

Article

An Improved Moth-Flame Optimization Algorithm with Adaptation Mechanism to Solve Numerical and Mechanical Engineering Problems

Mohammad H. Nadimi-Shahraki ^{1,2,*}, Ali Fatahi ^{1,2}, Hoda Zamani ^{1,2}, Seyedali Mirjalili ^{3,4,*} and Laith Abualigah ^{5,6}

¹ Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad 8514143131, Iran; fatahi.ali.edu@sco.iaun.ac.ir (A.F.); hoda_zamani@sco.iaun.ac.ir (H.Z.)

² Big Data Research Center, Najafabad Branch, Islamic Azad University, Najafabad 8514143131, Iran

³ Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Brisbane 4006, Australia

⁴ Yonsei Frontier Lab, Yonsei University, Seoul 03722, Korea

⁵ Faculty of Computer Sciences and Informatics, Amman Arab University, Amman 11953, Jordan; laythdyabat@aau.edu.jo

⁶ School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang 11800, Malaysia

* Correspondence: nadimi@iaun.ac.ir (M.H.N.-S.); ali.mirjalili@torrens.edu.au (S.M.)



Citation: Nadimi-Shahraki, M.H.; Fatahi, A.; Zamani, H.; Mirjalili, S.; Abualigah, L. An Improved Moth-Flame Optimization Algorithm with Adaptation Mechanism to Solve Numerical and Mechanical Engineering Problems. *Entropy* **2021**, *23*, 1637. <https://doi.org/10.3390/e23121637>

Academic Editor: Jaesung Lee

Received: 10 October 2021

Accepted: 25 November 2021

Published: 6 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: Moth-flame optimization (MFO) algorithm inspired by the transverse orientation of moths toward the light source is an effective approach to solve global optimization problems. However, the MFO algorithm suffers from issues such as premature convergence, low population diversity, local optima entrapment, and imbalance between exploration and exploitation. In this study, therefore, an improved moth-flame optimization (I-MFO) algorithm is proposed to cope with canonical MFO's issues by locating trapped moths in local optimum via defining memory for each moth. The trapped moths tend to escape from the local optima by taking advantage of the adapted wandering around search (AWAS) strategy. The efficiency of the proposed I-MFO is evaluated by CEC 2018 benchmark functions and compared against other well-known metaheuristic algorithms. Moreover, the obtained results are statistically analyzed by the Friedman test on 30, 50, and 100 dimensions. Finally, the ability of the I-MFO algorithm to find the best optimal solutions for mechanical engineering problems is evaluated with three problems from the latest test-suite CEC 2020. The experimental and statistical results demonstrate that the proposed I-MFO is significantly superior to the contender algorithms and it successfully upgrades the shortcomings of the canonical MFO.

Keywords: optimization; metaheuristic algorithms; swarm intelligence algorithm; moth-flame optimization; mechanical engineering problems

1. Introduction

In the majority of real-world optimization problems, a large number of decision variables are interacted with together, which is a very time-consuming process for finding an exact solution [1–7]. Metaheuristic algorithms have been widely used in recent years to approximate near-optimal solutions for real-world problems in various applications such as discrete optimization [8–17], continuous optimization [18–22], and constrained engineering problems [23–33]. Moreover, a novel research field has emerged in this area which successfully combines machine learning and swarm intelligence approaches to obtain outstanding results in different areas [34–36]. Metaheuristic algorithms can be classified into two main categories of non-nature-inspired and nature-inspired [2]. Simulated annealing (SA) [37], tabu search (TS) [38], adaptive dimensional search (ADS) [39], and iterated local search (ILS) [40] are some well-known non-nature-inspired metaheuristic algorithms. Although

these algorithms have demonstrated remarkable local search capabilities, they may easily be trapped in local optimum in complex problems [2,3].

Nature-inspired algorithms consist of three main categories: evolutionary, physics-based, and swarm intelligence (SI). Evolutionary algorithms are mostly inspired by Darwin's theory of evolution. Some examples are: genetic algorithm (GA) [41,42], genetic programming (GP) [43], differential evolution (DE) [44], evolution strategy (ES) [45], and quantum-based avian navigation optimizer (QANA) [46]. Ensemble of mutation strategies and parameters in differential evolution (EPSDE) algorithm [47], multi-population ensemble DE (MPEDE) [48], ensemble of differential evolution variants (EDEV) [49], and multi-trial vector-based differential evolution (MTDE) [50] are some successful improvements on evolutionary algorithms. Physics-based algorithms propose meaningful search strategies inspired by physics and mathematics laws to solve optimization problems. The big bang–big crunch (BB-BC) [51], charged system search (CSS) [52], ray optimization (RO) [53], atom search optimization (ASO) [54], arithmetic optimization algorithm (AOA) [55], and atomic orbital search (AOS) [56] are some of the successful physics-based algorithms in the literature.

Swarm intelligence (SI) algorithms are mainly derived from the social interaction behavior of terrestrial animals, aquatic animals, birds, and insects in nature [57]. Grey wolf optimizer (GWO) [58], chimp optimization algorithm (ChOA) [59], and gorilla troops optimizer (GTO) [60] are inspired by the behavior of terrestrial animals to solve optimization problems. Despite their simplicity and broad use, they may suffer from common drawbacks such as low population diversity, sinking into local optimum, and premature convergence problems. Therefore, there have been many improvements with different approaches applied to overcome their weaknesses [61–64]. Some SI algorithms are inspired by the behavior of aquatic animals such as prey besieging and foraging, which have been modeled in krill herd (KH) [65], dolphin echolocation (DE) [66], and whale optimization algorithm (WOA) [67]. The social intelligence behaviors of birds and insects encourage researchers to propose a new generation of SI algorithms that are modeled by foraging, hunting, and navigation behavior. Ant colony optimization (ACO) [68], ant lion optimizer (ALO) [69], social spider algorithm (SSA) [70], crow search algorithm (CSA) [71], African vultures optimization algorithm (AVOA) [72], Aquila optimizer (AO) [73], and moth-flame optimization (MFO) [74] are popular SI algorithms inspired by the behaviors of birds and insects.

The moth-flame optimizer (MFO) is a prominent SI algorithm inspired by the moths' locomotion toward the light source. The MFO is appealing because of its ease of implementation, acceptable time complexity, and small number of parameters, which make it applicable in real-world optimization problems such as image segmentation [75–77], feature selection [78–82], food processing [83–85], and engineering optimization [86–91]. Consequently, many MFO variants have been developed such as Lévy-flight moth-flame optimization (LMFO) [92], non-dominated sorting moth flame optimization (NS-MFO) [93], enhanced moth-flame optimization (EMFO) [94], water cycle–moth-flame optimization (WCMFO) [95], and sine-cosine moth-flame optimization (SMFO) [96]. However, MFO and its variants cannot satisfy the needs of the optimization process for challenging problems, and they still suffer from some weaknesses such as low population diversity [97,98], premature convergence, local optima trapping [99,100], and imbalance between exploration and exploitation [101]. The main reason for these MFO's weaknesses is that the majority of moths are trapped in the local optima in the early iterations which results in low population diversity. The question of this study is how moths can escape the local optima trapping and be moved to promising zones?

This paper proposes an improved MFO algorithm named I-MFO which uses moths' memory mechanism and an adapted version of the wandering around search (WAS) strategy introduced in our prior study [57] to find and possibly free trapped moths. In the I-MFO algorithm, trapped moths are detected by comparing their best experienced flame fitness (F_{best}) with their current positions' fitness (OM). If the current position of

each moth is not better than its memory, the moth is considered to be a trapped moth, and the adapted wandering around search (AWAS) strategy is employed to possibly free it from local optima by performing some random short flights, which also leads to amelioration of the premature convergence. Moreover, the f_l parameter is used to strike a balance between exploration and exploitation by limiting the moths' flight range. In the end, the new position and its fitness value replace the previous ones if the new fitness value is better than its F_{best} .

The proposed I-MFO is comprehensively investigated by the benchmark functions CEC 2018 [102] in different dimensions of 30, 50, and 100, and compared with the state-of-the-art metaheuristic algorithms and three MFO variants including SA, continuous genetic algorithm (CGA) [42], GWO, MFO, WOA, LMFO, WCMFO, ChOA, AOA, and SMFO. The results demonstrate that the I-MFO can avoid local optima trapping, maintain population diversity, mitigate premature convergence, and strike a balance between exploration and exploitation. The I-MFO algorithm is statistically evaluated by the Friedman test and post hoc analysis to prove the superiority of the algorithm. Moreover, the applicability of I-MFO to solve real-world optimization problems was evaluated by three mechanical engineering problems from the latest test-suite CEC 2020 [103]. All experimental evaluations and statistical tests indicate that the I-MFO algorithm outperforms contender algorithms with an overall effectiveness of 92%.

In the rest of this study, the MFO and its variants are discussed in Section 2. Section 3 illustrates the structure of the proposed I-MFO. The results of I-MFO in solving CEC 2018 benchmark test functions are given and analyzed in Section 4, and the results are proven by statistical analysis in Section 5. The applicability of the proposed I-MFO for solving real-world mechanical engineering problems is tested by three problems from the latest test-suite CEC 2020 in Section 6. Finally, conclusions and future directions are given in Section 7.

2. Related Works

In this section, the MFO algorithm is first described in detail, and then its variants are reviewed from the perspective of overcoming the MFO weaknesses.

The MFO algorithm proposed in [74] is a population-based SI algorithm that mimics moths' navigation behavior toward light sources. The moths navigate toward the real-light source (moon) with a straight path and a fixed angle which is called transverse orientation. Moreover, moths are highly attracted to artificial lights such as flames, and because of the close distance, they change their flight angles continuously, which forms a spiral path. This behavior is modeled by the MFO algorithm to solve optimization problems and is reviewed in [104–106]. In this algorithm, both moths and flames are considered as solutions. First, moths scatter in the search space randomly and their positions are saved in a matrix M , where rows indicate the number of moths (N), and columns represent dimensions (D). Then, the fitness value of M is evaluated and stored in an array OM as represented below.

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,D} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,D} \\ \vdots & \vdots & \vdots & \vdots \\ m_{N,1} & m_{N,2} & \cdots & m_{N,D} \end{bmatrix} \quad OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_N \end{bmatrix}$$

The positions and fitness values of flames are considered in a matrix F and an array OF , respectively. In the first iteration, the OF is initiated based on ascending order of OM , and the corresponding sorted positions are assigned to the matrix F . In the next iterations, the F will be updated by the best N search agents from F and current M populations. In the optimization process, each moth $M_i(t)$ in the current iteration t moves around its corresponding flame F_j using a logarithmic spiral defined in Equation (1), where Dis_i is

computed by Equation (2), b determines the shape of the logarithmic spiral, and k is a random number value between interval $(-1, 1)$.

$$M_i(t) = Dis_i \cdot e^{bk} \cdot \cos(2\pi k) + F_j(t) \quad (1)$$

$$Dis_i(t) = |F_j(t) - M_i(t)| \quad (2)$$

In this algorithm, the number of flames is computed by Equation (3), where t indicates the current iteration, and N and $MaxIt$ are the number of search agents and the maximum number of iterations, respectively.

$$flame_{no} = round\left(N - t \times \frac{N - 1}{MaxIt}\right) \quad (3)$$

The MFO algorithm is an effective problem solver that is widely applied for real-world optimization problems. However, MFO is prone to being trapped in local optimum and suffers premature convergence due to its loss of population diversity and imbalance between the two tendencies of exploration and exploitation. Therefore, many variants have been proposed to boost the MFO algorithm, which can be categorized in improved algorithms based on using new search strategies or operators and hybrid-based improvements with other algorithms. Figure 1 shows the classification of SI algorithms and MFO variants.

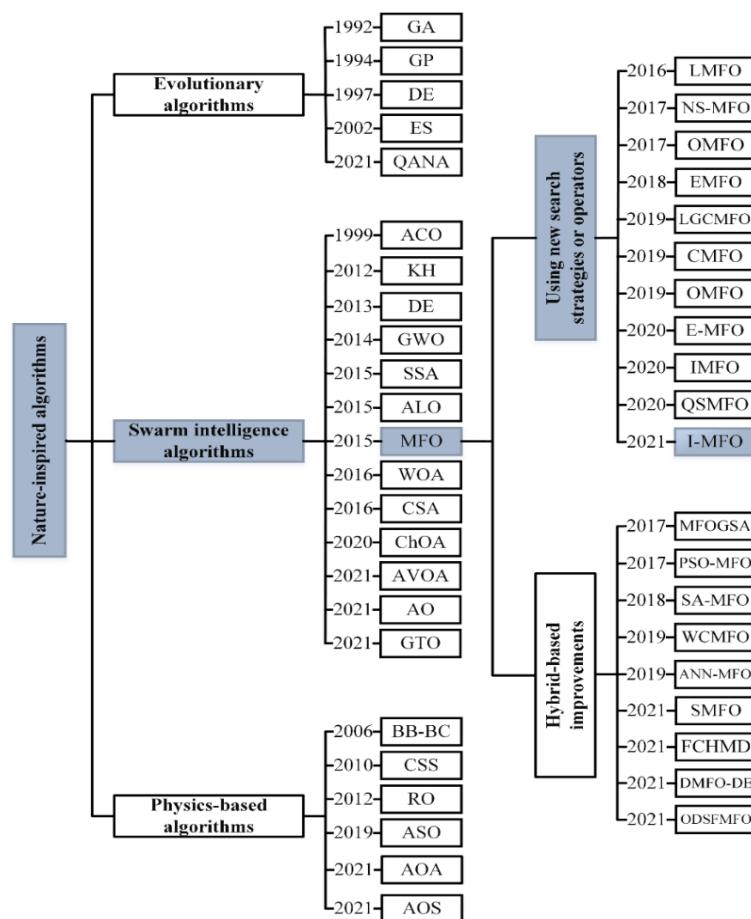


Figure 1. The classification of nature-inspired algorithms and MFO variants.

Many algorithms have been proposed by using new search strategies or operators in the canonical MFO. Li et al. [92] proposed the LMFO algorithm by employing the Lévy-flight strategy to increase the population diversity. Savsani et al. [93] proposed the effective non-dominated moth-flame optimization algorithm (NS-MFO) to solve multi-objective

problems using the elitist non-dominated sorting method. The opposition-based moth-flame optimization (OMFO) [107] presents an opposition-based scheme in the canonical MFO to avoid local optimum and increase global exploration. In the EMFO [94], the Gaussian mutation (GM) was added to MFO to increase the diversity. In LGC-MFO [100], Xu et al. used new operators such as Gaussian mutation (GM), Lévy mutation (LM), and Cauchy mutation (CM) to boost exploration and exploitation capabilities and encounter the local optima trapping of the MFO. In addition, Hongwei et al. [99] presented the chaos-enhanced moth-flame optimization (CMFO) with ten chaotic maps to cope with the MFO deficiency. Sapre et al. [108] brought up OMFO to cope with premature convergence and local optima trapping by proposing a combination of opposition-based, Cauchy mutation and evolutionary boundary constraint handling. In 2020, Kaur et al. [97] proposed E-MFO by adding a Cauchy mutation (CM) to improve the distribution of the algorithm in the search space. An improved moth-flame optimization (IMFO) [98] algorithm proposes a new flame generation strategy and divides optimization iterations into three phases to encounter low population diversity and enhance MFO's search balance, respectively. An improved MFO algorithm called QSMFO was proposed by [109] to boost MFO's exploitation capabilities while enhancing the exploration rate by introducing the simulated annealing strategy and quantum rotation gate, respectively.

Some variants proposed hybrid-based improvements to the MFO algorithm to boost its performance. MFOGSA [83] is a combination of MFO with gravitational search algorithm (GSA) to utilize MFO's exploration and GSA's exploitation capabilities. Bheshdadiya et al. [110] proposed a hybrid PSO-MFO algorithm to solve optimization problems. SA-MFO [111] combines MFO with simulated annealing (SA) to overcome local optima trapping and low convergence rate. Khalilpourazari et al. [95] proposed WCMFO to encounter MFO's entrapping at local optima and low convergence rate, while taking advantage of the water cycle algorithm (WCA). A combination of MFO and artificial neural network (ANN-MFO) was proposed by Singh et al. [112] to solve multi-objective problems in magnetic abrasive finishing of aluminum. Chen et al. [96] introduced SMFO to improve the exploration capability of MFO by integrating it with the sine cosine strategy. An enhanced MFO algorithm was proposed by MP Dang et al. [113] which is a hybridization of MFO and three different methods to solve the design problem of a flexure hinge. Mittal [114] brought up an enhanced moth-flame optimization by integrating MFO and variable neighborhood search to boost search capabilities and convergence accuracy of the canonical MFO. In a recent study, Abd Elaziz et al. [115] proposed the FCHMD algorithm which is a hybridization of Harris hawks optimizer and MFO. In this algorithm, fractional-order Gauss and 2xmod1 chaotic maps are used to generate the initial population. Moreover, the FCHMD algorithm ameliorates premature convergence and stagnation in local optima by applying evolutionary population dynamics. Ahmed et al. [116] brought up DMFO-DE which is a discrete hybrid algorithm developed by integrating differential evolution and MFO to encounter the local optima problem and ameliorate the convergence speed and prevent the local optima problem. Li et al. [117] proposed the ODSFMFO algorithm which consists of an improved flame generation mechanism based on opposition-based learning (OBL) and differential evolution (DE) algorithm, and an enhanced local search mechanism based on shuffled frog leaping algorithm (SFLA) and death mechanism.

Based on the above review on MFO and its proposed variants, the most serious drawbacks of MFO are premature convergence, getting stuck in local optimum, low population diversity, and deficient balance between exploration and exploitation. Therefore, in this study, the improved moth-flame optimization (I-MFO) algorithm is proposed to encounter MFO's shortcomings by introducing a memory mechanism and an adapted version of the wandering around search (WAS) strategy [57], called AWAS strategy, to the canonical MFO.

