processes

MDPI

Article

Deep Ensemble of Slime Mold Algorithm and Arithmetic Optimization Algorithm for Global Optimization

Rong Zheng [1,*], Heming Jia [1,*], Laith Abualigah [2,3,4], Qingxin Liu [5] and Shuang Wang [1]

[1] School of Information Engineering, Sanming University, Sanming 365004, China; wang_shuang@fjsmu.edu.cn
[2] Research and Innovation Department, Skyline University College, Sharjah 1797, United Arab Emirates; Aligah.2020@gmail.com
[3] Faculty of Computer Sciences and Informatics, Amman Arab University, Amman 11953, Jordan
[4] School of Computer Science, Universiti Sains Malaysia, Gelugor 11800, Malaysia
[5] School of Computer Science and Technology, Hainan University, Haikou 570228, China; qxliu@hainanu.edu.cn
* Correspondence: zhengr@fjsmu.edu.cn (R.Z.); jiaheming@fjsmu.edu.cn (H.J.)

Abstract: In this paper, a new hybrid algorithm based on two meta-heuristic algorithms is presented to improve the optimization capability of original algorithms. This hybrid algorithm is realized by the deep ensemble of two new proposed meta-heuristic methods, i.e., slime mold algorithm (SMA) and arithmetic optimization algorithm (AOA), called DESMAOA. To be specific, a preliminary hybrid method was applied to obtain the improved SMA, called SMAOA. Then, two strategies that were extracted from the SMA and AOA, respectively, were embedded into SMAOA to boost the optimizing speed and accuracy of the solution. The optimization performance of the proposed DESMAOA was analyzed by using 23 classical benchmark functions. Firstly, the impacts of different components are discussed. Then, the exploitation and exploration capabilities, convergence behaviors, and performances are evaluated in detail. Cases at different dimensions also were investigated. Compared with the SMA, AOA, and another five well-known optimization algorithms, the results showed that the proposed method can outperform other optimization algorithms with high superiority. Finally, three classical engineering design problems were employed to illustrate the capability of the proposed algorithm for solving the practical problems. The results also indicate that the DESMAOA has very promising performance when solving these problems.

Keywords: slime mold algorithm; arithmetic optimization algorithm; meta-heuristics algorithm; global optimization; engineering design problem

Citation: Zheng, R.; Jia, H.; Abualigah, L.; Liu, Q.; Wang, S. Deep Ensemble of Slime Mold Algorithm and Arithmetic Optimization Algorithm for Global Optimization. *Processes* **2021**, *9*, 1774. https://doi.org/10.3390/pr9101774

Academic Editor: Jae-Yoon Jung

Received: 18 August 2021
Accepted: 2 October 2021
Published: 4 October 2021

1. Introduction

Nowadays, optimization problems exist in various scenarios, for instance, the engineering design problems. The objective of these optimization problems is to find the extreme values with determined constraint conditions. Then, commonly, the cost is reduced as much as possible. To tackle these problems, researchers have proposed many optimization algorithms [1–4]. Generally speaking, traditional optimization algorithms, such as gradient-based methods, are susceptible to the initial positions and have difficulties to deal with the non-convex problems that may contain a mass of local optimums. In practice, when we are faced with complex constraint conditions in the real world, it is more essential to obtain the optimal solutions within limited time and cost. At this point, the important thing is not to find the theoretical optimal result but to obtain as good an approximate solution as possible under restricted conditions. For this purpose, many stochastic optimizers have been developed and employed to solve complex optimization problems. As its name implies, the random operator is the main feature for a stochastic optimizer, which allows the algorithms to avoid the stagnation and search the whole search region for global optimization result.

The meta-heuristic algorithms (MAs) have shown very powerful capability in the fields of computational sciences. In general, MAs have four types according to the sources of inspiration, namely, physics-inspired (PI), evolution-inspired (EI), swarm-inspired (SI), and human-inspired (HI). Some representative algorithms are shown below:

- Physics-inspired: multi-verse optimizer (MVO) [5], gravitational search algorithm (GSA) [6], thermal exchange optimization (TEO) [7], heat transfer relation-based optimization algorithm (HTOA) [8].
- Evolution-inspired: genetic algorithm (GA) [9], differential evolution (DE) [10], evolutionary programming (EP) [11].
- Swarm-inspired: particle swarm optimization (PSO) [12], emperor penguin optimizer (EPO) [13], Aquila optimizer (AO) [14], remora optimization algorithm (ROA) [4], marine predators algorithm (MPA) [15].
- Human-inspired: teaching–learning-based optimization (TLBO) [16], social group optimization (SGO) [17], β-hill climbing (βHC) [18], coronavirus optimization algorithm (COA) [19].

For a meta-heuristic algorithm, one important thing is to balance of the global search and local search [20]. It is known that the search agents are first randomly generated within the search spaces. Then, positions of these search agents are updated according to the formulas in the algorithm. In the early stage, drastic exploration in the search space should be performed as much as possible in the early stage. Then, in the later phase, more local exploitation should be conducted to improve the accuracy of obtained optimal solution. Although hundreds of MAs have been proposed for the optimization problems, there is still need for new algorithms to solve these optimization problems. According to the No-Free-Lunch (NFL) theory [21], on one optimization algorithm can solve all the optimization problems. Generally speaking, it is common that MAs suffer from local optimum stagnation and poor convergence speed as a result of poor optimization ability. Thus, it is very important to develop new optimization algorithms or improve existing MAs by taking some effective measures. Up until now, there have been three primary methods for the improvements of the existing algorithms, which are listed in Table 1.

