suggested h-MFOBOA method outperforms the other three algorithms, as shown in Table 11.

8 Conclusion with Future Direction

To improve the MFO algorithm, the hybrid moth flame optimization (h-MFOBOA) makes use of exploration and exploitation phases. Comparative tests are conducted on several benchmark functions to judge the performance of h-MFOBOA in comparison to the DE, PSO, JAYA, BOA, BSA, and MFO. In addition, this approach has been used to solve engineering problems for validating the proposed h-MFOBOA, which provides superior results to alternative algorithms. According to the simulation results, the proposed algorithm utilizes the global optimum solution, which helps it reach a solution quickly. The algorithm's position

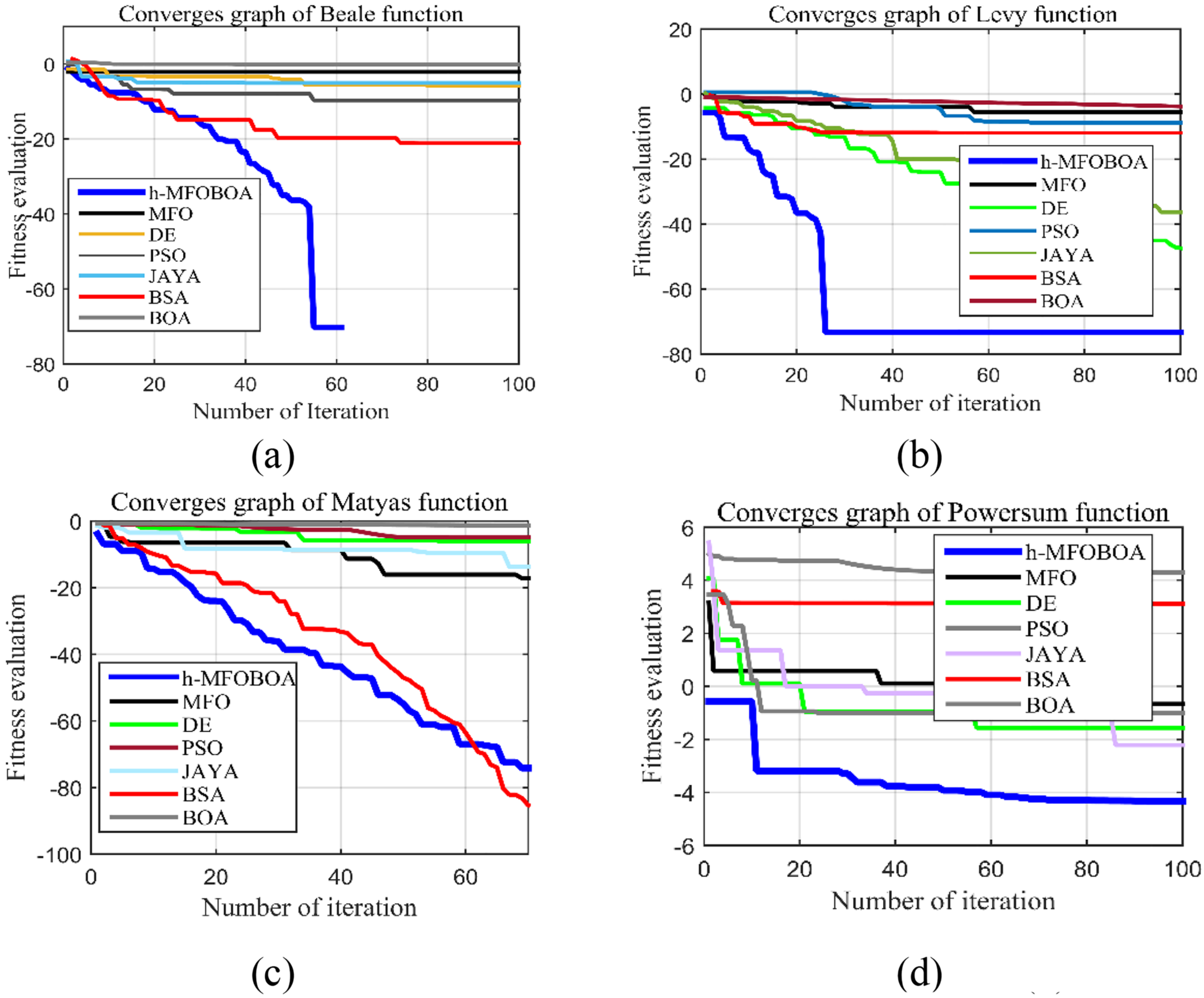

Fig. 6 Convergence graph of h-MFOBOA with MFO, DE, PSO, JAYA, BSA and BOA for (**a**) Beale function, (**b**) Levy function, (**c**) Matyas function and (**d**) Power-sum function