3. Proposed Algorithm

The proposed improved moth-flame optimization (I-MFO) algorithm is boosted using a moth memory mechanism and the adapted wandering around search (AWAS) strategy to

overcome the mentioned shortcomings of the canonical MFO algorithm. The moth memory mechanism is inspired by moths' behavior in nature in remembering their experiences [118], which is defined by Definition 1. Moreover, the AWAS strategy is introduced in Definition 2, to possibly escape the trapped moths from the local optima and alleviate the premature convergence. The pseudo-code and the flowchart of the proposed I-MFO are shown in Algorithm 1 and Figure 2, respectively.

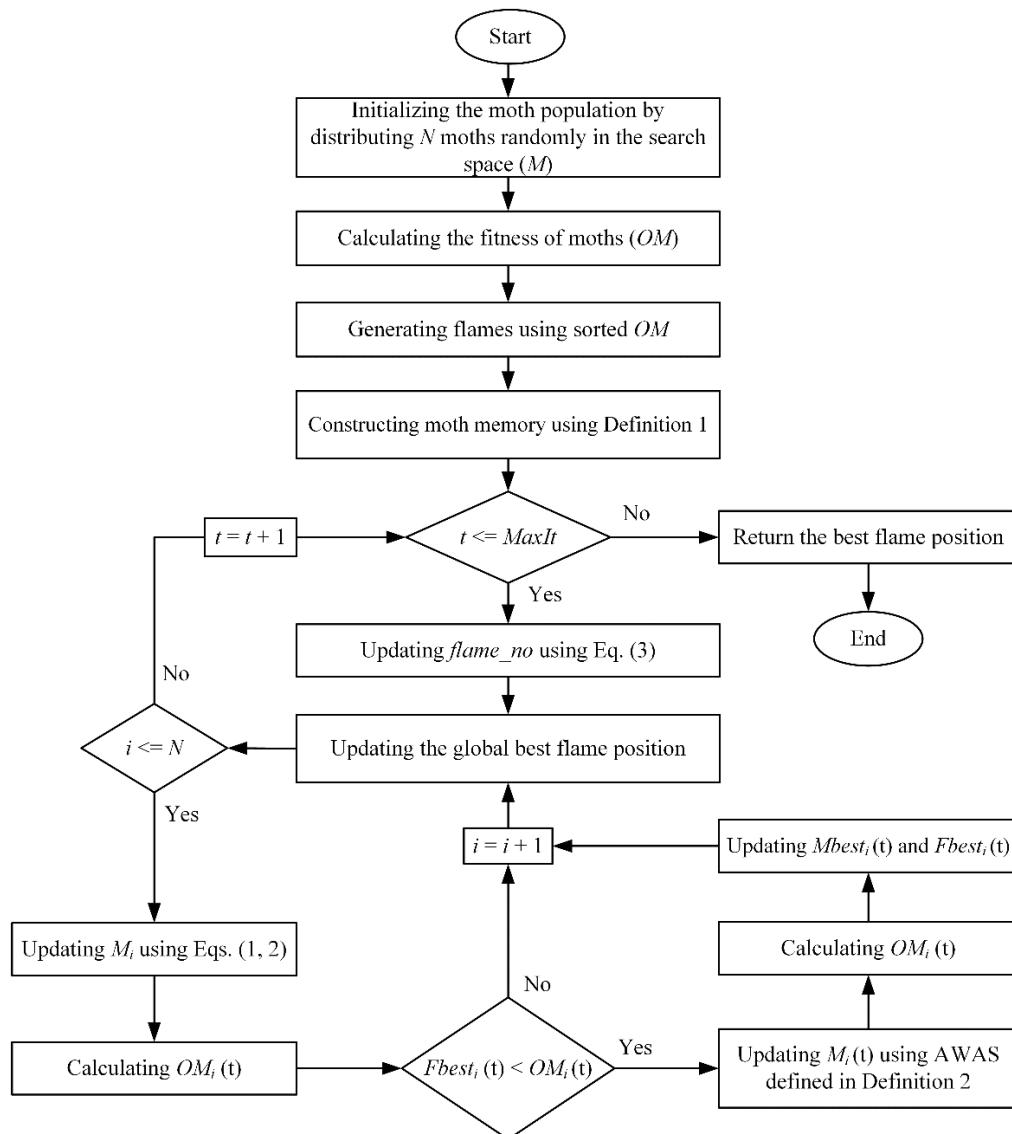


Figure 2. The flowchart of improved moth-flame optimization (I-MFO) algorithm.

Definition 1 (Moth memory construction). Suppose $\text{Mem} = \{\text{Mem}_1, \text{Mem}_2, \dots, \text{Mem}_i, \dots, \text{Mem}_N\}$ is finite set of N moths' memories. The moth memory M_i is denoted by $\text{Mem}_i = (M_{\text{best}}, F_{\text{best}})$, where M_{best} is the best position of M_i obtained so far, and F_{best} is the fitness value of M_{best} . In the first iteration t , the best position $M_{\text{best}}(t=1) \leftarrow M_i(t=1)$ and $F_{\text{best}}(t=1) \leftarrow OM_i(t=1)$. For the rest of the iterations, $M_{\text{best}}(t>1) \leftarrow M_i(P)$ and $F_{\text{best}}(t>1) \leftarrow OM_i(P)$ such that $\{OM_i(P) < F_{\text{best}}, P = 2, \dots, t\}$. The moth memory construction is formulated in Equation (4).

$$\begin{aligned}
 & \text{If } t = 1 \text{ then } Fbest_i(t) \leftarrow OM_i(t) \text{ and } Mbest_i(t) \leftarrow M_i(t) \\
 & \text{If } t > 1 \text{ then } Fbest_i(t) \leftarrow OM_i(P) \text{ and } Mbest_i(t) \leftarrow M_i(P), \\
 & \quad \text{Such that } \nexists s \in \{2, \dots, t\}, OM_i(s) < OM_i(P)
 \end{aligned} \tag{4}$$

Algorithm 1. The pseudo-code of I-MFO**Algorithm of improved moth-flame optimization (I-MFO)****Input:** Maximum iterations ($MaxIt$), Number of moths (N), and Dimension size (D).**Output:** The best flame position and its fitness value.

```

1  Begin
2      Randomly distributing  $M$  moths in the  $D$ -dimensional search space.
3      Calculating moths' fitness ( $OM$ ).
4      Set  $t = 1$ .
5       $OF \leftarrow \text{sort}(OM)$ .
6       $F \leftarrow \text{sort}(M)$ .
7      Defining the moth memory  $Mbest$  and  $Fbest$  using Definition 1.
8      While  $t \leq MaxIt$ 
9          Updating  $F$  and  $OF$  by the best  $N$  moths from  $F$  and current  $M$ .
10         Updating  $flame\_no$  using Equation (3)
11         For  $i = 1: N$ 
12             Computing the distance between moth  $M_i(t)$  and flame  $F_j(t)$  using
13             Equation (2).
14             Updating the position of  $M_i(t)$  using Equation (1).
15             Computing the fitness value of  $M_i(t)$  and update  $OM_i(t)$ .
16             If  $Fbest_i(t) < OM_i(t)$ 
17                 Selecting a random moth  $M_r(t)$ .
18                 Updating the position of  $M_i(t)$  using AWAS defined in Definition 2.
19                 Updating the fitness value  $OM_i(t)$ .
20             End if
21             Updating the moth memory  $M_i$  using Definition 1.
22         End for
23         Updating the position and fitness value of the global best flame.
24          $t = t + 1$ .
End while

```

Definition 2 (AWAS strategy). Consider $TM(t) = \{M_1, \dots, M_i, \dots\}$ as a finite set of moths trapped in the current iteration t such that M_i could not dominate its Mem_i ($OM_i(t) > Fbest_i$). Then, to possibly free the trapped moth $M_i(t+1)$ from the local optimum, its new position is computed by Equation (5), where $F_{gbestj}(t)$ is the j th dimension of the global best flame, r_i is a random number between interval $(0, 1)$, $M_{rj}(t)$ is the value of a random moth position. The flight length $fl_i(t)$ for moth M_i is computed by Equation (6), where δ_1 and δ_2 are defined by the user, NF is the number of flights determined randomly in $[1, D]$, and q is the current flight number. In fact, using AWAS strategy with the random NF provides advantage through which the trapped moth M_i can be moved to a better position.

$$M_{ij}(t+1) = F_{gbestj}(t) + r_i \times fl_i(t) \times (M_{rj}(t) - M_{ij}(t)) \tag{5}$$

$$fl_i(t) = \delta_1 - q \times \left(\frac{\delta_2}{NF} \right) \tag{6}$$

4. Numerical Experiment and Analysis

In this section, the performance of the proposed I-MFO has been evaluated using the CEC 2018 [102] benchmark. Moreover, the proposed algorithm was compared with the state-of-the-art metaheuristic algorithms including SA [37], CGA [42], GWO [58], WOA [67], ChOA [59], AOA [55], canonical MFO [74], and its variants such as LMFO [92],

WCMFO [95], and SMFO [96]. The parameter settings of comparative algorithms are adjusted as in their original papers and are reported in Table 1. The obtained results are reported in Tables 2–4, where the bold values show the winner algorithm. Furthermore, at the end of each table, the symbols W, T, and L demonstrate the number of wins, ties, and losses of each algorithm, respectively.

Table 1. Parameter settings of the I-MFO and other contender algorithms.

Algorithms	Parameter Settings
SA	$T_0 = 10$.
CGA	$IPMut = 0.9$, $PXcross = 0.5$.
GWO	The parameter a is linearly decreased from 2 to 0.
MFO	$b = 1$, a is decreased linearly from -1 to -2 .
WOA	α variable decreases linearly from 2 to 0, $b = 1$.
LMFO	$\beta = 1.5$, μ and v are normal distributions, Γ is the gamma function.
WCMFO	The number of rivers and sea = 4.
ChOA	f decreases linearly from 2 to 0.
AOA	$\mu = 0.5$, $\alpha = 5$.
SMFO	r_4 = random number between interval $(0, 1)$.
I-MFO	$\delta_1 = 2.02$, $\delta_2 = 1.08$, NF = random number between 1 and D .

4.1. Benchmark Test Functions and Experimental Environment

The performance of the proposed algorithm is evaluated using the CEC 2018 benchmark functions with various dimensions of 30, 50, and 100. This benchmark contains 29 test functions with a diverse set of characteristics: unimodal, simple multimodal, hybrid, and composition. Test functions F_1 and F_3 are unimodal functions and they are adequate for evaluating the exploitation of algorithms. Test functions F_4 – F_{10} are multimodal with many local optima which are suitable to assess the exploration abilities of algorithms. Test functions F_{11} – F_{20} are hybrid and F_{21} – F_{30} are composition functions that can evaluate the local optima avoidance ability and balance between exploration and exploitation.

Due to the randomization of SI algorithms and to guarantee that the comparisons are fair, all experiments for each function are repeated 30 times separately on a laptop with characteristics: Intel Core i7-10750H CPU (2.60 GHz) and 24 GB of memory. The MATLAB programming language version R2020a and Windows 10 operating system were used to conduct all experiments. All algorithms were run under the same conditions, with the population size (N) 100 and the maximum number of iterations ($MaxIt$) $(D \times 10^4)/N$.

4.2. Exploitation and Exploration Analysis

In this experimental evaluation, the unimodal functions F_1 and F_3 are considered to assess the exploitation abilities, while the multimodal test functions F_4 – F_{10} are dedicated to evaluating the exploration capabilities.

According to Table 2, the results prove that the I-MFO provides superior exploitation abilities around the optimum solution for all dimensions, particularly on test function F_1 . The results of multimodal test functions are evidence that the exploration capability of the I-MFO significantly outperforms all other algorithms in different dimensions. The comparison and reported results lead to the conclusion that the proposed I-MFO has an effective exploration ability due to the randomness movements of trapped moths by the AWAS strategy. Meanwhile, considering the best flame position and limited value of flight length (fl) causes the exploitation ability to remain functional in the course of iterations. Moreover, the comparison of fitness distribution is shown by box and whiskers diagrams in Figure 3. The diagrams predominantly demonstrate that the proposed I-MFO can find the best solutions during the optimization process, which verifies that its exploration and exploitation abilities are more sufficient than other competitors.

Table 2. Comparison of optimization results obtained from unimodal and multimodal test functions.

F	D	Metrics	SA (1983)	CGA (2000)	GWO (2014)	MFO (2015)	WOA (2016)	LMFO (2016)	WCMFO (2019)	ChOA (2020)	AOA (2021)	SMFO (2021)	I-MFO	
F ₁	30	Avg	2.251×10^{10}	3.794×10^{10}	8.223×10^8	6.952×10^9	1.906×10^6	2.402×10^7	1.328×10^4	2.238×10^{10}	3.943×10^{10}	3.091×10^{10}	5.859 × 10³	
		Min	1.897×10^{10}	2.437×10^{10}	4.404×10^7	1.027×10^9	5.654×10^5	1.731×10^7	1.214×10^2	1.123×10^{10}	2.791×10^{10}	2.010×10^{10}	1.488×10^2	
	50	Avg	7.170×10^{10}	1.126×10^{11}	4.522×10^9	3.099×10^{10}	7.172×10^6	1.091×10^8	2.826×10^4	4.407×10^{10}	9.968×10^{10}	6.933×10^{10}	1.430 × 10⁴	
	100	Avg	2.830×10^{11}	3.547×10^{11}	3.207×10^{10}	1.173×10^{11}	3.677×10^7	7.525×10^8	2.017×10^5	1.457×10^{11}	2.617×10^{11}	1.908×10^{11}	2.881 × 10⁴	
	30	Avg	1.439×10^5	1.095×10^5	2.993×10^4	1.009×10^5	1.715×10^5	2.786×10^3	1.887×10^3	5.221×10^4	7.488×10^4	8.189×10^4	6.106 × 10²	
	50	Avg	2.994×10^5	2.241×10^5	7.147×10^4	1.650×10^5	6.180×10^4	3.151×10^4	1.150×10^4	1.306×10^5	1.648×10^5	1.775×10^5	1.460×10^4	
	100	Avg	8.110×10^5	5.742×10^5	2.023×10^5	4.556×10^5	5.928×10^5	2.495×10^5	7.361×10^4	3.071×10^5	3.330×10^5	3.366×10^5	7.625×10^4	
	30	Avg	1.587×10^3	7.327×10^3	5.441×10^2	9.082×10^2	5.476×10^2	4.928×10^2	4.886×10^2	2.971×10^3	8.808×10^3	5.977×10^3	4.868 × 10²	
	50	Avg	1.354×10^3	6.137×10^3	4.963×10^2	5.424×10^2	4.995×10^2	4.755×10^2	4.239×10^2	1.134×10^3	3.824×10^3	3.030×10^3	4.704×10^2	
	100	Avg	3.845×10^4	2.608×10^4	8.767×10^2	4.097×10^3	6.676×10^2	5.907×10^2	5.493×10^2	9.176×10^3	2.582×10^4	1.879×10^4	5.550×10^2	
	30	Avg	3.845×10^4	2.805×10^4	1.870×10^3	6.742×10^3	8.615×10^2	6.726×10^2	5.980×10^2	2.116×10^4	6.019×10^4	3.485×10^4	5.772×10^2	
	50	Avg	8.120×10^2	8.777×10^2	5.855×10^2	6.894×10^2	8.044×10^2	6.278×10^2	6.744×10^2	7.877×10^2	7.905×10^2	8.721×10^2	5.499 × 10²	
	100	Avg	7.832×10^3	2.475×10^5	6.280×10^2	7.242×10^2	5.707×10^2	6.104×10^2	7.471×10^2	7.217×10^2	8.041×10^2	5.308×10^2		
F ₃	50	Avg	1.167×10^3	1.273×10^3	6.892×10^2	8.934×10^2	9.209×10^2	8.152×10^2	8.940×10^2	1.037×10^3	1.078×10^3	1.118×10^3	6.348 × 10²	
		Min	1.142×10^3	1.184×10^3	6.379×10^2	7.731×10^2	8.081×10^2	7.287×10^2	7.743×10^2	9.853×10^2	9.951×10^2	1.053×10^3	5.836×10^2	
	100	Avg	2.180×10^3	2.374×10^3	1.058×10^3	1.666×10^3	1.413×10^3	1.456×10^3	1.726×10^3	1.789×10^3	1.955×10^3	1.985×10^3	8.613 × 10²	
	30	Avg	6.582×10^2	6.734×10^2	6.043×10^2	6.267×10^2	6.671×10^2	6.038×10^2	6.225×10^2	6.604×10^2	6.655×10^2	6.830×10^2	6.000 × 10²	
	50	Avg	6.468×10^2	6.612×10^2	6.011×10^2	6.144×10^2	6.410×10^2	6.017×10^2	6.095×10^2	6.386×10^2	6.476×10^2	6.615×10^2	6.000×10^2	
	100	Avg	7.210×10^2	7.176×10^2	6.275×10^2	6.648×10^2	6.768×10^2	6.398×10^2	6.664×10^2	6.860×10^2	7.028×10^2	7.030×10^2	6.000 × 10²	
F ₅	30	Avg	1.728×10^3	1.965×10^3	8.418×10^2	1.011×10^3	1.238×10^3	8.735×10^2	8.985×10^2	1.190×10^3	1.295×10^3	1.349×10^3	7.964 × 10²	
		Min	1.593×10^3	1.798×10^3	7.801×10^2	8.671×10^2	1.089×10^3	8.438×10^2	8.402×10^2	1.063×10^3	1.154×10^3	1.175×10^3	7.595×10^2	
	50	Avg	3.572×10^3	3.656×10^3	1.015×10^3	1.701×10^3	1.684×10^3	1.092×10^3	1.141×10^3	1.663×10^3	1.860×10^3	1.919×10^3	9.208 × 10²	
	100	Avg	9.035×10^3	8.338×10^3	1.710×10^3	4.169×10^3	3.250×10^3	1.736×10^3	1.988×10^3	3.320×10^3	3.712×10^3	3.891×10^3	1.356 × 10³	
	30	Avg	1.074×10^3	1.116×10^3	8.713×10^2	9.790×10^2	1.000×10^3	9.379×10^2	9.841×10^2	1.032×10^3	1.041×10^3	1.096×10^3	8.574 × 10²	
	50	Avg	1.427×10^3	1.573×10^3	9.792×10^2	1.229×10^3	1.249×10^3	1.119×10^3	1.213×10^3	1.305×10^3	1.426×10^3	1.404×10^3	9.201 × 10²	
	100	Avg	2.448×10^3	2.751×10^3	1.397×10^3	1.968×10^3	1.897×10^3	1.740×10^3	2.026×10^3	2.157×10^3	2.404×10^3	2.422×10^3	1.160 × 10³	
F ₈	30	Avg	1.079×10^4	1.239×10^4	1.384×10^3	6.278×10^3	7.233×10^3	1.015×10^3	8.747×10^3	6.612×10^3	5.570×10^3	9.591×10^3	9.882 × 10²	
		Min	9.017×10^3	8.805×10^3	1.025×10^3	4.471×10^3	4.425×10^3	9.056×10^2	5.118×10^3	4.627×10^3	4.101×10^3	7.754×10^3	9.065×10^2	
	50	Avg	3.302×10^4	4.335×10^4	4.571×10^3	1.644×10^4	1.783×10^4	1.306×10^3	9.481×10^2	2.195×10^4	2.591×10^4	2.277×10^4	3.066×10^4	1.305 × 10³
	100	Avg	1.197×10^5	1.243×10^5	2.638×10^4	4.507×10^4	3.820×10^4	1.421×10^4	5.208×10^4	6.813×10^4	5.374×10^4	6.959×10^4	7.081 × 10³	
	30	Avg	6.721×10^3	8.149×10^3	3.909×10^3	5.130×10^3	6.156×10^3	4.422×10^3	4.808×10^3	8.037×10^3	6.487×10^3	8.363×10^3	2.745 × 10³	
	50	Avg	1.426×10^4	1.446×10^4	6.428×10^3	8.566×10^3	9.478×10^3	7.081×10^3	7.956×10^3	1.419×10^4	1.223×10^4	1.368×10^4	4.799 × 10³	
	100	Avg	3.040×10^4	3.121×10^4	1.497×10^4	1.728×10^4	2.012×10^4	1.758×10^4	1.618×10^4	3.141×10^4	2.783×10^4	3.071×10^4	1.168 × 10⁴	
F ₁₀	30	Avg	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	9/0/0	
		Min	1.337×10^4	1.342×10^4	4.582×10^3	6.288×10^3	6.969×10^3	6.045×10^3	6.204×10^3	1.301×10^4	1.058×10^4	1.234×10^4	3.734×10^3	
	50	Avg	3.136×10^4	3.121×10^4	1.497×10^4	1.728×10^4	2.012×10^4	1.758×10^4	1.618×10^4	3.035×10^4	2.582×10^4	2.843×10^4	9.083×10^3	
	100	Avg	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$1/0/8$	$0/0/9$	$0/0/9$	8/0/1	
Ranking	50	WITIL	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$2/0/7$	$0/0/9$	$0/0/9$	7/0/2	
	100	WITIL	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$0/0/9$	$1/0/8$	$0/0/9$	$0/0/9$	8/0/1	