The slime mold algorithm (SMA) [33] and arithmetic optimization algorithm (AOA) [34] are two newly proposed MAs, and both have the merits of simplicity, efficiency, and flexibility. The SMA has good population diversity and stable performance when solving optimization problems. However, it gets stuck in local optima sometimes for the limited global search capability. On the contrary, the AOA has powerful exploration capability by using the arithmetic operators. However, the performance of AOA is not stable because of the poor population diversity. Therefore, the SMA and AOA are considered to be hybridized together in this paper for solving the global optimization problems. To evaluate the performance of proposed algorithm, we employed 23 classical benchmark functions and 4 constrained engineering design problems. The main contributions of this works are as follows:

1. Hybridizing the slime mold algorithm (SMA) [33] and arithmetic optimization algorithm (AOA) [34] named SMAOA to improve the exploration capability of original SMA.
2. Applying the random contraction strategy (RCS), which is inspired from SMA to help the SMAOA jump out from local optimum.
3. Applying the subtraction and addition strategy (SAS), which is extracted from AOA to enhance the exploitation ability of SMAOA.
4. When the RCS and SAS were applied on SMAOA, the DESMAOA was finally obtained. By comparing seven well-known optimization algorithms, we identified the proposed DESMAOA to be powerful according to the experimental results.

Table 1. A summary of methods for improving the optimization algorithms developed in the literature.

Name of Method	Representative Algorithm	Description
Hybridize two or more algorithms	Hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) [22]	The capability of exploitation in SSO and the capability of exploration in GSA are combined for better performance.
	Imperialist competitive Harris hawks optimization (ICHHO) [23]	The exploration of ICA is utilized to improve the HHO for global optimization.
	Hybrid particle swarm and spotted hyena optimizer (HPSSHO) [24]	Particle swarm algorithm is used to improve the hunting strategy of spotted hyena optimizer.
Add one or more strategies onto an algorithm	Sine-cosine and spotted hyena-based chimp optimization algorithm (SSC) [25]	Sine-cosine functions and attacking strategy of SHO are embedded in ChoA for better exploration and exploitation.
	Representative-based grey wolf optimizer (R-GWO) [26]	A search strategy named representative-based hunting (RH) is utilized to improve the exploration and diversity of the population
	Reinforced salp swarm algorithm (CMSRSSSA) [27]	An ensemble/composite mutation strategy (CMS) is applied to boost the exploitation and exploration speed of SSA, while restart strategy (RS) is used to get away from local optimum.
	Boosting quantum rotation gate embedded slime mold algorithm (WQSMA) [28]	The quantum rotation gate mechanism and the operation from water cycle are applied to balance the exploration and exploitation inclinations.
	Enhanced salp swarm algorithm (ESSA) [29]	Orthogonal learning, quadratic interpolation, and generalized oppositional learning are embedded into SSA to boost the global exploration and local exploitation.
Hybridize two or more algorithms that are further improved by one or more strategies	Whale optimization with seagull algorithm (WSOA) [30]	WOA's contraction surrounding mechanism and SOA's spiral attack behavior work together, and then levy flight strategy is employed on the search process of SOA.
	Chaotic sine-cosine firefly (CSCF) algorithm [31]	Chaotic form of SCA and FA are integrated together to improve the convergence speed and efficiency.
	Hybrid grasshopper optimization algorithm with bat algorithm (BGOA) [32]	In BGOA, Levy fight, local search part of BA, and random strategy are introduced into basic GOA.

The rest of this paper is organized as follows: The basics of SMA and AOA are described in Section 2. Then, the hybrid method is presented in Section 3, including two strategies that are obtained from these two algorithms. In Section 4, a series of experimental tests are conducted to evaluate the performance of proposed DESMAOA. In Section 5, three engineering design problems are employed to assess the applicability of proposed algorithm in practice. Finally, Section 6 concludes this paper and provides some directions for meaningful future research.

2. Preliminaries

2.1. Slime Mold Algorithm (SMA)

The slime mold algorithm (SMA) is a recent meta-heuristic algorithm proposed by Li et al. in 2020 [33]. The basic idea of SMA is based on the foraging behavior of slime mold, which have different feedback characteristics according to the food quality. Three special behaviors of the slime mold are mathematical formulated in the SMA, i.e., approaching food, wrapping food, and finally grabbling food. First, the process of approaching food can be expressed as

$$Xi(t+1) = \begin{cases} Xb(t) + vb \cdot (W \cdot XA(t) - XB(t)), & r1 < p \\ vc \cdot Xi(t), & r1 \geq p \end{cases} \quad (1)$$

where t is the number of current iteration, $X_i(t+1)$ is the newly generated position, $X_b(t)$ denotes the best position found by slime mold in iteration t, $X_A(t)$ and $X_B(t)$ are two random positions selected from the population of slime mold, and r_1 is a random value in [0, 1].

vb and vc are the coefficients that simulate the oscillation and contraction mode of slime mold, respectively, and vc is designed to linearly decrease from one to zero during the iterations. The range of vb is from $-a$ to a, and the computational formula of a is

$$a = \operatorname{arctanh}(1 - \frac{t}{T}) \quad (2)$$

where T is the maximum number of iterations.

According to Equations (1) and (2), it can be seen that as the number of iterations increases, the slime mold will wrap the food.