Table 10 Experimental results of h-MFOBOA and some other algorithms on optimal capacity of gas production facilities problem

Algorithm	Optimal variables		Optimal weight
	x_1	x_2	
h-MFOBOA	**17.5**	**600**	**71.4459**
DE	17.5	600	169.844
GSA	17.5	600	169.844
DE-GSA	17.5	600	169.844
MFO	17.5	600	71.4495
BOA	17.5	572.98	71.8010
WEMFO	17.5	598.89	71.4463
LMFO	17.5	597.38	71.4492
OMFO	17.5	599.62	71.4535

The boldface represents the best value

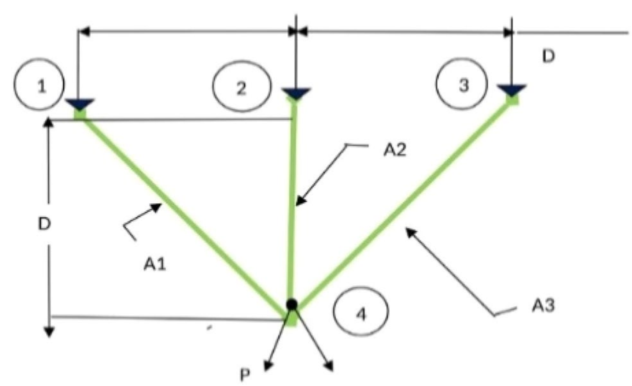

Fig. 7 Three-bar truss design problem

Table 11 Experimental results of h-MFOBOA and some other algorithms on three-bar truss design problem

Algorithm	Optimal variables		Optimal weight
	x_1	x_2	
h-MFOBOA	**0.408966**	**0.288146**	**174.2762166**
Tsa	0.788	0.408	263.68
DEDS	0.78867513	0.40824828	263.8958434
PSO-DE	0.7886751	0.4082482	263.8958433
MBA	0.7885650	0.4085597	263.8958522
MFO	0.7882447709319	0.7882447709319	263.8959796
CS	0.78867	0.40902	263.9716
WEMFO	0.399262577	0.3096396	174.2762410
LMFO	0.401404829	0.30649271	174.2785337
OMFO	0.399478073	0.31071487	174.2769516

The boldface represents the best value

update phase prevents it from being trapped in local optima and stopping before finding a true solution. Then, the proposed method is considered to be a useful way to solve both real-world and engineering design optimization problems.

In future studies, the proposed algorithm may be extended to a more efficient algorithm by adding different learning strategies (for example, opposition based learning, quasi opposition based learning, dynamic opposite learning technique), using non-linear parameter adaption, parameter tuning, etc. It may also be applied to a range of actual optimization problems including vehicle routing, job shop planning, parameter estimation of fuel cell problem, combined economic and emission dispatch problem, image segmentation problem, workflow planning, etc. Moreover, the suggested algorithm may be extended to multi-objective environment to check its capability to explore its possibility to solve multi-objective problems.

Appendix 1

Formulation of twenty-three benchmark functions.