Table 3. Comparison of optimization results obtained from hybrid test functions.

F	D	Metrics	SA (1983)	CGA (2000)	GWO (2014)	MFO (2015)	WOA (2016)	LMFO (2016)	WCMFO (2019)	ChOA (2020)	AOA (2021)	SMFO (2021)	I-MFO
F ₁₁	30	Avg	3.124×10^3	5.262×10^3	1.406×10^3	3.749×10^3	1.462×10^3	1.292×10^3	1.336×10^3	3.361×10^3	3.325×10^3	5.265×10^3	1.188×10^3
		Min	2.512×10^3	3.284×10^3	1.271×10^3	1.363×10^3	1.282×10^3	1.177×10^3	1.254×10^3	1.731×10^3	1.739×10^3	2.547×10^3	1.119×10^3
	50	Avg	1.128×10^4	1.978×10^4	3.078×10^3	7.297×10^3	1.591×10^3	1.532×10^3	1.491×10^3	8.609×10^3	1.605×10^4	1.426×10^4	1.326×10^3
		Min	9.143×10^3	1.250×10^4	1.480×10^3	1.574×10^3	1.421×10^3	1.380×10^3	1.344×10^3	6.220×10^3	9.287×10^3	8.985×10^3	1.212×10^3
F ₁₂	100	Avg	1.883×10^5	2.174×10^5	3.531×10^4	1.257×10^5	7.762×10^3	3.319×10^3	2.191×10^3	7.211×10^4	1.625×10^5	2.086×10^5	1.776×10^3
		Min	1.465×10^5	1.629×10^5	1.647×10^4	2.137×10^4	4.463×10^3	2.955×10^3	1.840×10^3	6.100×10^4	1.167×10^5	1.399×10^5	1.464×10^3
	30	Avg	7.895×10^8	3.540×10^9	3.900×10^7	6.158×10^7	3.770×10^7	5.460×10^6	1.254×10^6	3.360×10^9	7.828×10^9	4.462×10^9	2.499×10^5
		Min	3.859×10^8	1.934×10^9	2.109×10^6	7.305×10^4	2.509×10^6	1.046×10^6	3.718×10^4	6.620×10^8	3.034×10^9	1.749×10^9	5.318×10^4
F ₁₃	50	Avg	9.643×10^9	2.939×10^{10}	4.764×10^8	2.475×10^9	1.861×10^8	4.882×10^7	7.229×10^6	1.887×10^{10}	5.350×10^{10}	3.075×10^{10}	1.599×10^6
		Min	5.659×10^9	2.161×10^{10}	7.558×10^7	1.646×10^7	5.114×10^7	2.169×10^7	1.549×10^6	1.045×10^{10}	2.948×10^{10}	1.600×10^{10}	3.874×10^5
	100	Avg	6.862×10^{10}	1.398×10^{11}	4.919×10^9	3.523×10^{10}	6.875×10^8	3.370×10^8	3.428×10^7	6.764×10^{10}	1.766×10^{11}	9.544×10^{10}	3.131×10^6
		Min	5.696×10^{10}	1.132×10^{11}	1.450×10^9	1.435×10^{10}	2.918×10^8	2.296×10^8	3.806×10^6	4.928×10^{10}	1.296×10^{11}	5.096×10^{10}	1.250×10^6
F ₁₄	30	Avg	9.518×10^7	8.620×10^8	8.368×10^5	7.958×10^6	1.463×10^5	4.494×10^5	1.047×10^5	8.863×10^8	4.348×10^4	8.738×10^8	1.994×10^4
		Min	4.048×10^7	3.361×10^8	1.991×10^4	1.122×10^4	2.283×10^4	2.705×10^5	1.436×10^4	3.327×10^7	2.158×10^4	2.189×10^8	1.396×10^3
	50	Avg	1.722×10^9	8.709×10^9	1.532×10^8	2.427×10^8	1.657×10^5	2.510×10^6	8.895×10^4	5.260×10^9	3.917×10^9	1.288×10^{10}	1.434×10^4
		Min	1.103×10^9	5.824×10^9	1.312×10^5	1.136×10^5	4.764×10^4	1.415×10^6	2.174×10^4	5.002×10^8	1.041×10^7	1.435×10^9	1.582×10^3
F ₁₅	100	Avg	8.436×10^9	2.634×10^{10}	4.163×10^8	4.053×10^9	8.423×10^4	1.168×10^7	1.378×10^5	1.915×10^{10}	3.479×10^{10}	1.965×10^{10}	1.105×10^4
		Min	6.140×10^9	1.992×10^{10}	1.579×10^6	2.629×10^8	3.701×10^4	9.896×10^6	3.658×10^4	1.137×10^{10}	2.155×10^{10}	1.073×10^{10}	1.651×10^3
	30	Avg	1.022×10^5	3.619×10^5	1.438×10^5	8.969×10^4	9.075×10^5	2.614×10^4	2.073×10^4	3.244×10^5	4.223×10^4	1.548×10^6	1.671×10^4
		Min	3.932×10^4	7.165×10^4	3.679×10^3	2.197×10^3	1.364×10^3	2.724×10^3	6.252×10^3	4.503×10^4	2.213×10^3	7.879×10^4	5.615×10^3
F ₁₆	50	Avg	1.285×10^6	5.160×10^6	4.016×10^5	3.086×10^5	6.358×10^5	1.086×10^5	8.151×10^4	1.206×10^6	3.933×10^5	2.634×10^7	8.426×10^4
		Min	8.972×10^5	2.700×10^6	4.749×10^4	1.071×10^4	9.639×10^4	2.360×10^4	1.194×10^4	5.706×10^5	4.727×10^4	8.185×10^5	8.798×10^3
	100	Avg	2.710×10^7	6.024×10^7	3.480×10^6	7.558×10^6	1.876×10^6	1.207×10^6	3.627×10^5	7.928×10^6	2.267×10^7	3.200×10^7	3.439×10^5
		Min	2.008×10^7	3.524×10^7	1.056×10^6	3.097×10^5	6.461×10^5	1.938×10^5	1.387×10^5	4.302×10^6	4.241×10^6	7.455×10^6	1.557×10^5
F ₁₇	30	Avg	2.966×10^6	4.746×10^7	3.637×10^5	3.412×10^4	8.683×10^4	9.006×10^4	3.448×10^4	5.434×10^6	2.498×10^4	4.469×10^7	1.983×10^4
		Min	1.113×10^6	6.971×10^6	1.847×10^4	3.640×10^3	1.368×10^4	5.002×10^4	2.547×10^3	1.019×10^6	1.454×10^4	2.375×10^6	2.006×10^3
	50	Avg	1.739×10^8	1.276×10^9	9.314×10^6	2.145×10^7	7.839×10^4	5.206×10^5	7.164×10^4	1.070×10^8	3.131×10^4	8.129×10^8	9.247×10^3
		Min	6.074×10^7	5.105×10^8	1.565×10^4	4.235×10^4	2.225×10^4	3.594×10^5	1.422×10^4	5.379×10^7	1.979×10^4	1.237×10^8	1.622×10^3
F ₁₈	100	Avg	2.250×10^9	8.426×10^9	9.478×10^7	1.045×10^9	2.527×10^5	2.824×10^6	9.337×10^4	4.851×10^9	4.659×10^9	8.332×10^9	7.383×10^3
		Min	1.743×10^9	6.068×10^9	5.864×10^5	1.058×10^5	2.549×10^4	1.949×10^4	1.223×10^4	1.096×10^9	1.070×10^9	1.272×10^9	1.752×10^3
	30	Avg	3.496×10^3	4.179×10^3	2.287×10^3	2.995×10^3	3.519×10^3	2.640×10^3	2.807×10^3	3.475×10^3	3.676×10^3	4.402×10^3	1.928×10^3
		Min	3.252×10^3	3.709×10^3	1.744×10^3	2.487×10^3	2.728×10^3	2.110×10^3	2.095×10^3	2.940×10^3	2.867×10^3	3.607×10^3	1.617×10^3
F ₁₉	50	Avg	5.575×10^3	6.744×10^3	2.791×10^3	4.150×10^3	4.689×10^3	3.621×10^3	3.778×10^3	5.240×10^3	6.261×10^3	6.930×10^3	2.546×10^3
		Min	5.216×10^3	6.027×10^3	2.209×10^3	3.133×10^3	3.895×10^3	2.949×10^3	3.014×10^3	4.488×10^3	3.693×10^3	5.302×10^3	2.186×10^3
	100	Avg	1.236×10^4	1.660×10^4	5.610×10^3	8.085×10^3	9.811×10^3	6.439×10^3	6.869×10^3	1.231×10^4	1.814×10^4	1.679×10^4	4.533×10^3
		Min	1.111×10^4	1.563×10^4	4.748×10^3	6.389×10^3	7.512×10^3	5.301×10^3	4.978×10^3	1.047×10^4	1.301×10^4	1.394×10^4	3.471×10^3
F ₂₀	30	Avg	2.207×10^3	9.064×10^7	2.913×10^5	4.071×10^6	2.647×10^6	6.193×10^4	3.223×10^4	4.950×10^7	1.071×10^6	1.072×10^8	2.012×10^4
		Min	6.005×10^6	3.289×10^7	9.466×10^3	2.093×10^3	1.744×10^3	1.764×10^4	2.168×10^3	2.507×10^6	8.696×10^5	5.192×10^6	1.940×10^3
	50	Avg	9.348×10^7	6.178×10^8	2.362×10^6	6.151×10^6	2.457×10^6	2.445×10^5	2.361×10^4	3.031×10^8	4.614×10^5	9.214×10^8	$$

Table 4. Comparison of optimization results obtained from composition test functions.

F	D	Metrics	SA (1983)	CGA (2000)	GWO (2014)	MFO (2015)	WOA (2016)	LMFO (2016)	WCMFO (2019)	ChOA (2020)	AOA (2021)	SMFO (2021)	I-MFO
F ₂₁	30	Avg	2.593×10^3	2.655×10^3	2.383×10^3	2.476×10^3	2.558×10^3	2.439×10^3	2.493×10^3	2.565×10^3	2.604×10^3	2.653×10^3	2.363×10^3
		Min	2.565×10^3	2.626×10^3	2.352×10^3	2.421×10^3	2.463×10^3	2.378×10^3	2.398×10^3	2.503×10^3	2.515×10^3	2.551×10^3	2.334×10^3
	50	Avg	2.945×10^3	3.065×10^3	2.485×10^3	2.694×10^3	2.888×10^3	2.609×10^3	2.694×10^3	2.886×10^3	3.002×10^3	3.064×10^3	2.430×10^3
		Min	2.893×10^3	3.026×10^3	2.440×10^3	2.575×10^3	2.744×10^3	2.542×10^3	2.580×10^3	2.819×10^3	2.885×10^3	2.935×10^3	2.399×10^3
	100	Avg	4.058×10^3	4.393×10^3	2.845×10^3	3.594×10^3	3.884×10^3	3.280×10^3	3.539×10^3	4.044×10^3	4.581×10^3	4.394×10^3	2.738×10^3
		Min	3.956×10^3	4.275×10^3	2.751×10^3	3.262×10^3	3.502×10^3	3.106×10^3	3.233×10^3	3.804×10^3	4.161×10^3	4.128×10^3	2.652×10^3
F ₂₂	30	Avg	6.618×10^3	6.787×10^3	4.413×10^3	5.842×10^3	5.949×10^3	5.006×10^3	6.637×10^3	9.124×10^3	7.785×10^3	8.654×10^3	2.889×10^3
		Min	4.837×10^3	5.363×10^3	2.420×10^3	3.150×10^3	2.315×10^3	2.325×10^3	5.330×10^3	8.503×10^3	5.492×10^3	5.677×10^3	2.300×10^3
	50	Avg	1.569×10^4	1.600×10^4	8.634×10^3	1.029×10^4	1.208×10^4	8.858×10^3	1.001×10^4	1.653×10^4	1.468×10^4	1.616×10^4	6.525×10^3
		Min	1.464×10^4	1.489×10^4	7.065×10^3	7.958×10^3	8.721×10^3	7.176×10^3	8.600×10^3	1.554×10^4	1.304×10^4	1.529×10^4	5.551×10^3
	100	Avg	3.336×10^4	3.346×10^4	1.777×10^4	2.032×10^4	2.397×10^4	1.948×10^4	1.943×10^4	3.374×10^4	3.092×10^4	3.277×10^4	1.393×10^4
		Min	3.264×10^4	3.169×10^4	1.413×10^4	1.778×10^4	2.087×10^4	1.791×10^4	1.671×10^4	3.233×10^4	2.790×10^4	3.043×10^4	1.189×10^4
F ₂₃	30	Avg	2.916×10^3	3.149×10^3	2.731×10^3	2.801×10^3	3.032×10^3	2.759×10^3	2.785×10^3	3.011×10^3	3.312×10^3	3.283×10^3	2.700×10^3
		Min	2.819×10^3	3.102×10^3	2.695×10^3	2.762×10^3	2.886×10^3	2.710×10^3	2.721×10^3	2.930×10^3	3.093×10^3	3.027×10^3	2.680×10^3
	50	Avg	3.380×10^3	3.816×10^3	2.907×10^3	3.135×10^3	3.592×10^3	3.027×10^3	3.104×10^3	3.515×10^3	4.310×10^3	3.929×10^3	2.858×10^3
		Min	3.345×10^3	3.644×10^3	2.835×10^3	3.046×10^3	3.377×10^3	2.990×10^3	2.980×10^3	3.373×10^3	3.850×10^3	3.594×10^3	2.820×10^3
	100	Avg	4.322×10^3	5.475×10^3	3.405×10^3	3.716×10^3	4.823×10^3	3.475×10^3	3.545×10^3	4.661×10^3	6.745×10^3	6.024×10^3	3.035×10^3
		Min	4.238×10^3	5.225×10^3	3.289×10^3	3.547×10^3	4.263×10^3	3.366×10^3	3.306×10^3	4.424×10^3	6.011×10^3	5.104×10^3	2.971×10^3
F ₂₄	30	Avg	3.084×10^3	3.342×10^3	2.904×10^3	2.974×10^3	3.167×10^3	2.927×10^3	2.978×10^3	3.201×10^3	3.682×10^3	3.433×10^3	2.871×10^3
		Min	3.063×10^3	3.270×10^3	2.855×10^3	2.910×10^3	3.021×10^3	2.897×10^3	2.928×10^3	3.128×10^3	3.473×10^3	3.217×10^3	2.852×10^3
	50	Avg	3.459×10^3	4.039×10^3	3.087×10^3	3.227×10^3	3.733×10^3	3.136×10^3	3.231×10^3	3.713×10^3	4.749×10^3	4.359×10^3	3.008×10^3
		Min	3.416×10^3	3.856×10^3	3.000×10^3	3.152×10^3	3.545×10^3	3.071×10^3	3.135×10^3	3.588×10^3	4.340×10^3	3.875×10^3	2.964×10^3
	100	Avg	5.059×10^3	8.073×10^3	3.962×10^3	4.272×10^3	5.854×10^3	4.086×10^3	4.293×10^3	5.913×10^3	1.065×10^4	8.875×10^3	3.651×10^3
		Min	4.961×10^3	7.435×10^3	3.819×10^3	4.124×10^3	5.238×10^3	3.976×10^3	4.048×10^3	5.524×10^3	8.928×10^3	6.869×10^3	3.556×10^3
F ₂₅	30	Avg	4.148×10^3	5.436×10^3	2.957×10^3	3.107×10^3	2.945×10^3	2.889×10^3	2.887×10^3	4.099×10^3	4.426×10^3	3.940×10^3	2.888×10^3
		Min	3.828×10^3	4.615×10^3	2.913×10^3	2.889×10^3	2.898×10^3	2.888×10^3	2.884×10^3	3.456×10^3	3.635×10^3	3.463×10^3	2.887×10^3
	50	Avg	1.054×10^4	1.824×10^4	3.371×10^3	4.930×10^3	3.155×10^3	3.043×10^3	3.041×10^3	8.621×10^3	1.388×10^4	1.083×10^4	3.000×10^3
		Min	7.933×10^3	1.397×10^4	3.055×10^3	3.159×10^3	3.039×10^3	2.994×10^3	2.962×10^3	6.928×10^3	1.101×10^4	7.534×10^3	2.978×10^3
	100	Avg	5.160×10^4	5.583×10^4	5.277×10^3	1.123×10^4	3.590×10^3	3.456×10^3	3.321×10^3	1.363×10^4	2.328×10^4	2.002×10^4	3.262×10^3
		Min	4.633×10^4	4.701×10^4	4.686×10^3	4.792×10^3	3.464×10^3	3.365×10^3	3.206×10^3	1.142×10^4	1.986×10^4	1.680×10^4	3.116×10^3
F ₂₆	30	Avg	6.408×10^3	8.876×10^3	4.424×10^3	5.689×10^3	7.599×10^3	5.012×10^3	5.447×10^3	6.328×10^3	9.412×10^3	8.871×10^3	4.300×10^3
		Min	5.542×10^3	7.753×10^3	3.954×10^3	4.921×10^3	5.975×10^3	4.607×10^3	4.955×10^3	5.882×10^3	7.702×10^3	5.057×10^3	2.900×10^3
	50	Avg	1.063×10^4	1.594×10^4	5.735×10^3	8.121×10^3	1.306×10^4	7.041×10^3	8.059×10^3	1.028×10^4	1.546×10^4	1.587×10^4	5.179×10^3
		Min	1.002×10^4	1.454×10^4	5.192×10^3	6.910×10^3	9.977×10^3	6.161×10^3	7.062×10^3	9.047×10^3	1.326×10^4	1.396×10^4	4.512×10^3
	100	Avg	2.452×10^4	4.461×10^4	1.263×10^4	1.741×10^4	3.111×10^4	1.493×10^4	1.752×10^4	2.492×10^4	4.995×10^4	4.315×10^4	9.748×10^3
		Min	2.363×10^4	4.099×10^4	1.124×10^4	1.526×10^4	2.326×10^4	1.333×10^4	1.518×10^4	2.276×10^4	4.219×10^4	3.583×10^4	9.123×10^3
F ₂₇	30	Avg	3.279×10^3	3.667×10^3	3.229×10^3	3.236×10^3	3.346×10^3	3.221×10^3	3.228×10^3	3.493×10^3	4.286×10^3	3.688×10^3	3.213×10^3
		Min	3.250×10^3	3.497×10^3	3.212×10^3	3.208×10^3	3.282×10^3	3.200×10^3	3.201×10^3	3.355×10^3	3.633×10^3	3.397×10^3	3.184×10^3
	50	Avg	3.730×10^3	5.183×10^3	3.471×10^3	3.550×10^3	4.305×10^3	3.356×10^3	3.504×10^3	4.272×10^3	6.565×10^3	5.306×10^3	3.337×10^3
		Min	3.669×10^3	4.697×10^3	3.342×10^3	3.407×10^3	3.678×10^3	3.249×10^3 </					