W is a very important factor that indicates the weight of slime mold, and it is calculated as follows:

$$W(SmellIndex(i)) = \begin{cases} 1 + rand \cdot \log(\frac{bF - S(i)}{bF - wF} + 1), & i \leq N/2 \\ 1 - rand \cdot \log(\frac{bF - S(i)}{bF - wF} + 1), & i > N/2 \end{cases} \quad (3)$$

$$SmellIndex(i) = sort(S(i)) \quad (4)$$

where *rand* means a random value between 0 and 1; bF and wF are the best and worst fitness values, respectively, obtained by far; $S(i)$ is the fitness value of ith slime mold; N is the popsize of the population; and *SmellIndex* is a ranking of fitness values for individuals in the population.

In Equation (1), it is also worth noting that p is the probability of determining the update location for slime mold, which is related to the fitness values of slime mold and food and can be calculated as follows:

$$p = \tanh|S(i) - DF| \quad (5)$$

where DF denotes the best fitness obtained by population.

Finally, when the slime mold has found the food (i.e., grabble food), it still has a certain chance (z) to search other new food, which is formulated as

$$X(t+1) = rand \cdot (UB - LB) + LB, \; r_2 < z \quad (6)$$

where UB and LB are the upper boundary and lower boundary, respectively, and r_2 implies a random value in the region [0, 1].

In general, z should be very small; thus, it is set to 0.03 in SMA. Finally, the pseudo-code of SMA is given in Algorithm 1.

Algorithm 1. Pseudo-code of SMA

Initialize the parameters popsize (N) and maximum iterations (T)
Initialize the positions of all slime mold X_i (i = 1, 2, . . . , N)
While ($t \leq T$)
Calculate the fitness of all slime mold
Update *bestFitness*, X_b
Calculate the weight W by Equation (3) and (4)
For each search agent
If $r_2 < z$
Update position by Equation (6)
Else
Update p, vb, and vc
Update position by Equation (1)
End if
End for
$t = t + 1$
End While
Return *bestFitness*, X_b

2.2. Arithmetic Optimization Algorithm (AOA)

Arithmetic optimization algorithm (AOA) is a very new meta-heuristic method proposed by Abualigah and others in 2021 [34]. The main inspiration of this algorithm is to combine the four traditional arithmetic operators in mathematics, i.e., multiplication (M), division (D), subtraction (S), and addition (A). Similar to sine-cosine algorithm (SCA) [35], AOA also has a very simple structure and low computation complexity. Considering the

M and D operators can produce large steps in the iterations, M and D are hence mainly conducted in the exploration phase. The expression is as follows:

$$Xi(t+1) = \begin{cases} Xb(t)/(MOP + eps) \cdot ((UB - LB)\mu + LB), \ rand < 0.5 \\ Xb(t) \cdot MOP \cdot ((UB - LB)\mu + LB), \ rand \geq 0.5 \end{cases} \quad (7)$$

where eps is a very small positive number, and μ is a constant coefficient (0.499) that is carefully designed for this algorithm.

MOP is non-linearly decreased from 1 to 0 during the iterations, and the expression is as follows:

$$MOP = 1 - (\frac{t}{T})^{1/\alpha} \quad (8)$$

where α is a constant value, which is set to 5 according to the AOA.

From Equation (7), it can be seen that both M and D operators can generate very stochastic positions for the search agent on the basis of the best position. By contrast, S and A operators are applied to emphasize the local exploitation that will generate smaller steps in the search space. The mathematical expression is defined as

$$Xi(t+1) = \begin{cases} Xb(t) - MOP \cdot ((UB - LB)\mu + LB), \ rand < 0.5 \\ Xb(t) + MOP \cdot ((UB - LB)\mu + LB), \ rand \geq 0.5 \end{cases} \quad (9)$$

There is no doubt that the importance of balance between exploration and exploitation for an optimization algorithm. In AOA, the parameter MOA is utilized to switch the exploration and exploitation over the course of iterations, which is expressed as

$$MOA(t) = Min + t(\frac{Max - Min}{T}) \quad (10)$$

where Min and Max are constant values.

According to Equation (10), MOA increases from Min to Max. Thus, in the early phase, search agent has more chance to perform exploration in the search space, while in the later stage, search agent will be more likely to conduct search near the best position. The pseudo-code of AOA is shown in Algorithm 2.

Algorithm 2. Pseudo-code of AOA

Initialize the parameters popsize (N) and maximum iterations (T)
Initialize the positions of all search agents X_i (i = 1, 2, ... , N)
Set the parameters α, μ, Min, and Max
While ($t \leq T$)
Calculate the fitness of all search agents
Update *bestFitness*, X_b
Calculate the MOP by Equation (8)
Calculate the MOA by Equation (10)
For each search agent
If *rand* > MOA
Update position by Equation (7)
Else
Update position by Equation (9)
End if
End for
$t = t + 1$
End While
Return *bestFitness*, X_b

3. The Proposed Hybridized Algorithm (DESMAOA)

It is well known that MAs have the merits of concision, flexibility, and especially utility. Hence, many scholars are working on developing new meta-heuristic-based approaches for optimization problems. However, several optimization algorithms such as slime mold algorithm and arithmetic optimization algorithm still have some drawbacks. For instance, when dealing with complex optimization problems, SMA tends to drop into local best, and also converges slowly. Similarly, AOA only utilizes the information of best position in the population, which may suffer the problem of low precision. Therefore, this paper aimed to develop a new hybridization algorithm composed of SMA and AOA for better optimization performance.

In this paper, the SMA and AOA are firstly integrated to form a hybridized style named SMAOA. Then, the preliminary hybrid algorithm is further enhanced by adding two strategies. One is the random contraction strategy (RCS), which is an improved version of contraction formula in SMA. The other is the subtraction and addition strategy (SAS), which is extracted from the local search in AOA. Finally, the deep ensemble of SMA and AOA is accomplished, and the hybridized algorithm (i.e., DESMAOA) is obtained. The detailed implement of proposed algorithm is delineated in the following.