Sl. No.	Functions	Formulation of objective functions	d	Fmin	Search space
Unimodal Benchmark Functions					
F1	Beale	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	0	[-100, 100]
F2	Booth	$f(x) = (2x_1 + x_2 - 5)^2 + (x_1 + 2x_2 - 7)^2$	2	0	[-10, 10]
F3	Matyas	$f(x) = 0.26(x_1{}^2 + x_2{}^2) - 0.48x_1x_2$	2	0	[-10, 10]
F4	SUMSQUARE	$f(x) = \sum_{i=1}^{D} x_i{}^2 \times i$	30	0	[-10, 10]
F5	Zettl	$f(x) = (x - 1^2 + x - 2^2 - 2x_1)^2 + 0.25x_1$	2	-0.00379	[-1, 5]
F6	Leon	$f(x) = 100(x_2 - x_1{}^3)^2 + (1 - x_1)^2$	2	0	[-1.2, 1.2]
F7	Zakhrov	$f(x) = \sum_{j=1}^{d} x_i{}^2 + \left(0.5\sum_{j=1}^{d} jx_j\right)^2 + \left(0.5\sum_{j=1}^{d} jx_j\right)^4$	2	0	[-5, 10]
Multimodal Benchmark Functions					
F8	Bohachevsky	$f(x) = x_1{}^2 + 2x_2{}^2 - 0.3\cos(3\pi x_1) - 0.3$	2	0	[-100, 100]
F9	Bohachevsky 3	$f(x) = x_1{}^2 + 2x_2{}^2 - 0.3\cos(3\pi x_1) - 0.3$	2	0	[-50, 50]
F10	Levy	$f(x) = sin^2(\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2[1 + 10sin^2(\pi x_i + 1)] + (x_D - 1)^2[1 + sin^2(2\pi x_D)]$ Where, $x_i = 1 + \frac{1}{4}(x_i - 1), i = 1, 2, \ldots\ldots D$	30	0	[-10, 10]
F11	Michalewicz	$f(x) = -\sum_{i=1}^{D} \sin(x_i) sin^{2m}(\frac{ix_i^2}{\pi})$, m = 10	10	-9.66015	[0, π]
F12	Alpine	$f(x) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	30	0	[-10, 10]
F13	Schaffers	$f(x) = 0.5 + \frac{sin^2(x_1{}^2 + x_2{}^2) - 0.5}{[1 + 0.001(x_1{}^2 + x_2{}^2)]^2}$	2	0	[-100, 100]

Sl. No.	Functions	Formulation of objective functions	d	Fmin	Search space
F14	Powersum	$f(x) = \sum_{i=1}^{D}\left[\left(\sum_{k=1}^{D}(x_k{}^i) - b_i\right)^2\right]$			
F15	Penalized2	$f(x) = 0.1\left\{10sin^2(\pi x_i) + \sum_{i=1}^{D-1}(x_i-1)^2 [1+10sin^2(3\pi x_{i+1}) + (x_D-1)^2[1+sin^2(2\pi x_D)]]\right\} + \sum_{i=1}^{D} u(x_i,5,100,4)$		0	[-50, 50]
F16	Kowalik	$f(x) = \sum_{j=1}^{11}\left[a_j - \frac{x_1(b_j{}^2+b_jx_2)}{(b_j{}^2-b_jx_3-x_4)}\right]^2$	4	0.0003075	[-5, 5]
F17	Foxholes	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j} + \sum_{i=1}^{D}(x_i - a_{ij})^6\right]^{-1}$	2	3	[-65, 65]
Fixed dimension Multimodal Benchmark functions					
F18	Goldstein and Price	$f(x) = \left[1 + (1+x_1+x_2)^2(10-14x_1-14x_2+6x_1x_2+3x_1{}^2+3x_2{}^2)\right] \times [30+(2x_1-3x_2{}^2)(18-32x_1+12x_1{}^2+48x_2-36x_1x_2+27x_2{}^2)]$	2	3	[-2, 2]
F19	Hartmann3	$f(x) = -\sum_{i=1}^{4}\alpha_i exp(-\sum_{j=1}^{3} a_{ij}\left(x_{j-b_{ij}}\right)^2)$	3	-3.86	[0, 1]
F20	Hartmann6	$f(x) = -\sum_{i=1}^{4}\alpha_i exp(-\sum_{j=1}^{6} a_{ij}\left(x_{j-b_{ij}}\right)^2)$	6	-3.32	[0, 1]
F21	Shekel 5	$f(x) = -\sum_{j=1}^{5}\left[(x-a_i)(x-a_i)^T + c_j\right]^{-1}$	4	-10.1499	[0, 10]
F22	Shekel-7	$f(x) = -\sum_{j=1}^{7}\left[(x-a_i)(x-a_i)^T + c_j\right]^{-1}$	4	-10.3999	[0, 10]
F23	Shekel-10	$f(x) = -\sum_{j=1}^{10}\left[(x-a_i)(x-a_i)^T + c_j\right]^{-1}$	4	-10.5319	[0, 10]