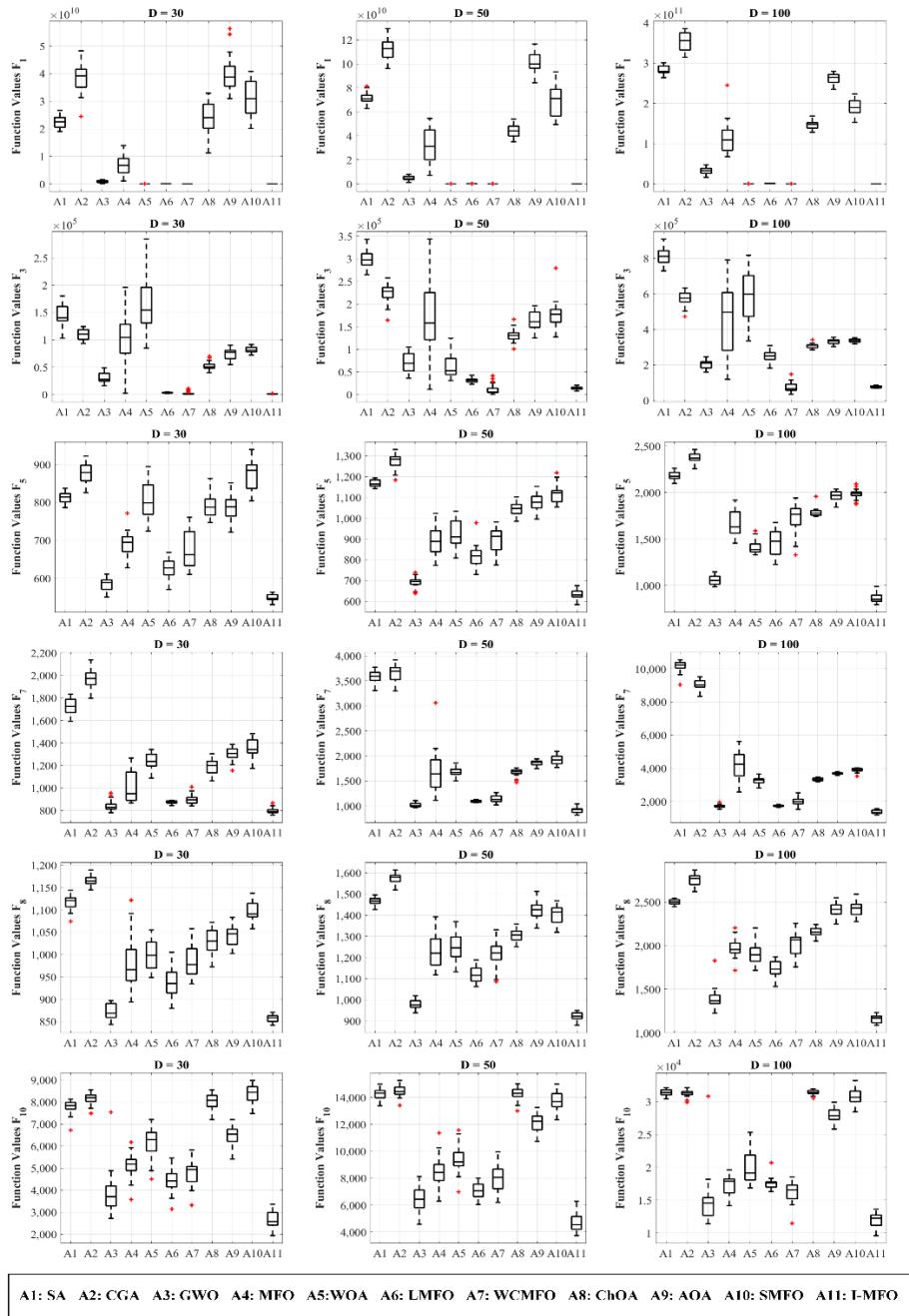


Figure 3. The comparison of fitness distribution on unimodal and multimodal functions.

4.3. Local Optima Avoidance Evaluation

This experimental evaluation is benchmarking the ability of the proposed algorithm against the contender algorithms in terms of local optima avoidance and striking a balance between exploration and exploitation by considering hybrid and composition function results. The obtained results tabulated in Tables 3 and 4 indicate that the proposed I-MFO algorithm is superior to the contender algorithms in dimensions 30, 50, and 100. The main reason is that the AWAS strategy helps trapped moths to escape the local optima and obtain a better position by changing random dimensions of trapped moths with dimensions of the best flame and a random moth's position. The random moth causes the trapped moth to explore the search space and increases the population diversity while considering the best flame enhances the exploitation capabilities of the algorithm simultaneously. Furthermore, Figure 4 visualizes the comparison of fitness distribution using box and whiskers diagrams

in which almost all diagrams demonstrate that the proposed I-MFO can find the best solutions during the optimization process. It verifies that I-MFO can provide satisfactory equilibration between exploration and exploitation.

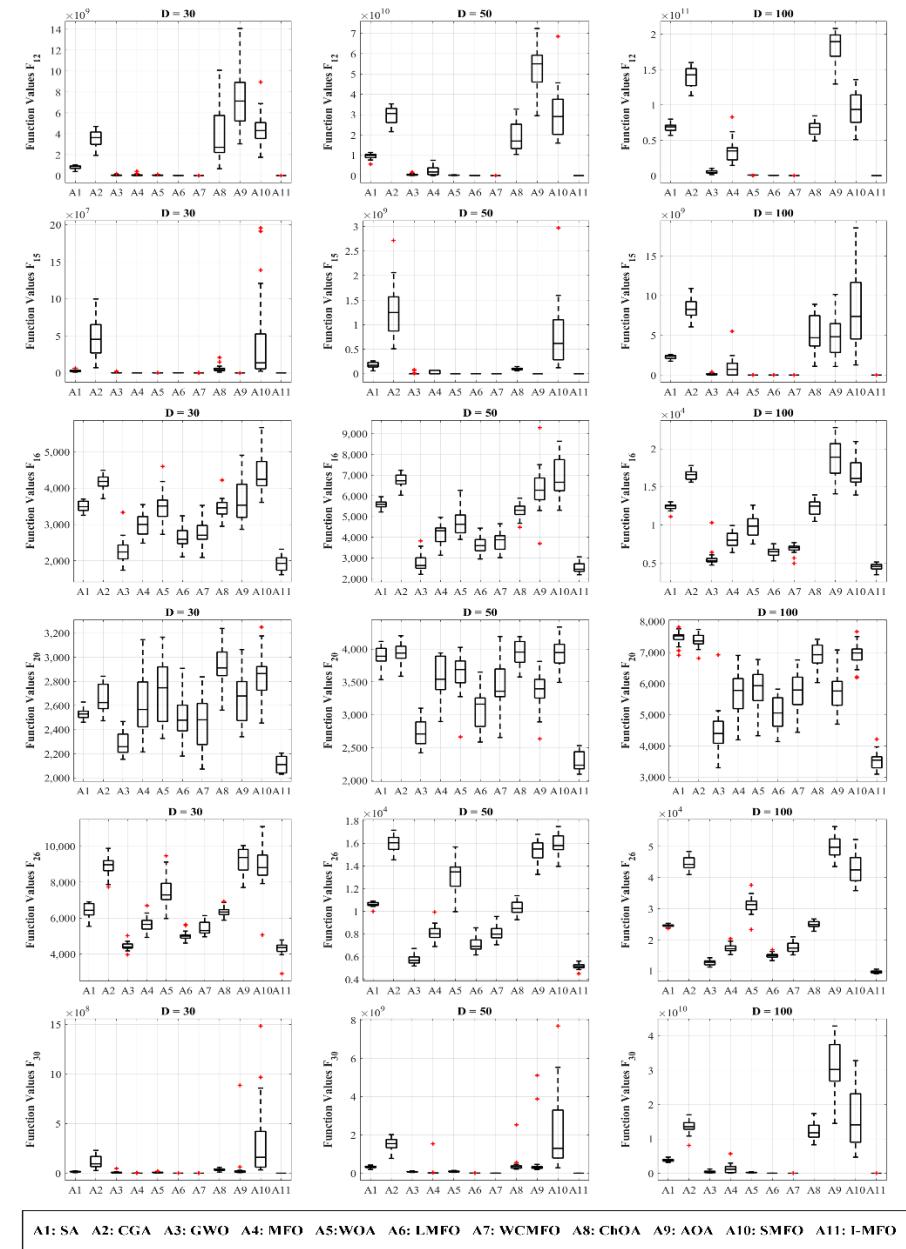


Figure 4. The comparison of fitness distribution on hybrid and composition functions.

4.4. I-MFO Overall Effectiveness

The overall effectiveness (OE) [50] of the I-MFO and other contender algorithms is computed by using results reported in Tables 2–4. The OE results tabulated in Table 5 are calculated using Equation (7), where N indicates the number of test functions and L is the number of losses of each algorithm. The results reveal that I-MFO with overall effectiveness of 92% is the most effective algorithm for all dimensions: 30, 50, and 100.

$$OE (\%) = \left(\frac{N - L}{N} \right) \times 100 \quad (7)$$

Table 5. The overall effectiveness of the I-MFO and contender algorithms.

Algorithms	SA (W/T/L)	CGA (W/T/L)	GWO (W/T/L)	MFO (W/T/L)	WOA (W/T/L)	LMFO (W/T/L)	WCMFO (W/T/L)	ChOA (W/T/L)	AOA (W/T/L)	SMFO (W/T/L)	I-MFO (W/T/L)
D = 30	0/0/29	0/0/29	0/0/29	0/0/29	0/0/29	0/0/29	2/0/27	0/0/29	0/0/29	0/0/29	27/0/2
D = 50	0/0/29	0/0/29	0/0/29	0/0/29	0/0/29	0/0/29	3/0/26	0/0/29	0/0/29	0/0/29	26/0/3
D = 100	0/0/29	0/0/29	0/0/29	0/0/29	0/0/29	0/0/29	2/0/27	0/0/29	0/0/29	0/0/29	27/0/2
Total	0/0/87	0/0/87	0/0/87	0/0/87	0/0/87	0/0/87	7/0/80	0/0/87	0/0/87	0/0/87	80/0/7
OE	0%	0%	0%	0%	0%	0%	8%	0%	0%	0%	92%

4.5. Convergence Behavior Analysis

In this section, the convergence behavior of I-MFO is assessed and compared with contender algorithms on some selected functions with dimensions 30, 50, and 100. The convergence curves of the best fitness values obtained by each algorithm on unimodal and multimodal test functions are plotted in Figure 5. Moreover, the convergence curves of hybrid and composition test functions are plotted in Figure 6.

Investigating convergence behaviors of the I-MFO reveals that it shows various convergence behaviors. The most common behavior is an accelerated descent with the fastest accurate solutions toward the promising area in the early iterations, which can be seen in 30D ($F_5, F_{15}, F_{16}, F_{26}$), 50D ($F_6, F_{15}, F_{16}, F_{26}, F_{30}$), and 100D ($F_5, F_7, F_8, F_{15}, F_{16}, F_{26}$). For some functions such as 30D ($F_1, F_7, F_8, F_{18}, F_{30}$), 50D (F_1, F_8, F_{18}), and 100D ($F_1, F_{10}, F_{18}, F_{30}$), the I-MFO shows abrupt movements in the first half of iterations and very low variations for the second half, which proves the efficient balance between exploration and exploitation. Finally, for 30D ($F_3, F_7, F_{10}, F_{12}, F_{20}$), 50D ($F_3, F_5, F_8, F_{10}, F_{12}, F_{18}, F_{20}$), and 100D ($F_3, F_{12}, F_{18}, F_{20}$), the I-MFO starts its convergence with a steep descent slope and then changes to a gradual trend toward the optimum solutions until final iterations. This behavior demonstrates the ability of the I-MFO in escaping from the local optimum and taking advantage of the last iterations.

4.6. Population Diversity Analysis

In metaheuristic algorithms, the population diversity maintenance is important throughout the optimization process. The low diversity among search agents may cause the algorithm to plunge into the local optimum. In this experiment, the population diversity of the proposed I-MFO and contender algorithms is measured by a moment of inertia (I_c) [119], where the I_c is the spreading of each individual from their mass center given by Equation (8) and the mass center c_j for $j = 1, 2 \dots D$ is calculated by Equation (9).

$$I_c = \sum_{i=1}^N \sum_{j=1}^D (M_{ij} - c_j)^2 \quad (8)$$

$$c_j = \frac{1}{D} \sum_{i=1}^N M_{ij} \quad (9)$$

The presented population diversity measures the distribution of search agents, and the diversity's changing slope for the proposed algorithm and contender algorithms is plotted in Figure 7. This experiment is conducted on some CEC 2018 benchmark functions with dimensions 30, 50, and 100. Comparing the convergence curves in Figures 5 and 6 and the plotted diversity in Figure 8 reveals that I-MFO can effectively maintain diversification among solutions until the near-optimal solution is met.