3.1. The Hybridization of SMA and AOA

In SMA, the contraction formula (see Equations (1) and (2)) is utilized to help slime mold jump out of local minima, which will tend to zero in the later iterations. Thus, it will not play the role of global exploration. On the other hand, the multiplication and division methods in AOA display a powerful capability in global exploration. Thus, the formulas of multiplication and division (see in Equation (7)) are considered to replace the contraction equation. Therefore, the hybrid algorithm SMAOA will perform good global search in the whole stage. To be specific, for the search agent that is close to best position, the multiplication and division operators will make it more likely to search other spaces.

3.2. Random Contraction Strategy (RCS)

In this work, we present the RCS on the basis of the mathematical formula of contraction mode in SMA, which is applied to expand exploration space and avoid local optimum. The coefficient vc is replaced by a random value lying between -1 and 1. The position update formula is calculated as follows:

$$Vi2(t+1) = (2rand - 1)Xi(t) \tag{11}$$

From Equation (11), we should note that the generated position of RCS is within the range $[-|X_i(t)|, |X_i(t)|]$ with uniform distribution, which adds more flexibility for the search agents in the proposed algorithm. Note that the generated position of RCS is taken as a candidate solution.

3.3. Subtraction and Addition Strategy (SAS)

The other strategy proposed here is the SAS, which is also the exploitation method of AOA. According to the AOA, SAS can be performed locally and increase the accuracy of solutions effectively. It is worth mentioning here that the SAS is conducted behind the RCS.

In the same way, the position generated by SAS is treated as a candidate solution, and if a better position is found, then it will be adopted.

3.4. The Deep Ensemble of SMA and AOA

As mentioned above, the SMA and AOA are hybridized together firstly to achieve the SMAOA. Then, two strategies are introduced in the SMAOA, namely, random contraction strategy and subtraction and addition strategy. In order to perform a better balance effect between exploration and exploitation, we utilize a parameter, b, that is related with

iterations to represent the probability of conducting the strategies. Its computational formula is given below:

$$b = 1 - \frac{t}{T} \tag{12}$$

The pseudo-code of DESMAOA is shown in Algorithm 3. Moreover, the flowchart of proposed method is shown in Figure 1.