Appendix 2

Optimal Capacity of Gas Production Facilities

$$
\begin{aligned}
Min f(x) = {} & 61.8 + 5.72 \times x_1 \\
& \times 0.2623 \times \left[(40 - x_1) \times \ln\frac{x_2}{200}\right]^{-0.85} \\
& + 0.087 \times (40 - x_1) \times \ln\frac{x_2}{200} \\
& + 700.23 \times x_2^{-0.75}
\end{aligned}
$$

$$x_1 \ge 17.5, x_2 \ge 200, 17.5 \le x_1 \le 40, 300 \le x_2 \le 600;$$

Appendix 3

Three-bar truss problem

$$\vec{k} = \{k_1, k_2,\}$$

Objective function:

$$Min.f(k) = L\left\{k_2 + 2\sqrt{2}k_1\right\}$$

Subject to:

$$h_1(k) = \frac{k_2}{2k_2k_1 + \sqrt{2}k_1^2}P - \sigma \le 0,$$

$$h_2(k) = \frac{k_2 + \sqrt{2}k_1}{2k_2k_1 + \sqrt{2}k_1^2}P - \sigma \le 0,$$

$$h_3(k) = \frac{1}{k_1 + \sqrt{2}k_2}P - \sigma \le 0, Whre 0 \le k_1, k_2 \le 1, and$$

$$and P = 2, L = 100 \& \sigma = 2$$

Data Availability All data generated or analysed during this study are included in the article.

Declarations

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

Ethical Approval No human or animal studies were conducted by any of the authors.