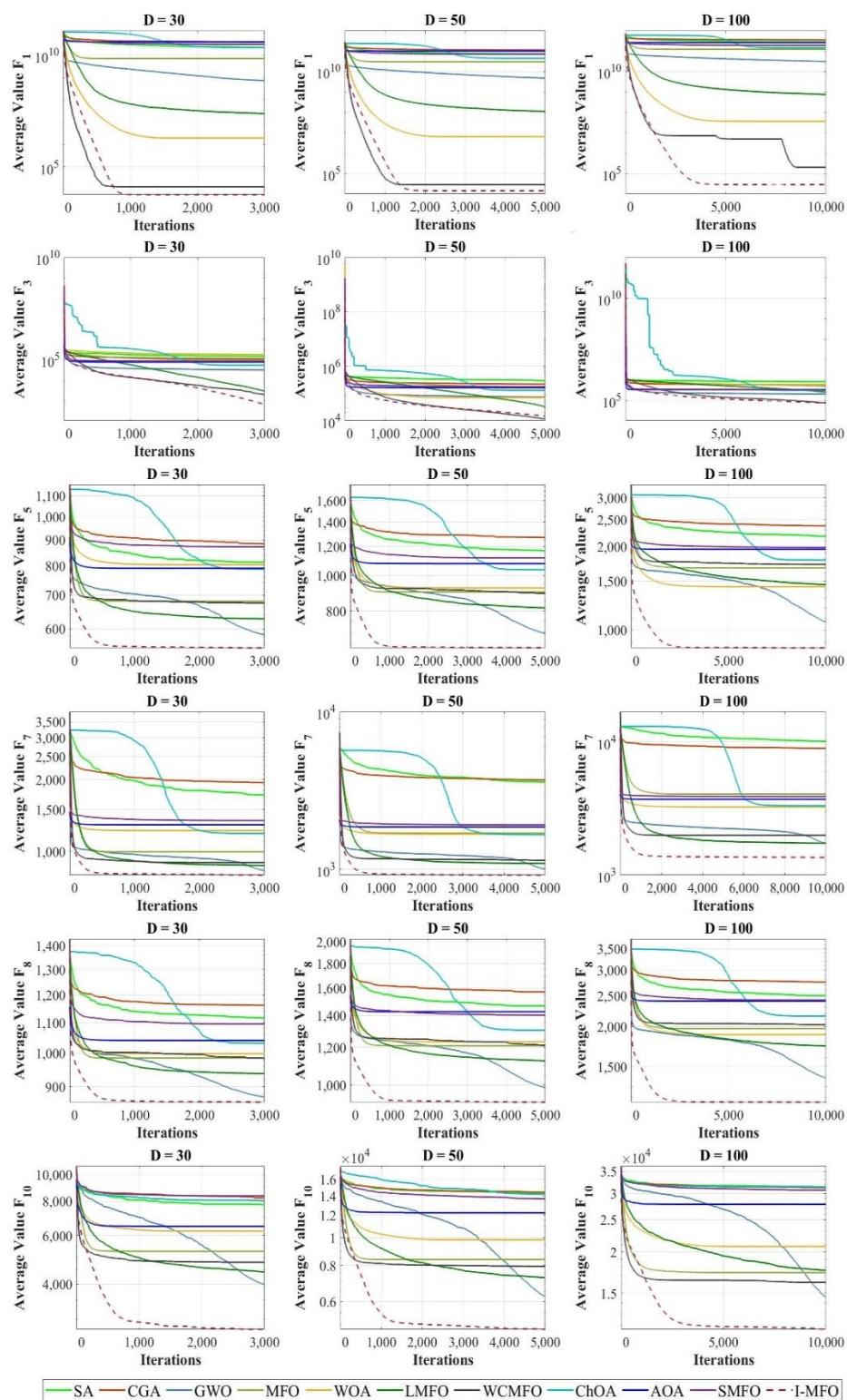


Figure 5. The convergence curves of algorithms in unimodal and multimodal test functions.

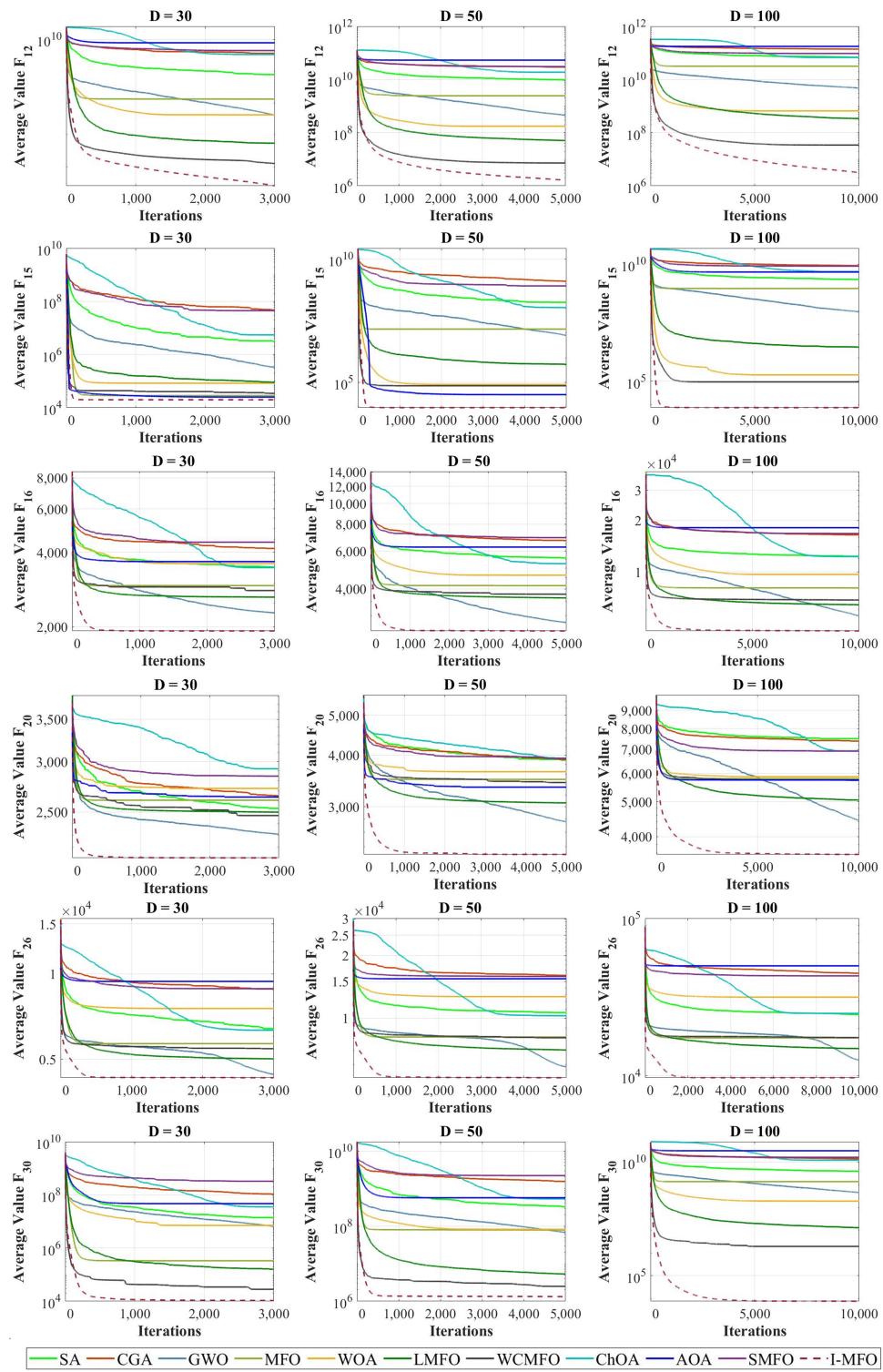


Figure 6. The convergence curves of algorithms in hybrid and composition test functions.

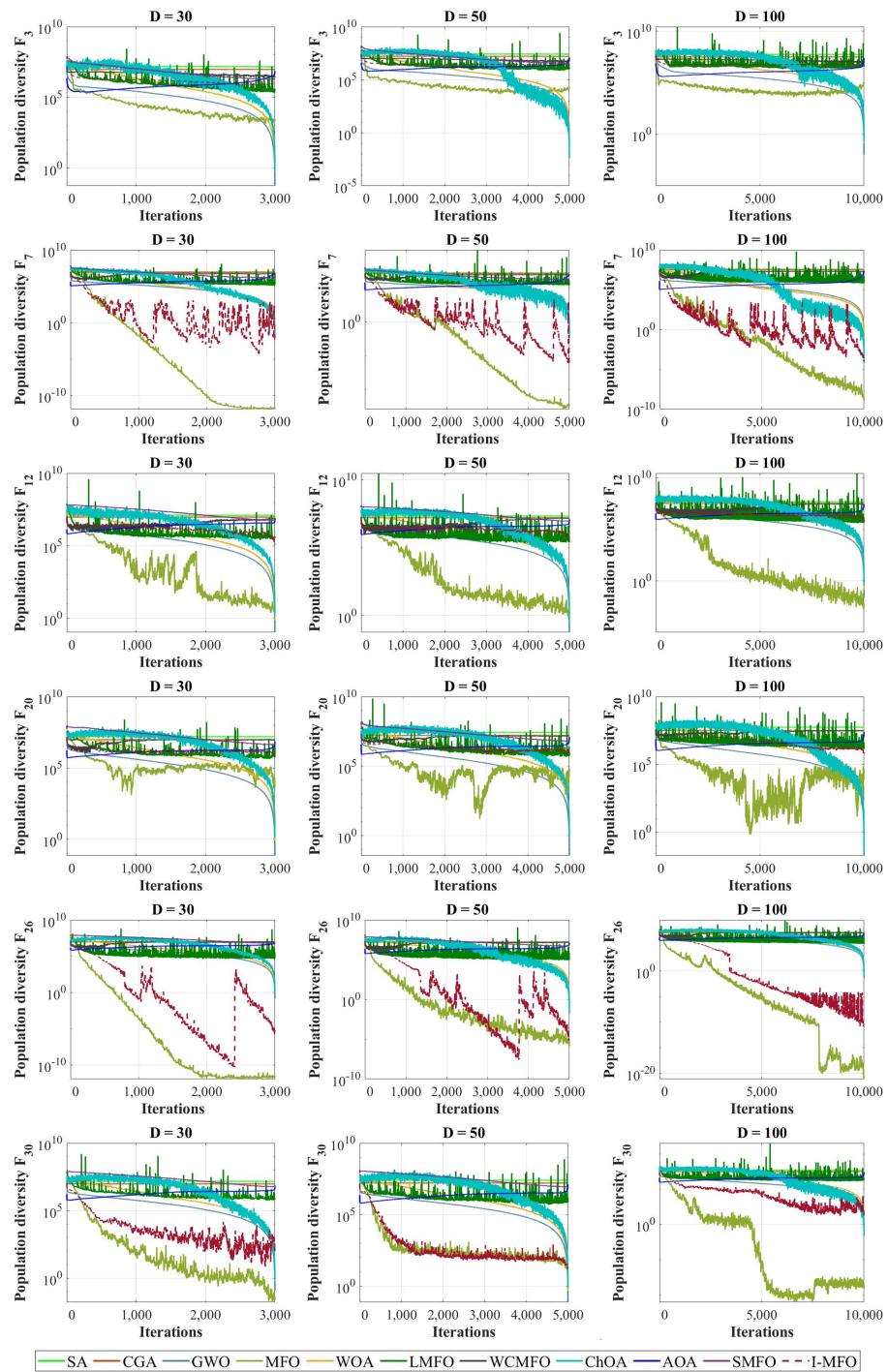


Figure 7. The population diversity of algorithms in different test functions.

4.7. Sensitivity Analysis on the Number of Flight (NF) Parameter

As discussed in Definition 2, the NF parameter is the number of opportunities for each trapped moth to fly in the search space and possibly obtain a better position. Hence, in this experiment, the impact of considering different values for the NF parameter is evaluated and discussed. The plotted curves in Figure 8 illustrate the convergence behavior of the MFO compared with different variants of I-MFO algorithm. In I-MFO-NF1 and I-MFO-NF5, the value of NF is considered by 1 and 5 while in I-MFO, the value of NF is set by a random number in $[1, D]$. The results gained for different dimensions 30, 50, and 100 reveal that setting NF by a random number limited by the dimension has an advantage for trapped moths to possibly escape from the local optima for different test functions.

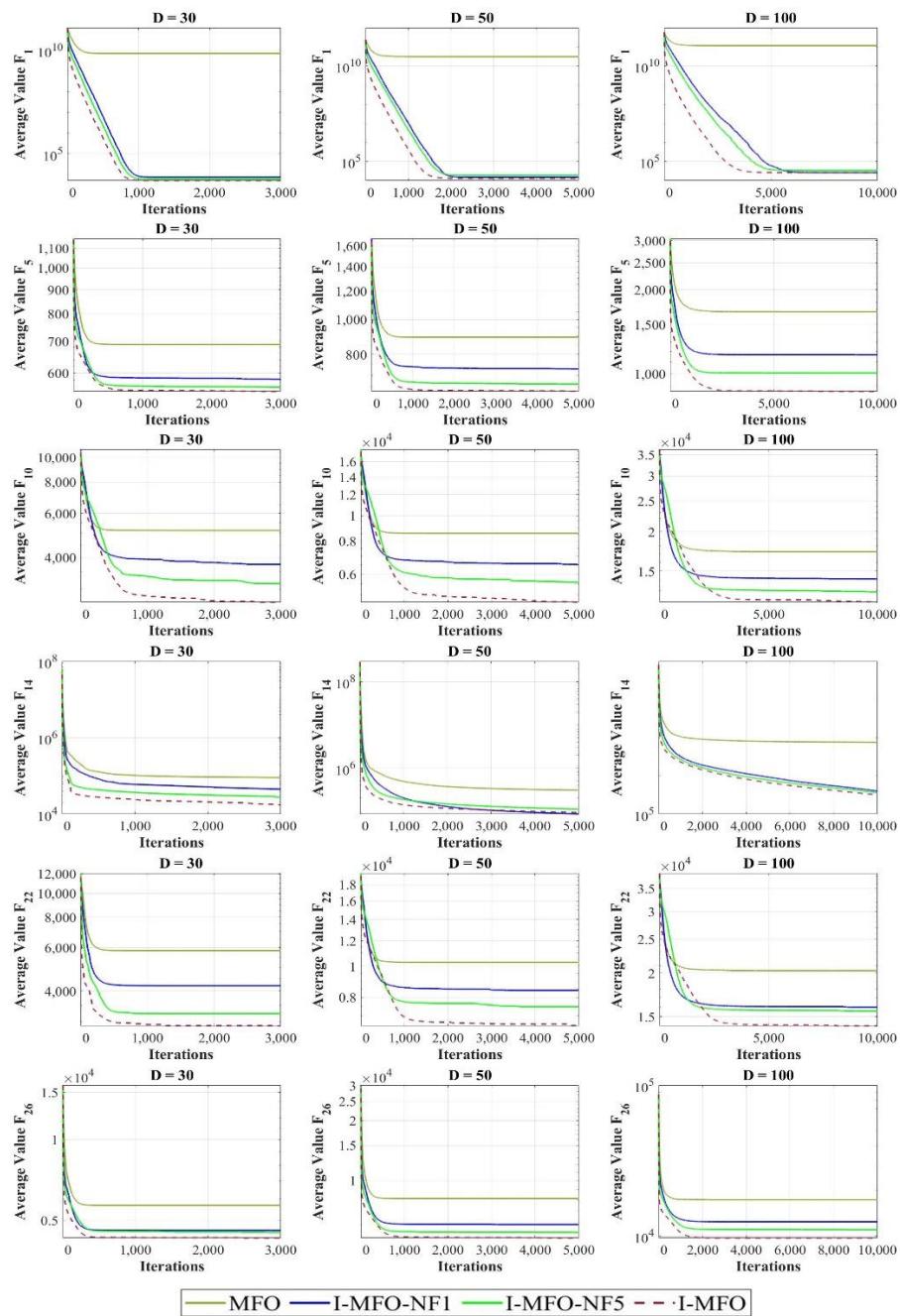


Figure 8. The sensitivity analysis on the *NF* parameter.

4.8. Impact Analysis of Applying AWAS Strategy

In this experiment, the impact of applying the AWAS strategy is analyzed on some selected functions of the CEC 2018 benchmark for different dimensions 30, 50, and 100. The proposed AWAS strategy can ameliorate the MFO's weaknesses described in Section 2. To adequately assess the impact of applying the AWAS strategy, in this experiment we consider MFO, I-MFO, and its three variations including I-MFO-10%, I-MFO-40%, and I-MFO-80% which indicate the percentage of trapped moths that are randomly selected to possibly escape from the local optima using the proposed AWAS strategy.

The first row of Figure 9 indicates convergence curves for unimodal F_1 , where the I-MFO and its variations outperform the MFO for all dimensions. Specifically, for dimension 100, the I-MFO-10% offers superior outcomes while it has less computational cost compared to the I-MFO. The curves provided for multimodal F_5 and F_{10} indicate that the I-MFO

offers better solutions, while in the next ranks, I-MFO-80%, I-MFO-40%, and I-MFO-10% outperform the MFO. The hybrid test function is shown in the fourth row of Figure 9, where the I-MFO and its variations keep converging toward the global optimum with a steep slope until the final iterations. The I-MFO and its variations can also find better solutions for the composition functions F_{22} and F_{26} , wherein for these functions, as the number of dimensions grows, the significance of the AWAS strategy in guiding the population toward the global optimum region and avoiding local optima entrapment becomes clearer. Although the provided results demonstrate that the I-MFO with 100% of trapped moths applied to AWAS strategy mostly provides better solutions for different dimensions and search spaces, other variations of the I-MFO can also provide competitive performance while they have the advantage of lower computational cost compared to the I-MFO.