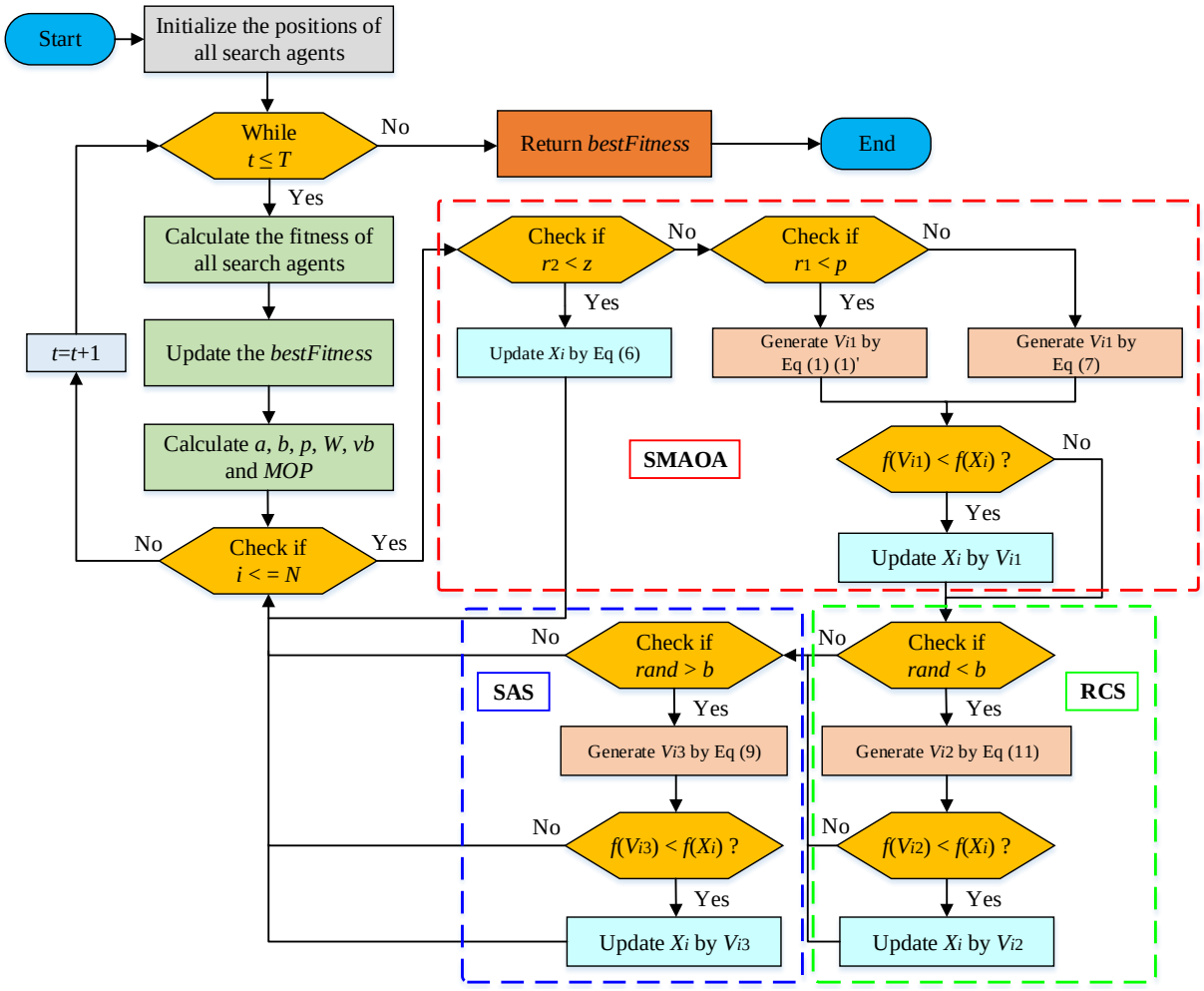

Figure 1. Flowchart of the proposed DESMAOA.

3.5. The Computational Complexity of DESMAOA

The computational complexity of DESMAOA depends on the population size (N), dimension size (D), and maximum iterations (T). First, the computational complexity of initialization is $O(N \times D)$. Then, in the iterations, the computational complexity of calculating the fitness values of all search agents is $O(N)$. The computational complexity of sorting is $O(N \times \log N)$. Moreover, the computational complexity of updating the positions of search agents in SMAOA is $O(N \times D)$. Considering the worst cases, the computational complexity of RCS and SAS is $O(2N \times D)$. In summary, the final computational complexity of the DESMAOA is $O(N \times D + T \times N(1 + \log N + 3D))$.

Algorithm 3. Pseudo-code of DESMAOA

Initialize the parameters popsize (N) and maximum iterations (T)
Initialize the positions of all search agents X_i (i = 1, 2, . . . , N)
Set the parameters α, μ, Min, and Max
While ($t \leq T$)
Calculate the fitness of all search agents
Update $bestFitness$, X_b
Calculate a, b, p, and W by Equation (2)–(5)
Calculate the MOP by Equation (8)
Update vb
For each search agent
If $r_2 < z$
Update position by Equation (6)
Else
If $r_1 < p$
Update position V_{i1} by Equation (1) (1)'
Else
Update position V_{i1} by Equation (7)
End if
If $f(V_{i1}) < f(X_i)$
$X_i = V_{i1}$
End if
If $rand < b$
Apply RCS and generate candidate position V_{i2} by Equation (11)
If $f(V_{i2}) < f(X_i)$
$X_i = V_{i2}$
End if
End if
If $rand > b$
Apply SAS and generate candidate position V_{i3} by Equation (9)
If $f(V_{i3}) < f(X_i)$
$X_i = V_{i3}$
End if
End if
End if
End for
$t = t + 1$
End While
Return $bestFitness$, X_b

4. Experimental Results and Discussions

In this section, we provide the results of a series of comparative experiments that were conducted by using 23 classical benchmark functions and 10 IEEE CEC2021 single objective optimization functions to evaluate the performance of proposed DESMAOA [36,37]. Table 2 lists the detailed parameter values of these test functions. It can be seen that these classical test functions included unimodal functions (F1–F7), multimodal functions (F8–F13), and also fixed-dimension multimodal functions (F14–F23). Moreover, the CEC2021 test functions contained four types of functions: unimodal function, basic functions, hybrid functions, and composition functions. The unimodal functions are suitable for testing the exploitation capability of algorithms, while the other types of test functions that contain a large number of local minimas can reveal the exploration capability and stability of algorithms.

In the experiments of test functions, the impacts of two applied strategies were firstly analyzed by using the classical test functions. Then, the test results of DESMAOA in classical test functions were compared with seven well-known algorithms. Multiple aspects of the analysis including exploitation capability, exploration capability, qualitative analysis, and convergence behavior are described. Moreover, the results of CEC2021 test functions were also analyzed to investigate the performance of proposed algorithm.

Table 2. Benchmark function properties (*D* indicates dimension).

Function Type	Function	Dimension	Range	Theoretical Optimization Value
Unimodal test functions	F1	30, 50, 200, 1000	[−100, 100]	0
	F2	30, 50, 200, 1000	[−10, 10]	0
	F3	30, 50, 200, 1000	[−100, 100]	0
	F4	30, 50, 200, 1000	[−100, 100]	0
	F5	30, 50, 200, 1000	[−30, 30]	0
	F6	30, 50, 200, 1000	[−100, 100]	0
	F7	30, 50, 200, 1000	[−1.28, 1.28]	0
Multimodal test functions	F8	30, 50, 200, 1000	[−500, 500]	$-418.9829 \times D$
	F9	30, 50, 200, 1000	[−5.12, 5.12]	0
	F10	30, 50, 200, 1000	[−32, 32]	0
	F11	30, 50, 200, 1000	[−600, 600]	0
	F12	30, 50, 200, 1000	[−50, 50]	0
	F13	30, 50, 200, 1000	[−50, 50]	0
Fixed-dimension multimodal test functions	F14	2	[−65, 65]	0.998004
	F15	4	[−5, 5]	0.0003075
	F16	2	[−5, 5]	−1.03163
	F17	2	[−5, 5]	0.398
	F18	2	[−2, 2]	3
	F19	3	[−1, 2]	−3.8628
	F20	6	[0, 1]	−3.3220
	F21	4	[0, 10]	−10.1532
	F22	4	[0, 10]	−10.4028
	F23	4	[0, 10]	−10.5363
CEC2021 unimodal test functions	CEC_01	10	[−100, 100]	100
CEC2021 basic test functions	CEC_02	10	[−100, 100]	1100
	CEC_03	10	[−100, 100]	700
	CEC_04	10	[−100, 100]	1900
CEC2021 hybrid test functions	CEC_05	10	[−100, 100]	1700
	CEC_06	10	[−100, 100]	1600
	CEC_07	10	[−100, 100]	2100
CEC2021 composition test functions	CEC_08	10	[−100, 100]	2200
	CEC_09	10	[−100, 100]	2400
	CEC_10	10	[−100, 100]	2500