References

1. Holland, J. H. (1992). Genetic algorithms. *Scientific American, 267*, 66–73.
2. Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.
3. Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*, 702–713.
4. Askarzadeh, A. (2014). Bird mating optimizer: An optimization algorithm inspired by bird mating strategies. *Communications in Nonlinear Science and Numerical Simulation, 19*, 1213–1228.
5. Kennedy, J., Eberhart, R. Particle swarm optimization. In: Proceedings of ICNN'95—International conference on neural networks, Perth, Australia, 1995, 1942–1948.
6. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114*, 163–191.
7. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.
8. Cheng, M. Y., & Prayogo, D. (2014). Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers & Structures, 139*, 98–112.
9. Arora, S., Singh, S. (2015). Butterfly algorithm with levy flights for global optimization. In *International Conference on Signal Processing*, *Computing and Control* (*ISPCC*), Waknaghat, India, pp. 220–224.
10. Wang, G. G., Deb, S., & Cui, Z. H. (2019). Monarch butterfly optimization. *Neural Computing and Applications, 31*, 1995–2014.
11. Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.
12. Civicioglu, P. (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation, 219*, 8121–8144.
13. Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations, 7*, 19–34.
14. Li, S. M., Chen, H. L., Wang, M. J., Heidari, A. A., & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems, 111*, 300–323.
15. Wang, G. G. (2018). Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Computing, 10*, 151–164.
16. Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. L. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97*, 849–872.
17. Yang, Y. T., Chen, H. L., Heidari, A. A., & Gandomi, A. H. (2021). Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Systems with Applications, 177*, 114864.
18. Tu, J., Chen, H. L., Wang, M. J., & Gandomi, A. H. (2021). The colony predation algorithm. *Journal of Bionic Engineering, 18*, 674–710.
19. Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*, 495–513.
20. Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences, 179*, 2232–2248.
21. Lam, A. Y., & Li, V. O. (2009). Chemical-reaction-inspired metaheuristic for optimization. *IEEE Transactions on Evolutionary Computation, 14*, 381–399.
22. Zhao, W. G., Wang, L. Y., & Zhang, Z. X. (2019). Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowledge-Based Systems, 163*, 283–304.
23. Zou, F., Wang, L., Hei, X. H., & Chen, D. B. (2015). Teaching–learning-based optimization with learning experience of other learners and its application. *Applied Soft Computing, 37*, 725–736.
24. Li, M. D., Zhao, H., Weng, X. W., & Han, T. (2016). Cognitive behaviour optimization algorithm for solving optimization problems. *Applied Soft Computing, 39*, 199–222.
25. Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120–133.
26. Ahmadianfar, I., Heidari, A. A., Gandomi, A. H., Chu, X. F., & Chen, H. L. (2021). RUN beyond the metaphor: An efficient optimization algorithm based on runge-kutta method. *Expert Systems with Applications, 181*, 115079.
27. Ahmadianfar, I., Heidari, A. A., Noshadian, S., Chen, H. L., & Gandomi, A. H. (2022). INFO: An efficient optimization algorithm based on weighted mean of vectors. *Expert Systems with Applications, 195*, 116516.
28. Singh, P., & Prakash, S. (2019). Optical network unit placement in fiber-wireless (FiWi) access network by whale optimization algorithm. *Optical Fiber Technology, 52*, 101965.
29. Mohanty, B. (2019). Performance analysis of moth flame optimization algorithm for AGC system. *International Journal of Modelling and Simulation, 39*, 73–87.
30. Khairuzzaman, A. K. M., & Chaudhury, S. (2020). Modified moth-flame optimization algorithm-based multilevel minimum cross entropy thresholding for image segmentation. *International Journal of Swarm Intelligence Research, 11*, 123–139.
31. Gupta, D., Ahlawat, A. K., Sharma, A., & Rodrigues, J. J. P. C. (2020). Feature selection and evaluation for software usability model using modified moth-flame optimization. *Computing, 102*, 1503–1520.
32. Muduli, D., Dash, R., & Majhi, B. (2020). Automated breast cancer detection in digital mammograms: A moth flame optimization-based ELM approach. *Biomedical Signal Processing and Control, 59*, 101912.
33. Kadry, S., Rajinikanth, V., Raja, N. S. M., Hemanth, D. J., Hannon, N. M., & Raj, A. N. J. (2021). Evaluation of brain tumor using brain MRI with modified-moth-flame algorithm and Kapur's thresholding: A study. *Evolutionary Intelligence, 14*, 1053–1063.
34. Suja, K. R. (2021). Mitigation of power quality issues in smart grid using levy flight-based moth flame optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing, 12*, 9209–9228.
35. Khalilpourazari, S., & Khalilpourazary, S. (2019). An efficient hybrid algorithm based on water cycle and moth-flame optimization algorithms for solving numerical and constrained engineering optimization problems. *Soft Computing, 23*, 1699–1722.
36. Hongwei, L., Jianyong, L., Liang, C., Jingbo, B., Yangyang, S., & Kai, L. (2019). Chaos-enhanced moth-flame optimization algorithm for global optimization. *Journal of Systems Engineering and Electronics, 30*, 1144–1159.
37. Xu, Y. T., Chen, H. L., Luo, J., Zhang, Q., Jiao, S., & Zhang, X. Q. (2019). Enhanced moth-flame optimizer with mutation strategy for global optimization. *Information Sciences, 492*, 181–203.
38. Xu, Y. T., Chen, H. L., Heidari, A. A., Luo, J., Zhang, Q., Zhao, X. H., & Li, C. Y. (2019). An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks. *Expert Systems with Applications, 129*, 135–155.
39. Kaur, K., Singh, U., & Salgotra, R. (2020). An enhanced moth flame optimization. *Neural Computing and Applications, 32*, 2315–2349.
40. Tumar, I., Hassouneh, Y., Turabieh, H., & Thaher, T. (2020). Enhanced binary moth flame optimization as a feature selection