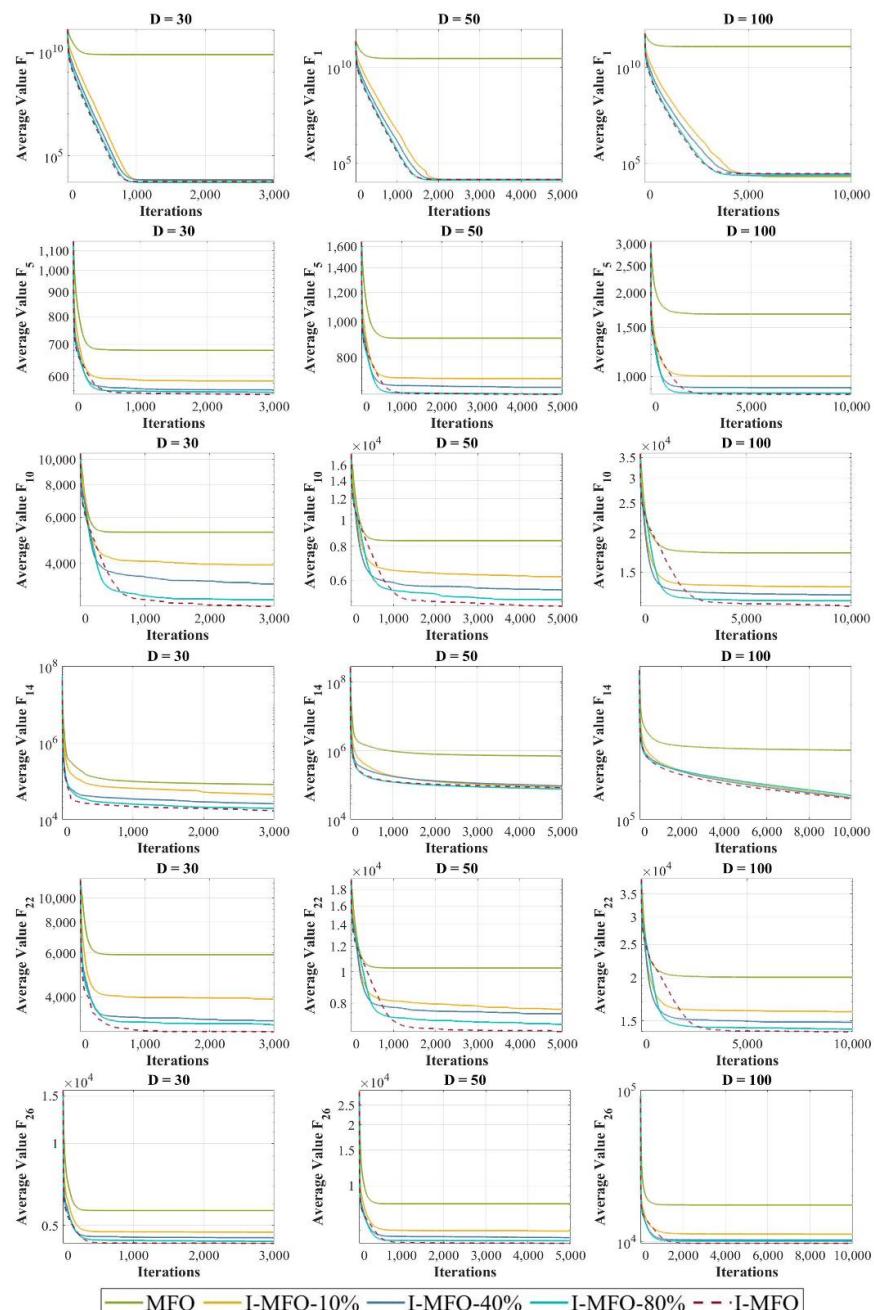


Figure 9. Impact analysis of applying AWAS strategy on different percentages of trapped moths.

5. Statistical Analysis

In this section, the results obtained in the preceding section are first statistically analyzed using the non-parametric Friedman test. The Bonferroni and Tukey post hoc producers are then conducted to establish proper comparisons between the proposed algorithm and comparative algorithms.

5.1. Non-Parametric Friedman Test

The Friedman test is performed to rank the significance of the superiority algorithms statistically [120,121]. The obtained results for unimodal and multimodal test functions are tabulated in Table 6 and the results for hybrid and composition functions are reported in Table 7. This statistical analysis shows that the I-MFO is first rank on all test functions for dimensions of 30, 50, and 100.

Table 6. Friedman test for unimodal and multimodal functions of the CEC 2018.

Functions		Unimodal Functions						Multimodal Functions					
Dimensions		30		50		100		30		50		100	
Algorithms	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	
SA	9.22	10	9.77	10	10.50	11	8.67	9	9.45	10	10.15	10	
CGA	9.75	11	10.42	11	10.25	10	10.49	11	10.74	11	10.29	11	
GWO	4.57	4	5.12	5	4.12	3	2.52	2	2.66	2	2.578	2	
MFO	6.30	6	6.07	6	6.30	7	4.65	5	4.67	5	5.26	6	
WOA	6.70	7	3.60	4	6.10	5	6.21	6	5.45	6	4.53	4	
LMFO	3.45	3	3.57	3	4.15	4	2.99	3	2.85	3	3.29	3	
WCMFO	1.77	2	1.62	2	1.70	2	4.61	4	4.52	4	4.67	5	
ChOA	6.17	5	6.70	7	6.17	6	7.19	7	7.27	7	7.34	7	
AOA	8.12	8	9.02	9	7.80	9	7.67	8	8.04	8	7.98	8	
SMFO	8.57	9	8.57	8	7.57	8	9.75	10	9.14	9	8.82	9	
I-MFO	1.35	1	1.50	1	1.32	1	1.20	1	1.17	1	1.06	1	

Table 7. Friedman test for hybrid and composition functions of the CEC 2018.

Functions		Hybrid Functions						Composition Functions					
Dimensions		30		50		100		30		50		100	
Algorithms	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	Avg. Rank	Overall Rank	
SA	7.96	8	8.74	9	8.52	8	7.20	7	8.74	9	7.68	8	
CGA	9.94	10	10.19	11	10.20	11	9.50	9	10.19	11	9.65	10	
GWO	3.79	2	3.79	3	4.17	5	3.12	2	3.795	3	3.14	2	
MFO	3.87	4	4.56	5	5.4	6	4.36	5	4.56	5	4.95	5	
WOA	6.49	7	5.14	6	4.04	4	6.05	6	5.14	6	5.83	6	
LMFO	4.63	5	4.06	4	3.65	3	3.39	3	4.06	4	3.23	3	
WCMFO	3.86	3	3.57	2	2.95	2	3.99	4	3.57	2	3.55	4	
ChOA	8.18	9	7.74	8	7.48	7	7.70	8	7.74	8	7.4	7	
AOA	5.81	6	6.96	7	8.8	9	9.64	10	6.96	7	10.15	11	
SMFO	10.21	11	10.12	10	9.62	10	9.90	11	10.12	10	9.36	9	
I-MFO	1.23	1	1.10	1	1.09	1	1.12	1	1.10	1	1.03	1	

Figure 10 contains six charts to visually show the ranking of the I-MFO and contender algorithms in different dimensions. The left side illustrates the ranking of algorithms in various functions of the CEC 2018 benchmark, while the right side shows the bar chart of Friedman test average results. The radar graph shows that the I-MFO outperforms other algorithms in different dimensions as the smaller size of the I-MFO indicates its first and second rank for all functions. The bar chart provided on the right side reveals that the I-MFO is superior to other comparative algorithms as it has the shortest bar in various dimensions of 30, 50, and 100.

5.2. Post Hoc Analysis

In the post hoc analysis [120], we evaluated the proposed hypothesis between the control method and the rest of the compared methods in Table 8 by employing Bonferroni and Tukey's multiple comparison producers. In this experiment, the level of significance is $\alpha = 0.05$, which determines whether or not a hypothesis is acceptable by comparing the significant difference (p -value) between each pair of algorithms. Since gained p -values for all dimensions 30, 50, and 100 are less than $\alpha = 0.05$, it reveals that there are significant differences between the performances of the I-MFO and other compared algorithms.

Table 8. Adjusted p -values for the Friedman test on different dimensions (I-MFO is the control method).

Dimensions	30		50		100	
	Algorithms	Bonferroni p -Value	Tukey p -Value	Bonferroni p -Value	Tukey p -Value	Bonferroni p -Value
SA	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}
CGA	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}
GWO	5.337×10^{-7}	5.338×10^{-7}	5.337×10^{-7}	5.338×10^{-7}	7.238×10^{-8}	7.247×10^{-8}
MFO	5.337×10^{-7}	5.338×10^{-7}	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}
WOA	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}
LMFO	3.444×10^{-6}	3.444×10^{-6}	5.337×10^{-7}	5.338×10^{-7}	5.337×10^{-7}	5.338×10^{-7}
WCMFO	1.595×10^{-3}	1.595×10^{-3}	3.444×10^{-6}	3.444×10^{-6}	3.444×10^{-6}	3.444×10^{-6}
ChOA	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}
AOA	5.337×10^{-7}	5.338×10^{-7}	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}
SMFO	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}	7.238×10^{-8}	7.247×10^{-8}

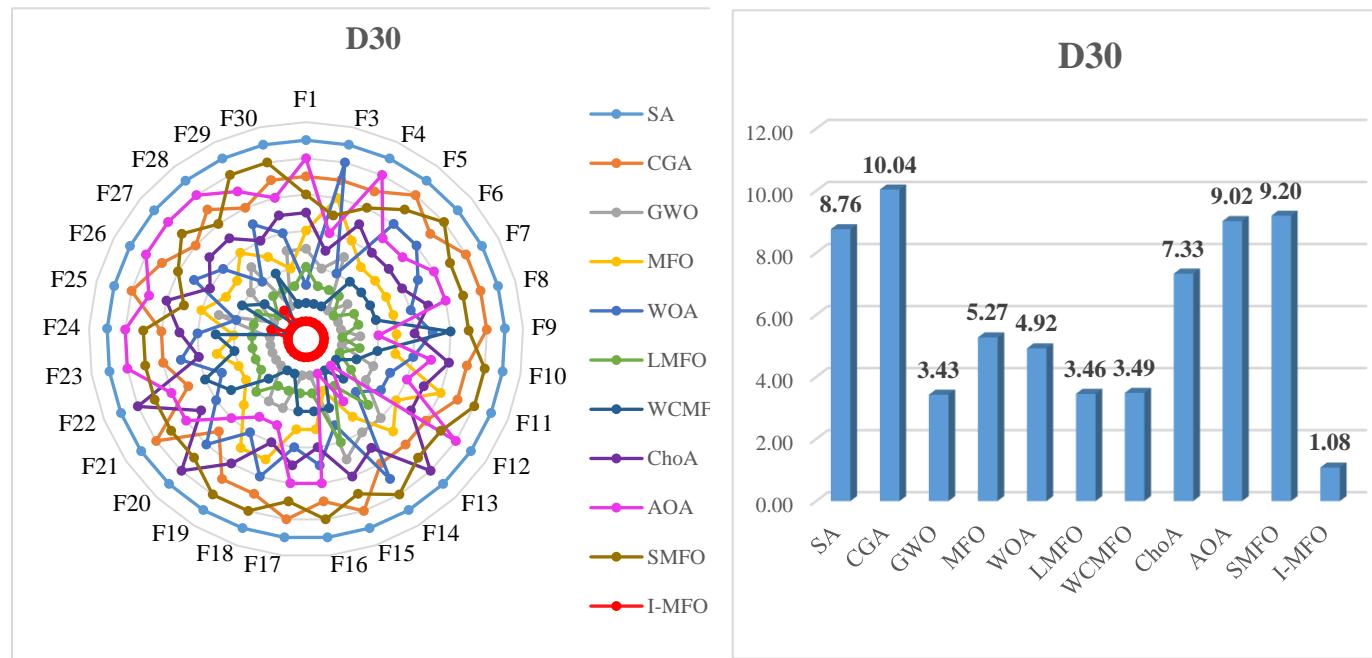


Figure 10. Cont.

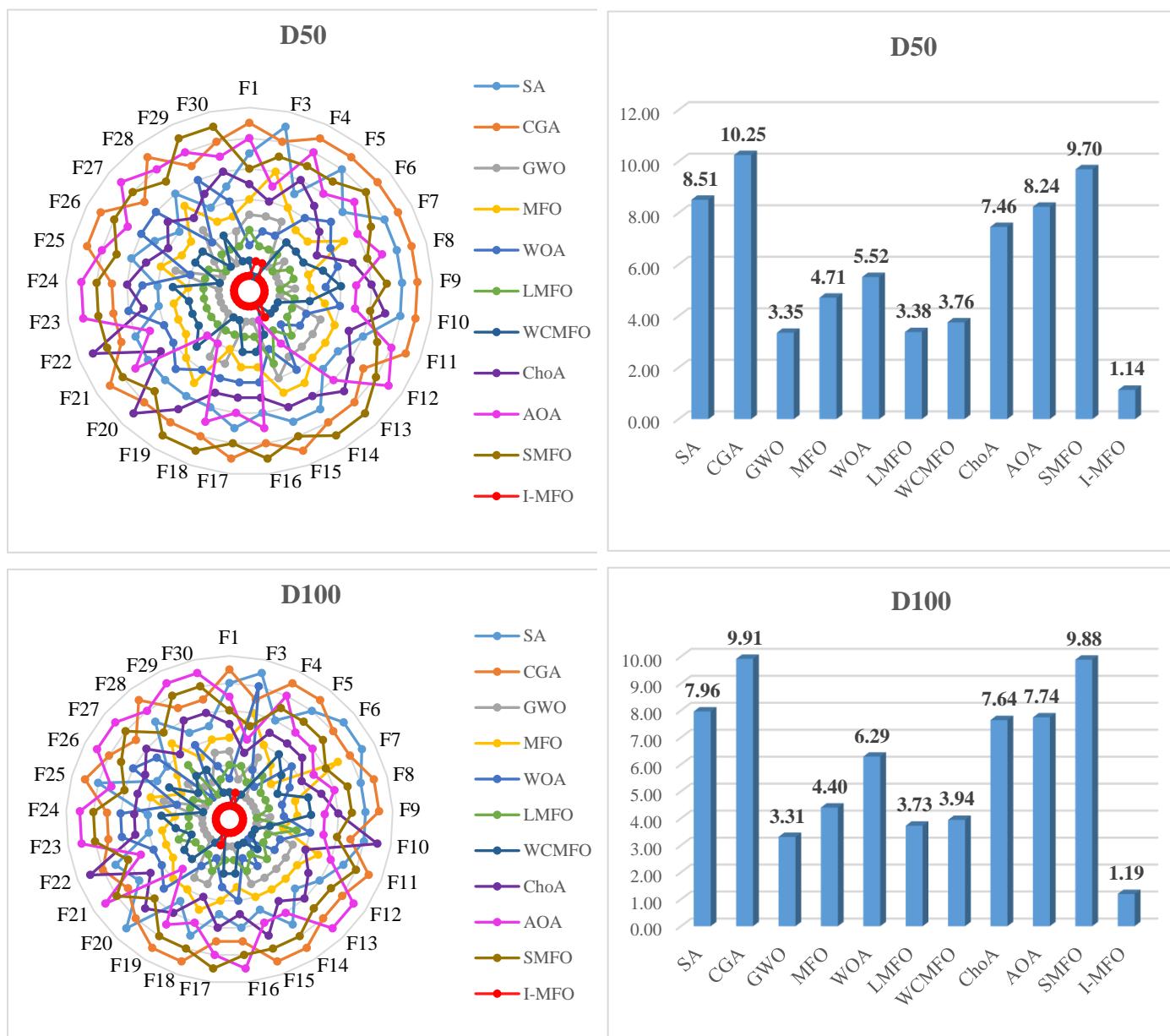


Figure 10. The radar graphs and bar charts of algorithms in different dimensions.

6. Applicability of I-MFO Algorithm to Solve Mechanical Engineering Problems

In this section, three constrained mechanical engineering problems from the latest test-suite CEC 2020 [103] are considered to evaluate the applicability of the I-MFO algorithm in real-world applications. To achieve a fair comparison, the algorithms were run 20 times with the population size (N) 20 and maximum iterations ($MaxIt$) $(D \times 10^4)/N$. In this experimental evaluation, the proposed algorithm and contender algorithms compete to solve three different problems that consist of a gas transmission compressor design problem, three-bar truss, and tension/compression spring design.

P₁: Gas transmission compressor design problem

Minimization of the objective function using four design variables is the main goal of the gas transmission compressor design problem. This problem is illustrated and formulated in Figure 11 and Equation (10). The performance of the proposed algorithm is evaluated against the contender algorithms to solve this problem and the obtained results

are tabulated in Table 9. As shown in this table, the I-MFO is superior in addressing this issue.

$$\begin{aligned} \text{Minimize } f(\bar{x}) &= 8.61 \times 10^5 x_1^{\frac{1}{2}} x_2 x_3^{-\frac{2}{3}} x_4^{-\frac{1}{2}} + (3.69) \times 10^4 x_3 + (7.72) \times 10^8 x_1^{-1} x_2^{0.219} - (765.43) \times 10^6 x_1^{-1} \\ \text{Subject to } x_4 x_2^{-2} + x_2^{-2} - 1 &\leq 0 \\ \text{Variable range } 20 \leq x_1 &\leq 50, 1 \leq x_2 \leq 10, 20 \leq x_3 \leq 45, 0.1 \leq x_4 \leq 60 \end{aligned} \quad (10)$$

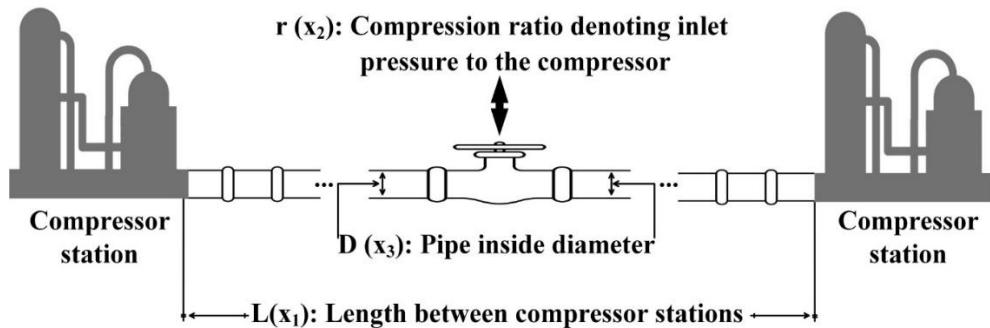


Figure 11. Gas transmission compressor design problem.