4.1. Impacts of Components

The impacts of different versions are investigated in this section. SMA showed very outstanding performance in optimization problems. However, it still had the problems of premature convergence and local optima. According to the works of Abualigah [34], AOA shows powerful global exploration and local exploitation capability. Hence, we first hybridized the SMA and AOA to obtain the SMAOA. Then, in order to help the search agent jump out of local optima, we integrated the RCS into SMAOA. Moreover, SAS was introduced into SMAOA to improve the local search capability. Different combinations between SMAOA and two strategies are listed below:

- SMAOA;
- SMAOA combined with RCS (SMAOA1);
- SMAOA combined with SAS (SMAOA2);
- SMAOA combined with RCS and SAS (DESMAOA).

For impartial comparison, the number of iterations and population size for all tests were set as 500 and 30, respectively. Moreover, we conducted independent tests 30 times for each algorithm. The averages and standard deviations were utilized for analysis and comparison between these algorithms. The results are listed in Table 3. Note that the dimension of F1–F13 was set to 30.

Table 3. Comparison of the SMAOA, SMAOA1, SMAOA2, and DESMAOA.

Function	SMAOA		SMAOA1		SMAOA2		DESMAOA	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F1	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
F2	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
F3	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
F4	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
F5	2.82×10^{0}	7.00×10^{0}	5.85×10^{-1}	1.07×10^{0}	2.46×10^{-1}	1.33×10^{0}	1.17×10^{-3}	1.55×10^{-3}
F6	2.44×10^{-2}	2.34×10^{-2}	7.77×10^{-3}	1.21×10^{-2}	5.80×10^{-6}	1.92×10^{-6}	4.95×10^{-6}	2.01×10^{-6}
F7	1.16×10^{-4}	9.72×10^{-5}	5.67×10^{-5}	5.61×10^{-5}	6.72×10^{-5}	6.77×10^{-5}	4.27×10^{-5}	4.76×10^{-5}
F8	−12,569.2361	1.89×10^{-1}	−12,569.3229	1.38×10^{-1}	−12,569.4866	4.96×10^{-6}	−12,569.4866	4.11×10^{-6}
F9	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}
F10	8.8818×10^{-16}	0.00×10^{0}	8.8818×10^{-16}	0.00×10^{0}	8.8818×10^{-16}	0.00×10^{0}	8.8818×10^{-16}	0.00×10^{0}
F11	2.13×10^{-1}	2.92×10^{-1}	0.00×10^{0}	0.00×10^{0}	8.85×10^{-3}	2.30×10^{-2}	0.00×10^{0}	0.00×10^{0}
F12	2.63×10^{-3}	4.65×10^{-3}	5.08×10^{-5}	8.33×10^{-5}	4.84×10^{-8}	7.54×10^{-8}	1.09×10^{-7}	1.59×10^{-7}
F13	1.70×10^{-2}	3.67×10^{-2}	8.82×10^{-4}	1.15×10^{-3}	2.50×10^{-3}	6.04×10^{-3}	5.91×10^{-7}	1.06×10^{-6}
F14	9.98×10^{-1}	2.60×10^{-11}	9.98×10^{-1}	9.58×10^{-12}	9.98×10^{-1}	9.91×10^{-16}	9.98×10^{-1}	8.05×10^{-16}
F15	4.16×10^{-4}	1.56×10^{-4}	3.63×10^{-4}	9.61×10^{-5}	4.07×10^{-4}	1.90×10^{-4}	3.34×10^{-4}	8.63×10^{-5}
F16	-1.0316×10^{0}	3.97×10^{-8}	-1.0316×10^{0}	9.46×10^{-8}	-1.0316×10^{0}	1.67×10^{-11}	-1.0316×10^{0}	1.87×10^{-11}
F17	3.9789×10^{-1}	6.82×10^{-7}	3.9789×10^{-1}	3.97×10^{-7}	3.9789×10^{-1}	5.63×10^{-12}	3.9789×10^{-1}	5.94×10^{-12}
F18	3.00×10^{0}	2.79×10^{-9}	3.00×10^{0}	6.69×10^{-10}	3.00×10^{0}	8.56×10^{-11}	3.00×10^{0}	9.42×10^{-11}
F19	-3.8627×10^{0}	4.32×10^{-5}	-3.8628×10^{0}	4.68×10^{-5}	-3.8628×10^{0}	5.61×10^{-5}	-3.8627×10^{0}	7.91×10^{-5}
F20	-3.25×10^{0}	5.98×10^{-2}	-3.2859×10^{0}	5.59×10^{-2}	-3.2583×10^{0}	6.06×10^{-2}	-3.286×10^{0}	5.59×10^{-2}
F21	-1.01528×10^{1}	4.26×10^{-4}	-1.01529×10^{1}	3.67×10^{-4}	-1.01531×10^{1}	9.07×10^{-5}	-1.01531×10^{1}	1.42×10^{-4}
F22	-1.04023×10^{1}	4.61×10^{-4}	-1.04025×10^{1}	4.47×10^{-4}	-1.04028×10^{1}	8.31×10^{-5}	-1.04028×10^{1}	7.38×10^{-5}
F23	-1.0536×10^{1}	3.47×10^{-4}	-1.05362×10^{1}	2.47×10^{-4}	-1.05363×10^{1}	9.44×10^{-5}	-1.05363×10^{1}	8.26×10^{-5}