algorithm to predict software fault prediction. *IEEE Access, 8*, 8041–8055.
41. Gu, W., Xiang, G. Improved moth flame optimization with multi operator for solving real-world optimization problems. 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2021, 2459–2462
42. Ma, L., Wang, C., Xie, N. G., Shi, M., Ye, Y., & Wang, L. (2021). Moth-flame optimization algorithm based on diversity and mutation strategy. *Applied Intelligence, 51*, 5836–5872.
43. Shehab, M., Alshawabkah, H., Abualigah, L., & AL-Madi, N. (2020). Enhanced a hybrid moth-flame optimization algorithm using new selection schemes. *Engineering with Computers, 37*, 2931–2956.
44. Wu, Y., Chen, R. L., Li, C. Q., Zhang, L. G., & Cui, Z. L. (2020). Hybrid symbiotic differential evolution moth-flame optimization algorithm for estimating parameters of photovoltaic models. *IEEE Access, 8*, 156328–156346.
45. Bhesdadiya, R. H., Trivedi, I. N., 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, 553*, 569–577.
46. Kamalapathi, K., Priyadarshi, N., Padmanaban, S., Holm-Nielsen, J. B., Azam, F., Umayal, C., & Ramachandara Murthy, V. K. (2018). A hybrid moth-flame fuzzy logic controller based integrated cuk converter fed brushless DC motor for power factor correction. *Electronics, 7*, 288.
47. Sarma, A., Bhutani, A., Goel, L. Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality. Intelligent Systems Conference (IntelliSys), London, UK, 2017, 52–60.
48. Sahoo, S. K., Saha, A. K., Sharma, S., Mirjalili, S., & Chakraborty, S. (2022). An enhanced moth flame optimization with mutualism scheme for function optimization. *Soft Computing, 26*, 2855–2882.
49. Arora, S., & Singh, S. (2016). An improved butterfly optimization algorithm for global optimization. *Advanced Science, Engineering and Medicine, 8*, 711–717.
50. Sharma, T. K. (2021). Enhanced butterfly optimization algorithm for reliability optimization problems. *Journal of Ambient Intelligence and Humanized Computing, 12*, 7595–7619.
51. Guo, Y. J., Liu, X. J., & Chen, L. (2021). Improved butterfly optimization algorithm based on guiding weight and population restart. *Journal of Experimental & Theoretical Artificial Intelligence, 33*, 127–145.
52. Dhanya, K. M., & Kanmani, M. (2019). Mutated butterfly optimization algorithm. *International Journal of Engineering and Advanced Technology, 8*, 375–381.
53. Li, Y., Yu, X. M., & Liu, J. S. (2022). Enhanced butterfly optimization algorithm for large-scale optimization problems. *Journal of Bionic Engineering, 19*, 554–570.
54. Bahgat, G. A., Fawzy, A. A., Emara, H. M. (2020). An unbiased butterfly optimization algorithm. In L. Pan, J. Liang, & B. Qu (Eds.), Bio-inspired computing: theories and applications. BICTA 2019. Communications in computer and information science, Springer.
55. Lohar, G., Sharma, S., Saha, A. K., Ghosh, S. Optimization of geotechnical parameters used in slope stability analysis by metaheuristic algorithms. *Applications of Internet of Things, 2021*, 223–231.
56. Arora, S., & Singh, S. (2017). An effective hybrid butterfly optimization algorithm with artificial bee colony for numerical optimization. *International Journal of Interactive Multimedia and Artificial Intelligence, 4*, 14–21.
57. Arora, S., & Anand, P. (2019). Binary butterfly optimization approaches for feature selection. *Expert Systems with Applications, 116*, 147–160.
58. Sharma, S., Saha, A. K., Majumder, A., & Nama, S. (2021). MPBOA-A novel hybrid butterfly optimization algorithm with symbiosis organisms search for global optimization and image segmentation. *Multimedia Tools and Applications, 80*, 12035–12076.
59. Sharma, S., Saha, A. K., & Lohar, G. (2021). Optimization of weight and cost of cantilever retaining wall by a hybrid metaheuristic algorithm. *Engineering with Computers*. https://doi.org/10.1007/s00366-021-01294-x
60. Sharma, S., Saha, A. K. (2021) Bosca—A hybrid butterfly optimization algorithm modified with sine cosine algorithm. In *Progress in Advanced Computing and Intelligent Engineering* (Vol. 1198). Singapore: Springer, pp. 360–372.
61. Sharma, S., & Saha, A. K. (2020). m-MBOA: A novel butterfly optimization algorithm enhanced with mutualism scheme. *Soft Computing, 24*, 4809–4827.
62. Liu, G. M., Jia, W. Y., Luo, Y. G., Wang, M. J., Heidari, A. A., Ouyang, J. S., Chen, H. L., & Chen, M. Y. (2020). Prediction optimization of cervical hyperextension injury: Kernel extreme learning machines with orthogonal learning butterfly optimizer and broyden- fletcher-goldfarb-shanno algorithms. *IEEE Access, 8*, 119911–119930.
63. Yu, H. L., Yuan, K., Li, W. S., Zhao, N. N., Chen, W. B., Huang, C. C., Chen, H. L., & Wang, M. J. (2021). Improved butterfly optimizer-configured extreme learning machine for fault diagnosis. *Complexity, 2021*, 1–17.
64. Saka, M., Çoban, M., Eke, İ, Tezcan, S. S., & Taplamacioğlu, M. C. (2021). A novel hybrid global optimization algorithm having training strategy: Hybrid taguchi-vortex search algorithm. *Turkish Journal of Electrical Engineering & Computer Sciences, 29*, 1908–1928.
65. Chakraborty, S., Sharma, S., Saha, A. K., & Chakraborty, S. (2021). SHADE-WOA: A metaheuristic algorithm for global optimization. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2021.107866
66. Singh, N., & Singh, S. B. (2017). Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. *Journal of Applied Mathematics, 2017*, 1–15.
67. Wang, Z. W., Wu, G. M., & Wan, Z. P. (2017). A novel hybrid vortex search and artificial bee colony algorithm for numerical optimization problems. *Wuhan University Journal of Natural Sciences, 22*, 295–306.
68. Nama, S., & Saha, A. K. (2019). A novel hybrid backtracking search optimization algorithm for continuous function optimization. *Decision Science Letters, 8*, 163–174.
69. Yıldız, A. R. (2008). Hybrid Taguchi-harmony search algorithm for solving engineering optimization problems. *International Journal of Industrial Engineering, 15*, 286–293.
70. Nama, S., Saha, A. K., & Ghosh, S. (2017). A hybrid symbiosis organisms search algorithm and its application to real world problems. *Memetic Computing, 9*, 261–280.
71. Chakraborty, S., Saha, A. K., Chakraborty, R., Saha, M., & Nama, S. (2022). HSWOA: An ensemble of hunger games search and whale optimization algorithm for global optimization. *International Journal of Intelligent Systems, 37*, 52–104.
72. Sharma, S., Chakraborty, S., Saha, A. K., Nama, S., & Sahoo, S. K. (2022). mLBOA: A modified butterfly optimization algorithm with lagrange interpolation for global optimization. *Journal of Bionic Engineering*. https://doi.org/10.1007/s42235-022-00175-3
73. Li, C. Q., Niu, Z., Song, Z. S., Li, B. X., Fan, J. G., & Liu, P. X. (2018). A double evolutionary learning moth-flame optimization