Table 9. Results of the gas transmission compressor design problem.

Algorithms	Optimal Values for Variables				Optimal Cost
	x_1	x_2	x_3	x_4	
SA	46.76	1.62	25.79	0.55	4.390311×10^6
CGA	49.97	20.01	31.47	49.83	1.735023×10^7
GWO	20.00	7.81	20.00	60.00	2.964974×10^6
MFO	50.00	1.18	24.57	0.39	2.964902×10^6
WOA	50.00	1.18	24.86	0.39	2.965002×10^6
LMFO	49.46	1.18	24.64	0.39	2.965456×10^6
WCMFO	50.00	1.18	24.61	0.39	2.964897×10^6
ChOA	50.00	1.19	24.24	0.41	2.966828×10^6
AOA	50.00	1.23	20.00	0.51	3.014615×10^6
SMFO	23.66	1.09	23.66	0.19	3.052254×10^6
I-MFO	50.00	1.18	24.60	0.39	2.964896×10^6

P₂: Three-bar truss problem

In this problem, three constraints and two variables are utilized to formulate the objective function, which is the weight of the bar structures. The schematic and formulation of this problem are represented in Figure 12 and Equation (11), respectively. The proposed I-MFO algorithm and comparative algorithms are compared for solving this problem. The attained results from this experiment are tabulated in Table 10, in which the I-MFO algorithm outperforms other algorithms in approximating the optimal values for variables with minimum weight.

$$\begin{aligned} \text{Minimize } f(x) &= l \times \left(x_2 + 2\sqrt{2} x_1 \right) \\ \text{Subject to } g_1(x) &= \frac{x_2}{2x_2 x_1 + \sqrt{2} x_1^2} p - \sigma \leq 0 \\ g_2(x) &= \frac{x_2 + \sqrt{2} x_1}{2x_2 x_1} p - \sigma \leq 0 \\ g_3(x) &= \frac{1}{x_1 + \sqrt{2} x_2} p - \sigma \leq 0 \end{aligned} \quad (11)$$

where $l = 100 \text{ cm}$, $p = \frac{2\text{KN}}{\text{cm}^2}$, and $\sigma = \frac{2\text{KN}}{\text{cm}^2}$

Variable range $0 \leq x_1 \leq 1$

$0 \leq x_2 \leq 1$

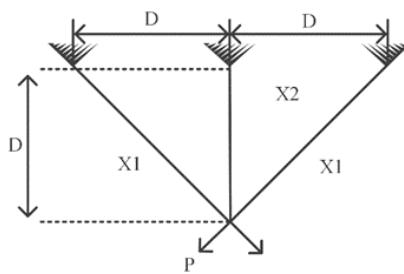


Figure 12. Three-bar truss problem.

Table 10. Results of the three-bar truss problem.

Algorithms	Optimal Values for Variables		Optimal Weight
	x_1	x_2	
SA	0.768630	0.474232	2.6482456×10^2
CGA	0.792428	0.397752	2.6390770×10^2
GWO	0.787771	0.410872	2.6389619×10^2
MFO	0.789186	0.406806	2.6389603×10^2
WOA	0.787713	0.410977	2.6389653×10^2
LMFO	0.791713	0.399909	2.6392114×10^2
WCMFO	0.788472	0.408822	2.6389589×10^2
ChOA	0.787802	0.410724	2.6389653×10^2
AOA	0.792789	0.396906	2.6392526×10^2
SMFO	0.792044	0.398859	2.6390973×10^2
I-MFO	0.788792	0.407919	2.6389585×10^2

P₃: Tension/compression spring design problem

In the tension/compression spring design problem, the objective is to minimize the weight of the tension/compression spring by considering three variables and four constraints. As shown in Figure 13, the variables are wire diameter (d), the number of active coils (N), and mean coil diameter (D). The problem and its constraints are described in Equation (12) and results are reported in Table 11.

$$\begin{aligned}
 & \text{Minimize } f(x) = x_1^2 x_2 (2 + x_3) \\
 & \text{Subject to } g_1(x) = 1 - \frac{x_2^3 x_3}{71,785 x_1^4} \leq 0 \\
 & g_2(x) = \frac{4x_2^2 - x_1 x_2}{12,566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \\
 & g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\
 & g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
 & \text{Variable range } 0.05 \leq x_1 \leq 2.00 \\
 & 0.25 \leq x_2 \leq 1.30 \\
 & 2.00 \leq x_3 \leq 15.0
 \end{aligned} \tag{12}$$

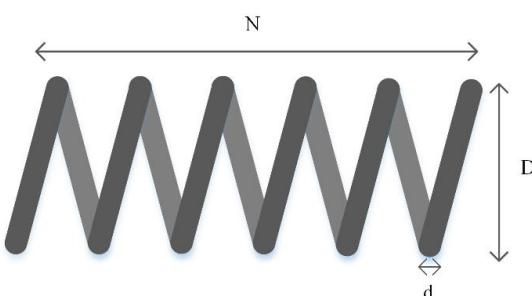


Figure 13. Tension/compression spring design problem.

Table 11. Results for tension/compression spring design problem.

Algorithms	Optimal Values for Variables			Optimum Weight
	<i>d</i>	<i>D</i>	<i>N</i>	
SA	0.075935	0.993094	3.879891	0.033670
CGA	0.071031	1.019975	1.726076	0.019749
GWO	0.051231	0.345699	11.970135	0.012676
MFO	0.053064	0.390718	9.542437	0.012699
WOA	0.050451	0.327675	13.219341	0.012694
LMFO	0.050000	0.317154	14.107156	0.012771
WCMFO	0.051509	0.352411	11.545969	0.012666
ChOA	0.051069	0.341746	12.251078	0.012702
AOA	0.050000	0.310475	15.000000	0.013195
SMFO	0.050000	0.314692	14.696505	0.013136
I-MFO	0.051710	0.357217	11.259785	0.012665

The results of the mechanical engineering problems tabulated in Tables 9–11 demonstrate the fact that the I-MFO is superior to other algorithms for solving real-world mechanical engineering problems.

7. Conclusions and Future Works

The transverse orientation behavior of moths while encountering artificial lights is the main inspiration behind the MFO algorithm to successfully solve optimization problems. However, as with most of the SI algorithms, the MFO suffers from premature convergence, local optima entrapping, low population diversity, and imbalance between exploration and exploitation. These drawbacks make the MFO uncompetitive in solving complex and real-world optimization problems. Therefore, an improved version of the MFO named I-MFO is proposed to improve the MFO algorithm from the perspective of alleviating premature convergence, maintaining population diversity, avoiding local optima trapping, and striking a balance between exploration and exploitation.

To detect local optima-trapped moths, a memory mechanism is defined for each moth. Then, the adapted wandering around search (AWAS) strategy is introduced to possibly free detected trapped moths from local optima by changing their positions while considering the best flame and a random moth position. The CEC 2018 benchmark tasks were conducted to evaluate the performance of the I-MFO, where the reported results in Tables 2–4 and OE in Table 5 prove I-MFO's superior performance over 92% of test functions. The multimodal test function results reported in Table 2 are clear evidence of the fact that the I-MFO boosts exploration rate, especially in more complex problems. The hybrid and composition test function results tabulated in Tables 3 and 4 support the claim that the proposed I-MFO enhances the balance between exploration and exploitation, by which the I-MFO can get out of local optima effectively. The convergence curves also show the local optima avoidance ability and enhanced balance between exploration and exploitation. Moreover, it can be deduced from the population diversity plots that the I-MFO successfully maintains population diversity until a near-optimal solution emerges.

The sensitivity of the AWAS strategy and NF parameter is evaluated on some CEC 2018 benchmark functions for different dimensions, where the results reveal that although the I-MFO offers better solutions in most test functions, other variations of the I-MFO can also provide competitive outcomes for some functions and dimensions. The statistical efficiency of the I-MFO is investigated by the Friedman test and post-hoc analysis, which revealed that the proposed I-MFO outperforms other contender algorithms for various test functions. In the end, the outcomes of the mechanical engineering problems from the latest test-suite CEC 2020 demonstrate that the proposed I-MFO is applicable for solving real-world mechanical engineering problems. Although I-MFO provides competitive results for solving global optimization and engineering tasks, like most improvements, it consumes more time compared to the canonical MFO because it uses the AWAS strategy. Hence, in

practice, the I-MFO may not be suitable for solving large-scale real-time problems. For future works, a multi-objective version of I-MFO can be developed for solving continuous multi-objective problems. Moreover, extending I-MFO to the discrete version for solving discrete optimization tasks such as the community detection problem is a worthwhile direction.

Author Contributions: Conceptualization, M.H.N.-S.; methodology, M.H.N.-S. and H.Z.; software, M.H.N.-S., A.F. and H.Z.; validation, M.H.N.-S. and H.Z.; formal analysis, M.H.N.-S., A.F. and H.Z.; investigation, M.H.N.-S., A.F. and H.Z.; resources, M.H.N.-S., S.M. and L.A.; data curation, M.H.N.-S., A.F. and H.Z.; writing, M.H.N.-S., A.F. and H.Z.; original draft preparation, M.H.N.-S., A.F. and H.Z.; writing—review and editing, M.H.N.-S., A.F., H.Z., S.M. and L.A.; visualization, M.H.N.-S., A.F. and H.Z.; supervision, M.H.N.-S. and S.M.; project administration, M.H.N.-S. and S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and code used in the research may be obtained from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Del Ser, J.; Osaba, E.; Molina, D.; Yang, X.; Salcedo-Sanz, S.; Camacho, D.; Das, S.; Suganthan, P.; Coello, C.C.; Herrera, F. Bio-inspired computation: Where we stand and what's next. *Swarm Evol. Comput.* **2019**, *48*, 220–250. [[CrossRef](#)]
2. Talbi, E.-G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: New York, NY, USA, 2009; Volume 74.
3. Kar, A.K. Bio inspired computing—A review of algorithms and scope of applications. *Expert Syst. Appl.* **2016**, *59*, 20–32. [[CrossRef](#)]
4. Dezfouli, M.B.; Nadimi-Shahraki, M.H.; Zamani, H. A novel tour planning model using big data. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018; pp. 1–6.
5. Zahrani, H.K.; Nadimi-Shahraki, M.H.; Sayarshad, H.R. An intelligent social-based method for rail-car fleet sizing problem. *J. Rail Transp. Plan. Manag.* **2021**, *17*, 100231. [[CrossRef](#)]
6. Javadian, N.; Sayarshad, H.R.; Najafi, S. Using simulated annealing for determination of the capacity of yard stations in a railway industry. *Appl. Soft Comput.* **2011**, *11*, 1899–1907. [[CrossRef](#)]
7. Sayarshad, H.R.; Javadian, N.; Tavakkoli-Moghaddam, R.; Forghani, N. Solving multi-objective optimization formulation for fleet planning in a railway industry. *Ann. Oper. Res.* **2010**, *181*, 185–197. [[CrossRef](#)]
8. Abdollahzadeh, B.; Gharehchopogh, F.S. A multi-objective optimization algorithm for feature selection problems. *Eng. Comput.* **2021**, *1*–19. [[CrossRef](#)]
9. Ewees, A.A.; Al-qaness, M.A.A.; Abualigah, L.; Oliva, D.; Algarni, Z.Y.; Anter, A.M.; Ali Ibrahim, R.; Ghoniem, R.M.; Abd Elaziz, M. Boosting Arithmetic Optimization Algorithm with Genetic Algorithm Operators for Feature Selection: Case Study on Cox Proportional Hazards Model. *Mathematics* **2021**, *9*, 2321. [[CrossRef](#)]
10. Mienye, I.D.; Sun, Y. Improved Heart Disease Prediction Using Particle Swarm Optimization Based Stacked Sparse Autoencoder. *Electronics* **2021**, *10*, 2347. [[CrossRef](#)]
11. Taghian, S.; Nadimi-Shahraki, M.H.; Zamani, H. Comparative analysis of transfer function-based binary Metaheuristic algorithms for feature selection. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018; pp. 1–6.
12. Zamani, H.; Nadimi-Shahraki, M.H. Swarm intelligence approach for breast cancer diagnosis. *Int. J. Comput. Appl.* **2016**, *151*, 40–44. [[CrossRef](#)]
13. Zamani, H.; Nadimi-Shahraki, M.H. Feature selection based on whale optimization algorithm for diseases diagnosis. *Int. J. Comput. Sci. Inf. Secur.* **2016**, *14*, 1243.
14. Ibrahim, R.A.; Abualigah, L.; Ewees, A.A.; Al-Qaness, M.A.; Yousri, D.; Alshathri, S.; Abd Elaziz, M. An Electric Fish-Based Arithmetic Optimization Algorithm for Feature Selection. *Entropy* **2021**, *23*, 1189. [[CrossRef](#)] [[PubMed](#)]
15. Wang, L.; Shi, R.; Dong, J. A Hybridization of Dragonfly Algorithm Optimization and Angle Modulation Mechanism for 0-1 Knapsack Problems. *Entropy* **2021**, *23*, 598. [[CrossRef](#)]
16. Lee, J.; Park, J.; Kim, H.-C.; Kim, D.-W. Competitive Particle Swarm Optimization for Multi-Category Text Feature Selection. *Entropy* **2019**, *21*, 602. [[CrossRef](#)] [[PubMed](#)]
17. Nadimi-Shahraki, M.H.; Moeini, E.; Taghian, S.; Mirjalili, S. DMFO-CD: A Discrete Moth-Flame Optimization Algorithm for Community Detection. *Algorithms* **2021**, *14*, 314. [[CrossRef](#)]

18. Alsalibi, B.; Abualigah, L.; Khader, A.T. A novel bat algorithm with dynamic membrane structure for optimization problems. *Appl. Intell.* **2021**, *51*, 1992–2017. [[CrossRef](#)]
19. Asghari, K.; Masdari, M.; Gharehchopogh, F.S.; Saneifard, R. A chaotic and hybrid gray wolf-whale algorithm for solving continuous optimization problems. *Prog. Artif. Intell.* **2021**, *10*, 349–374. [[CrossRef](#)]
20. Goldanloo, M.J.; Gharehchopogh, F.S. A hybrid OBL-based firefly algorithm with symbiotic organisms search algorithm for solving continuous optimization problems. *J. Supercomput.* **2021**, *1*–34. [[CrossRef](#)]
21. Zaman, H.R.R.; Gharehchopogh, F.S. An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems. *Eng. Comput.* **2021**, *1*–35. [[CrossRef](#)]
22. Doumari, S.A.; Givi, H.; Dehghani, M.; Montazeri, Z.; Leiva, V.; Guerrero, J.M. A New Two-Stage Algorithm for Solving Optimization Problems. *Entropy* **2021**, *23*, 491. [[CrossRef](#)] [[PubMed](#)]
23. Abd Elaziz, M.; Elsheikh, A.H.; Oliva, D.; Abualigah, L.; Lu, S.; Ewees, A.A. Advanced Metaheuristic Techniques for Mechanical Design Problems. *Arch. Comput. Methods Eng.* **2021**, *1*–22. [[CrossRef](#)]
24. Akay, B.; Karaboga, D. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *J. Intell. Manuf.* **2012**, *23*, 1001–1014. [[CrossRef](#)]
25. Aloui, M.; Hamidi, F.; Jerbi, H.; Omri, M.; Popescu, D.; Abbassi, R. A Chaotic Krill Herd Optimization Algorithm for Global Numerical Estimation of the Attraction Domain for Nonlinear Systems. *Mathematics* **2021**, *9*, 1743. [[CrossRef](#)]
26. Gharehchopogh, F.S.; Farnad, B.; Alizadeh, A. A farmland fertility algorithm for solving constrained engineering problems. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6310. [[CrossRef](#)]
27. Ivanov, O.; Neagu, B.-C.; Grigoraş, G.; Scarlatache, F.; Gavrilaş, M. A Metaheuristic Algorithm for Flexible Energy Storage Management in Residential Electricity Distribution Grids. *Mathematics* **2021**, *9*, 2375. [[CrossRef](#)]
28. Wang, S.; Jia, H.; Abualigah, L.; Liu, Q.; Zheng, R. An Improved Hybrid Aquila Optimizer and Harris Hawks Algorithm for Solving Industrial Engineering Optimization Problems. *Processes* **2021**, *9*, 1551. [[CrossRef](#)]
29. Ziadeh, A.; Abualigah, L.; Abd Elaziz, M.; Şahin, C.B.; Almazroi, A.A.; Omari, M. Augmented grasshopper optimization algorithm by differential evolution: A power scheduling application in smart homes. *Multimed. Tools Appl.* **2021**, *80*, 31569–31597. [[CrossRef](#)]
30. Varaei, H.; Ghasemi, M.R. Engineering optimization based on ideal gas molecular movement algorithm. *Eng. Comput.* **2017**, *33*, 71–93. [[CrossRef](#)]
31. Ghasemi, M.R.; Varaei, H. A fast multi-objective optimization using an efficient ideal gas molecular movement algorithm. *Eng. Comput.* **2017**, *33*, 477–496. [[CrossRef](#)]
32. Hua, Z.; Xiao, Y.; Cao, J. Misalignment Fault Prediction of Wind Turbines Based on Improved Artificial Fish Swarm Algorithm. *Entropy* **2021**, *23*, 692. [[CrossRef](#)]
33. Wang, S.; Liu, Q.; Liu, Y.; Jia, H.; Abualigah, L.; Zheng, R.; Wu, D. A Hybrid SSA and SMA with Mutation Opposition-Based Learning for Constrained Engineering Problems. *Comput. Intell. Neurosci.* **2021**, *2021*, 6379469. [[CrossRef](#)]
34. Selvaraj, S.; Choi, E. Swarm Intelligence Algorithms in Text Document Clustering with Various Benchmarks. *Sensors* **2021**, *21*, 3196. [[CrossRef](#)]
35. Bacanin, N.; Bezdan, T.; Venkatachalam, K.; Al-Turjman, F. Optimized convolutional neural network by firefly algorithm for magnetic resonance image classification of glioma brain tumor grade. *J. Real-Time Image Process.* **2021**, *18*, 1085–1098. [[CrossRef](#)]
36. Bacanin, N.; Bezdan, T.; Tuba, E.; Strumberger, I.; Tuba, M. Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics. *Algorithms* **2020**, *13*, 67. [[CrossRef](#)]
37. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [[CrossRef](#)]
38. Glover, F.; Laguna, M. Tabu search. In *Handbook of Combinatorial Optimization*; Springer: Boston, MA, USA, 1998; pp. 2093–2229.
39. Hasançebi, O.; Azad, S.K. Adaptive dimensional search: A new metaheuristic algorithm for discrete truss sizing optimization. *Comput. Struct.* **2015**, *154*, 1–16. [[CrossRef](#)]
40. Lourenço, H.R.; Martin, O.C.; Stützle, T. Iterated local search: Framework and applications. In *Handbook of Metaheuristics*; Springer: Cham, Switzerland, 2019; pp. 129–168.
41. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [[CrossRef](#)]
42. Chelouah, R.; Siarry, P. A continuous genetic algorithm designed for the global optimization of multimodal functions. *J. Heuristics* **2000**, *6*, 191–213. [[CrossRef](#)]
43. Koza, J.R. Genetic programming as a means for programming computers by natural selection. *Stat. Comput.* **1994**, *4*, 87–112. [[CrossRef](#)]
44. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [[CrossRef](#)]
45. Beyer, H.-G.; Schwefel, H.-P. Evolution strategies—A comprehensive introduction. *Nat. Comput.* **2002**, *1*, 3–52. [[CrossRef](#)]
46. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. QANA: Quantum-based avian navigation optimizer algorithm. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104314. [[CrossRef](#)]
47. Mallipeddi, R.; Suganthan, P.N.; Pan, Q.-K.; Tasgetiren, M.F. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl. Soft Comput.* **2011**, *11*, 1679–1696. [[CrossRef](#)]
48. Wu, G.; Mallipeddi, R.; Suganthan, P.N.; Wang, R.; Chen, H. Differential evolution with multi-population based ensemble of mutation strategies. *Inf. Sci.* **2016**, *329*, 329–345. [[CrossRef](#)]