From Table 3, it can be seen that these four improved algorithms could obtain the same optimal fitness in F1–F4, F9, F10, F14, and F16–F18. In particular, the theoretical optimization values were obtained in F1–F4, F9, and F18. Compared to SMAOA, SMAOA1, and SMAOA2, DESMAOA won in F5–F8, F11, F13, F15, and F20–F24. This demonstrates that the significant effect with the combination of RCS and SAS. In addition, it is worth mentioning here that the results of DESMAOA in F12 and F19 were very close to the best ones. From the values of standard deviations, it was also shown that DESMAOA had good stability and strong robustness in solving these test functions.

4.2. The Classical Benchmark Functions

This section outlines the 23 classical test functions that were employed for experiments. The performance of DESMAOA was compared with two newly proposed algorithms (SMA and AOA) and another five very famous optimization algorithms (GWO [38], WOA [39], SSA [40], MVO [5], and PSO [12]). Table 4 lists the main parameter values used in each algorithm. Note that the parameter values used in DESMAOA were the same as those used in two original algorithms. Therefore, the stable performance could be guaranteed to some extent for the proposed algorithm. In addition, the test conditions were the same as previously for equal comparison.

Table 4. Parameter values for the optimization algorithms.

Algorithm	Parameter Settings
DESMAOA	$z = 0.03$; $\alpha = 5$; $\mu = 0.499$
SMA [33]	$z = 0.03$
AOA [34]	$\alpha = 5$; $\mu = 0.499$; $Min = 0.2$; $Max = 1$
GWO [38]	$a = [2, 0]$
WOA [39]	$a_1 = [2, 0]$; $a_2 = [-2, -1]$; $b = 1$
SSA [40]	$c_1 \in [0, 1]$; $c_2 \in [0, 1]$
MVO [5]	$WEP \in [0.2, 1]$; $TDR \in [0, 1]$; $r_1, r_2, r_3 \in [0, 1]$
PSO [12]	$c_1 = 2$; $c_2 = 2$; $W \in [0.2, 0.9]$; $vMax = 6$

4.2.1. Exploration and Exploitation Capability Analysis

Table 5 lists the experimental results of these algorithms. It was shown that the performance of DESMAOA is not only better than the original SMA and AOA but also superior to other comparative algorithms on 20 out of 23 benchmark functions. In F1–F5, F7–F15, F22, and F23, DESMAOA had the lowest average values and stand deviations. This reveals that DESMAOA possesses very good stability and also can find the optimal solution. It is worth noting that the proposed algorithm can obtain the theoretical optimization values in test functions F1–F4, F9, F11, and F18. In F6, F19, and F20, the results of DESMAOA were very close to the best ones. Therefore, these results demonstrated the remarkable effect of the proposed hybrid method. With the help of RCS, the proposed algorithm can jump out of the local minima and obtain the global optimal solution. In the meantime, high precision results could be obtained by using SAS.

In addition, the Wilcoxon signed-rank test was utilized to confirm the statistical superiority of DESMAOA [41], which revealing the statistical differences between two algorithms. The results are given in Table 6. On the basis of these results and the results in Table 5, DESMAOA outperformed SMA for 15 benchmark functions (except F1, F3, F7, F9, F10, F11, F15, and F20) and AOA for 20 benchmark functions (except F7, F15, and F17). Moreover, DESMAOA was found to be better than other comparative algorithms in most of the functions. Furthermore, the results of test functions were also evaluated using the Friedman ranking test [42], which can reveal the overall performance ranking of the comparative algorithms to the test functions. As can be seen from Figure 2, the proposed DESMAOA achieved the first rank among these algorithms. In summary, DESMAOA had excellent optimization performance that was significantly better than SMA and AOA.

Table 5. The result statistics of benchmark functions for the DESMAOA and competitor algorithms.