for real-parameter global optimization problems. *IEEE Access, 6*, 76700–76727.
74. Apinantanakon, W., Sunat, K. OMFO: a new opposition-based moth-flame optimization algorithm for solving unconstrained optimization problems. In International Conference on Computing and Information Technology (IC2IT) Springer, Cham 22–31.
75. Li, Z. M., Zhou, Y. Q., Zhang, S., & Song, J. M. (2016). Levy-flight moth-flame algorithm for function optimization and engineering design problems. *Mathematical Problems in Engineering, 2016*, 1–22.
76. Shan, W. F., Qiao, Z. G., Heidari, A. A., Chen, H. L., Turabieh, H., & Teng, Y. T. (2021). Double adaptive weights for stabilization of moth flame optimizer: Balance analysis, engineering cases, and medical diagnosis. *Knowledge-Based Systems, 214*, 106728.
77. Chen, C. C., Wang, X. C., Yu, H. L., Wang, M. J., & Chen, H. L. (2021). Dealing with multi-modality using synthesis of Moth-flame optimizer with sine cosine mechanisms. *Mathematics and Computers in Simulation, 188*, 291–318.
78. Muangkote, N., Sunat, K., Chiewchanwattana, S. (2016). Multilevel thresholding for satellite image segmentation with moth-flame based optimization. In *2016 13th International Joint Conference on Computer Science and Software Engineering* (*JCSSE*), KhonKaen, Thailand, pp. 1–6.
79. Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2013). Mine blast algorithm: A new population-based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing, 13*, 2592–2612.
80. Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*, 17–35.