49. Wu, G.; Shen, X.; Li, H.; Chen, H.; Lin, A.; Suganthan, P.N. Ensemble of differential evolution variants. *Inf. Sci.* **2018**, *423*, 172–186. [[CrossRef](#)]
50. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S.; Faris, H. MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. *Appl. Soft Comput.* **2020**, *97*, 106761. [[CrossRef](#)]
51. Erol, O.K.; Eksin, I. A new optimization method: Big bang—Big crunch. *Adv. Eng. Softw.* **2006**, *37*, 106–111. [[CrossRef](#)]
52. Kaveh, A.; Talatahari, S. A novel heuristic optimization method: Charged system search. *Acta Mech.* **2010**, *213*, 267–289. [[CrossRef](#)]
53. Kaveh, A.; Khayatazad, M. A new meta-heuristic method: Ray optimization. *Comput. Struct.* **2012**, *112*, 283–294. [[CrossRef](#)]
54. Zhao, W.; Wang, L.; Zhang, Z. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowl.-Based Syst.* **2019**, *163*, 283–304. [[CrossRef](#)]
55. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [[CrossRef](#)]
56. Azizi, M. Atomic orbital search: A novel metaheuristic algorithm. *Appl. Math. Model.* **2021**, *93*, 657–683. [[CrossRef](#)]
57. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. CCSA: Conscious neighborhood-based crow search algorithm for solving global optimization problems. *Appl. Soft Comput.* **2019**, *85*, 105583. [[CrossRef](#)]
58. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [[CrossRef](#)]
59. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [[CrossRef](#)]
60. Abdollahzadeh, B.; Soleimanian Gharehchopogh, F.; Mirjalili, S. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *Int. J. Intell. Syst.* **2021**, *36*, 5887–5958. [[CrossRef](#)]
61. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S. An improved grey wolf optimizer for solving engineering problems. *Expert Syst. Appl.* **2021**, *166*, 113917. [[CrossRef](#)]
62. Jia, H.; Sun, K.; Zhang, W.; Leng, X. An enhanced chimp optimization algorithm for continuous optimization domains. *Complex Intell. Syst.* **2021**, *1*–18. [[CrossRef](#)]
63. Liu, Y.; Sun, J.; Yu, H.; Wang, Y.; Zhou, X. An Improved Grey Wolf Optimizer Based on Differential Evolution and OTSU Algorithm. *Appl. Sci.* **2020**, *10*, 6343. [[CrossRef](#)]
64. Chen, C.; Wang, X.; Chen, H.; Wu, C.; Mafarja, M.; Turabieh, H. Towards Precision Fertilization: Multi-Strategy Grey Wolf Optimizer Based Model Evaluation and Yield Estimation. *Electronics* **2021**, *10*, 2183. [[CrossRef](#)]
65. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [[CrossRef](#)]
66. Kaveh, A.; Farhoudi, N. A new optimization method: Dolphin echolocation. *Adv. Eng. Softw.* **2013**, *59*, 53–70. [[CrossRef](#)]
67. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [[CrossRef](#)]
68. Dorigo, M.; Di Caro, G. Ant colony optimization: A new meta-heuristic. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; pp. 1470–1477.
69. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [[CrossRef](#)]
70. James, J.; Li, V.O. A social spider algorithm for global optimization. *Appl. Soft Comput.* **2015**, *30*, 614–627.
71. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Comput. Struct.* **2016**, *169*, 1–12. [[CrossRef](#)]
72. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [[CrossRef](#)]
73. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization Algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [[CrossRef](#)]
74. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [[CrossRef](#)]
75. Abd El Aziz, M.; Ewees, A.A.; Hassanien, A.E. Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Syst. Appl.* **2017**, *83*, 242–256. [[CrossRef](#)]
76. Jia, H.; Ma, J.; Song, W. Multilevel thresholding segmentation for color image using modified moth-flame optimization. *IEEE Access* **2019**, *7*, 44097–44134. [[CrossRef](#)]
77. Khan, M.A.; Sharif, M.; Akram, T.; Damaševičius, R.; Maskeliūnas, R. Skin Lesion Segmentation and Multiclass Classification Using Deep Learning Features and Improved Moth Flame Optimization. *Diagnostics* **2021**, *11*, 811. [[CrossRef](#)] [[PubMed](#)]
78. Abd Elaziz, M.; Ewees, A.A.; Ibrahim, R.A.; Lu, S. Opposition-based moth-flame optimization improved by differential evolution for feature selection. *Math. Comput. Simul.* **2020**, *168*, 48–75. [[CrossRef](#)]
79. Gupta, D.; Ahlawat, A.K.; Sharma, A.; Rodrigues, J.J. Feature selection and evaluation for software usability model using modified moth-flame optimization. *Computing* **2020**, *102*, 1503–1520. [[CrossRef](#)]
80. Tumar, I.; Hassouneh, Y.; Turabieh, H.; Thaher, T. Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction. *IEEE Access* **2020**, *8*, 8041–8055. [[CrossRef](#)]
81. Abu Khurmaa, R.; Aljarah, I.; Sharieh, A. An intelligent feature selection approach based on moth flame optimization for medical diagnosis. *Neural Comput. Appl.* **2021**, *33*, 7165–7204. [[CrossRef](#)]
82. Nadimi-Shahraki, M.H.; Banaie-Dezfouli, M.; Zamani, H.; Taghian, S.; Mirjalili, S. B-MFO: A Binary Moth-Flame Optimization for Feature Selection from Medical Datasets. *Computers* **2021**, *10*, 136. [[CrossRef](#)]

83. Sarma, A.; Bhutani, A.; Goel, L. Hybridization of moth flame optimization and gravitational search algorithm and its application to detection of food quality. In Proceedings of the 2017 Intelligent Systems Conference (IntelliSys), London, UK, 7–8 September 2017; pp. 52–60.
84. Hassanien, A.E.; Gaber, T.; Mokhtar, U.; Hefny, H. An improved moth flame optimization algorithm based on rough sets for tomato diseases detection. *Comput. Electron. Agric.* **2017**, *136*, 86–96. [[CrossRef](#)]
85. Lei, X.; Fang, M.; Fujita, H. Moth–flame optimization-based algorithm with synthetic dynamic PPI networks for discovering protein complexes. *Knowl.-Based Syst.* **2019**, *172*, 76–85. [[CrossRef](#)]
86. Li, C.; Li, S.; Liu, Y. A least squares support vector machine model optimized by moth-flame optimization algorithm for annual power load forecasting. *Appl. Intell.* **2016**, *45*, 1166–1178. [[CrossRef](#)]
87. Mei, R.N.S.; Sulaiman, M.H.; Mustaffa, Z.; Daniyal, H. Optimal reactive power dispatch solution by loss minimization using moth-flame optimization technique. *Appl. Soft Comput.* **2017**, *59*, 210–222.
88. Allam, D.; Yousri, D.; Eteiba, M. Parameters extraction of the three diode model for the multi-crystalline solar cell/module using Moth-Flame Optimization Algorithm. *Energy Convers. Manag.* **2016**, *123*, 535–548. [[CrossRef](#)]
89. Ebrahim, M.A.; Becherif, M.; Abdelaziz, A.Y. Dynamic performance enhancement for wind energy conversion system using Moth-Flame Optimization based blade pitch controller. *Sustain. Energy Technol. Assess.* **2018**, *27*, 206–212. [[CrossRef](#)]
90. Raju, K.; Madurai Elavarasan, R.; Mihet-Popa, L. An assessment of onshore and offshore wind energy potential in India using moth flame optimization. *Energies* **2020**, *13*, 3063. [[CrossRef](#)]
91. Rezk, H.; Ali, Z.M.; Abdalla, O.; Younis, O.; Gomaa, M.R.; Hashim, M. Hybrid moth-flame optimization algorithm and incremental conductance for tracking maximum power of solar PV/thermoelectric system under different conditions. *Mathematics* **2019**, *7*, 875. [[CrossRef](#)]
92. Li, Z.; Zhou, Y.; Zhang, S.; Song, J. Lévy-flight moth-flame algorithm for function optimization and engineering design problems. *Math. Probl. Eng.* **2016**, *2016*, 1423930. [[CrossRef](#)]
93. Savsani, V.; Tawhid, M.A. Non-dominated sorting moth flame optimization (NS-MFO) for multi-objective problems. *Eng. Appl. Artif. Intell.* **2017**, *63*, 20–32. [[CrossRef](#)]
94. Xu, L.; Li, Y.; Li, K.; Beng, G.H.; Jiang, Z.; Wang, C.; Liu, N. Enhanced moth-flame optimization based on cultural learning and Gaussian mutation. *J. Bionic Eng.* **2018**, *15*, 751–763. [[CrossRef](#)]
95. Khalilpourazari, S.; Khalilpourazary, S. An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems. *Soft Comput.* **2019**, *23*, 1699–1722. [[CrossRef](#)]
96. Chen, C.; Wang, X.; Yu, H.; Wang, M.; Chen, H. Dealing with multi-modality using synthesis of Moth-flame optimizer with sine cosine mechanisms. *Math. Comput. Simul.* **2021**, *188*, 291–318. [[CrossRef](#)]
97. Kaur, K.; Singh, U.; Salgotra, R. An enhanced moth flame optimization. *Neural Comput. Appl.* **2020**, *32*, 2315–2349. [[CrossRef](#)]
98. Pelusi, D.; Mascella, R.; Tallini, L.; Nayak, J.; Naik, B.; Deng, Y. An Improved Moth-Flame Optimization algorithm with hybrid search phase. *Knowl.-Based Syst.* **2020**, *191*, 105277. [[CrossRef](#)]
99. Hongwei, L.; Jianyong, L.; Liang, C.; Jingbo, B.; Yangyang, S.; Kai, L. Chaos-enhanced moth-flame optimization algorithm for global optimization. *J. Syst. Eng. Electron.* **2019**, *30*, 1144–1159.
100. Xu, Y.; Chen, H.; Luo, J.; Zhang, Q.; Jiao, S.; Zhang, X. Enhanced Moth-flame optimizer with mutation strategy for global optimization. *Inf. Sci.* **2019**, *492*, 181–203. [[CrossRef](#)]
101. Li, Y.; Zhu, X.; Liu, J. An improved moth-flame optimization algorithm for engineering problems. *Symmetry* **2020**, *12*, 1234. [[CrossRef](#)]
102. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P. Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective bound constrained real-parameter numerical optimization. In *Technical Report*; Nanyang Technological University: Singapore, 2016.
103. Kumar, A.; Wu, G.; Ali, M.Z.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evol. Comput.* **2020**, *56*, 100693. [[CrossRef](#)]
104. Hussien, A.G.; Amin, M.; Abd El Aziz, M. A comprehensive review of moth-flame optimisation: Variants, hybrids, and applications. *J. Exp. Theor. Artif. Intell.* **2020**, *32*, 705–725. [[CrossRef](#)]
105. Mehne, S.H.H.; Mirjalili, S. Moth-flame optimization algorithm: Theory, literature review, and application in optimal nonlinear feedback control design. In *Nature-Inspired Optimizers*; Springer: Cham, Switzerland, 2020; pp. 143–166.
106. Shehab, M.; Abualigah, L.; Al Hamad, H.; Alabool, H.; Alshinwan, M.; Khasawneh, A.M. Moth–flame optimization algorithm: Variants and applications. *Neural Comput. Appl.* **2020**, *32*, 9859–9884. [[CrossRef](#)]
107. Apinantranakon, W.; Sunat, K. Omfo: A new opposition-based moth-flame optimization algorithm for solving unconstrained optimization problems. In Proceedings of the International Conference on Computing and Information Technology, Singapore, 27–29 December 2017; pp. 22–31.
108. Sapre, S.; Mini, S. Opposition-based moth flame optimization with Cauchy mutation and evolutionary boundary constraint handling for global optimization. *Soft Comput.* **2019**, *23*, 6023–6041. [[CrossRef](#)]
109. Yu, C.; Heidari, A.A.; Chen, H. A quantum-behaved simulated annealing algorithm-based moth-flame optimization method. *Appl. Math. Model.* **2020**, *87*, 1–19. [[CrossRef](#)]

110. Bhedsadiya, R.; Trivedi, I.N.; Jangir, P.; Kumar, A.; Jangir, N.; Totlani, R. A novel hybrid approach particle swarm optimizer with moth-flame optimizer algorithm. In *Advances in Computer and Computational Sciences*; Springer: Singapore, 2017; pp. 569–577.
111. Sayed, G.I.; Hassanien, A.E. A hybrid SA-MFO algorithm for function optimization and engineering design problems. *Complex Intell. Syst.* **2018**, *4*, 195–212. [[CrossRef](#)]
112. Singh, R.K.; Gangwar, S.; Singh, D.; Pathak, V.K. A novel hybridization of artificial neural network and moth-flame optimization (ANN-MFO) for multi-objective optimization in magnetic abrasive finishing of aluminium 6060. *J. Braz. Soc. Mech. Sci. Eng.* **2019**, *41*, 270. [[CrossRef](#)]
113. Dang, M.P.; Le, H.G.; Chau, N.L.; Dao, T.-P. Optimization for a flexure hinge using an effective hybrid approach of fuzzy logic and moth-flame optimization algorithm. *Math. Probl. Eng.* **2021**, *2021*, 6622655. [[CrossRef](#)]
114. Mittal, T. A hybrid moth flame optimization and variable neighbourhood search technique for optimal design of IIR filters. *Neural Comput. Appl.* **2021**, *1*–16. [[CrossRef](#)]
115. Abd Elaziz, M.; Yousri, D.; Mirjalili, S. A hybrid Harris hawks-moth-flame optimization algorithm including fractional-order chaos maps and evolutionary population dynamics. *Adv. Eng. Softw.* **2021**, *154*, 102973. [[CrossRef](#)]
116. Ahmed, O.H.; Lu, J.; Xu, Q.; Ahmed, A.M.; Rahmani, A.M.; Hosseinzadeh, M. Using differential evolution and Moth–Flame optimization for scientific workflow scheduling in fog computing. *Appl. Soft Comput.* **2021**, *112*, 107744. [[CrossRef](#)]
117. Li, Z.; Zeng, J.; Chen, Y.; Ma, G.; Liu, G. Death mechanism-based moth–flame optimization with improved flame generation mechanism for global optimization tasks. *Expert Syst. Appl.* **2021**, *183*, 115436. [[CrossRef](#)]
118. Blackiston, D.J.; Silva Casey, E.; Weiss, M.R. Retention of memory through metamorphosis: Can a moth remember what it learned as a caterpillar? *PLoS ONE* **2008**, *3*, e1736. [[CrossRef](#)]
119. Morrison, R.W. *Designing Evolutionary Algorithms for Dynamic Environments*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2004.
120. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [[CrossRef](#)]
121. Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* **1940**, *11*, 86–92. [[CrossRef](#)]