Function	Metric	DESMAOA	SMA	AOA	GWO	WOA	SSA	MVO	PSO
F1	Mean	0.00×10^{0}	9.93×10^{-302}	5.37×10^{-6}	7.21×10^{-28}	2.42×10^{-73}	3.96×10^{-7}	1.34×10^{0}	1.76×10^{-4}
	Std	0.00×10^{0}	0.00×10^{0}	2.14×10^{-6}	1.17×10^{-27}	8.81×10^{-73}	9.50×10^{-7}	5.38×10^{-1}	1.82×10^{-4}
F2	Mean	0.00×10^{0}	5.05×10^{-138}	1.74×10^{-3}	8.26×10^{-17}	8.81×10^{-52}	2.07×10^{0}	2.20×10^{0}	7.05×10^{0}
	Std	0.00×10^{0}	2.77×10^{-137}	2.08×10^{-3}	6.54×10^{-17}	2.46×10^{-51}	1.33×10^{0}	7.31×10^{0}	7.01×10^{0}
F3	Mean	0.00×10^{0}	5.43×10^{-323}	1.24×10^{-3}	1.55×10^{-5}	4.41×10^{4}	1.66×10^{3}	2.04×10^{2}	7.93×10^{1}
	Std	0.00×10^{0}	0.00×10^{0}	8.14×10^{-4}	3.50×10^{-5}	1.08×10^{4}	9.24×10^{2}	6.63×10^{1}	2.57×10^{1}
F4	Mean	0.00×10^{0}	7.56×10^{-154}	1.53×10^{-2}	8.03×10^{-7}	4.68×10^{1}	1.15×10^{1}	2.16×10^{0}	1.12×10^{0}
	Std	0.00×10^{0}	4.14×10^{-153}	1.06×10^{-2}	6.71×10^{-7}	2.77×10^{1}	4.04×10^{0}	8.66×10^{-1}	2.40×10^{-1}
F5	Mean	1.17×10^{-3}	8.56×10^{0}	2.79×10^{1}	2.71×10^{1}	2.82×10^{1}	2.90×10^{2}	7.89×10^{2}	8.16×10^{1}
	Std	1.55×10^{-3}	1.21×10^{1}	3.01×10^{-1}	8.49×10^{-1}	4.97×10^{-1}	4.77×10^{2}	8.74×10^{2}	7.03×10^{1}
F6	Mean	4.95×10^{-6}	5.74×10^{-3}	3.06×10^{0}	7.58×10^{-1}	3.72×10^{-1}	1.78×10^{-7}	1.34×10^{0}	1.37×10^{-4}
	Std	2.01×10^{-6}	3.38×10^{-3}	2.69×10^{-1}	4.94×10^{-1}	2.18×10^{-1}	1.51×10^{-7}	3.43×10^{-1}	1.65×10^{-4}
F7	Mean	4.27×10^{-5}	1.24×10^{-4}	6.74×10^{-5}	1.69×10^{-3}	3.15×10^{-3}	1.73×10^{-1}	3.21×10^{-2}	2.55×10^{0}
	Std	4.76×10^{-5}	1.07×10^{-4}	7.11×10^{-5}	8.95×10^{-4}	3.61×10^{-3}	5.61×10^{-2}	1.32×10^{-2}	4.54×10^{0}
F8	Mean	−12,569.4866	−12,569.1799	-5.48×10^{3}	-6.01×10^{3}	-1.06×10^{4}	-7.47×10^{3}	-7.55×10^{3}	-4.69×10^{3}
	Std	4.11×10^{-6}	2.66×10^{-1}	3.69×10^{2}	6.42×10^{2}	1.69×10^{3}	8.76×10^{2}	6.27×10^{2}	1.21×10^{3}
F9	Mean	0.00×10^{0}	0.00×10^{0}	1.66×10^{-6}	2.26×10^{0}	3.79×10^{-15}	5.53×10^{1}	1.20×10^{2}	1.02×10^{2}
	Std	0.00×10^{0}	0.00×10^{0}	1.27×10^{-6}	3.27×10^{0}	2.08×10^{-14}	1.83×10^{1}	3.29×10^{1}	3.19×10^{1}
F10	Mean	8.8818×10^{-16}	8.8818×10^{-16}	4.36×10^{-4}	1.01×10^{-13}	3.85×10^{-15}	2.56×10^{0}	2.03×10^{0}	1.69×10^{-2}
	Std	0.00×10^{0}	0.00×10^{0}	1.62×10^{-4}	1.81×10^{-14}	2.10×10^{-15}	6.94×10^{-1}	5.47×10^{-1}	1.30×10^{-2}
F11	Mean	0.00×10^{0}	0.00×10^{0}	8.42×10^{-4}	6.19×10^{-3}	1.68×10^{-2}	1.88×10^{-2}	8.60×10^{-1}	4.39×10^{-3}
	Std	0.00×10^{0}	0.00×10^{0}	3.12×10^{-3}	8.92×10^{-3}	6.38×10^{-2}	1.46×10^{-2}	8.21×10^{-2}	6.85×10^{-3}
F12	Mean	1.09×10^{-7}	5.81×10^{-3}	7.44×10^{-1}	4.42×10^{-2}	2.81×10^{-2}	7.54×10^{0}	2.43×10^{0}	2.07×10^{-2}
	Std	1.59×10^{-7}	6.50×10^{-3}	3.03×10^{-2}	1.86×10^{-2}	2.18×10^{-2}	3.43×10^{0}	1.39×10^{0}	4.22×10^{-2}
F13	Mean	5.91×10^{-7}	6.35×10^{-3}	2.96×10^{0}	6.94×10^{-1}	6.36×10^{-1}	1.38×10^{1}	1.96×10^{-1}	5.55×10^{-3}
	Std	1.06×10^{-6}	7.05×10^{-3}	1.03×10^{-2}	2.45×10^{-1}	3.53×10^{-1}	1.10×10^{1}	1.26×10^{-1}	9.01×10^{-3}
F14	Mean	9.98×10^{-1}	9.98×10^{-1}	9.87×10^{0}	4.16×10^{0}	2.54×10^{0}	1.36×10^{0}	9.98×10^{-1}	2.97×10^{0}
	Std	8.05×10^{-16}	3.93×10^{-13}	3.89×10^{0}	4.28×10^{0}	2.91×10^{0}	8.82×10^{-1}	4.31×10^{-11}	2.55×10^{0}
F15	Mean	3.34×10^{-4}	4.84×10^{-4}	8.39×10^{-3}	3.15×10^{-3}	8.25×10^{-4}	2.91×10^{-3}	5.24×10^{-3}	7.22×10^{-3}
	Std	8.63×10^{-5}	2.15×10^{-4}	1.29×10^{-2}	6.88×10^{-3}	5.40×10^{-4}	5.93×10^{-3}	1.26×10^{-2}	9.03×10^{-3}
F16	Mean	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}	-1.0316×10^{0}
	Std	1.87×10^{-11}	8.36×10^{-10}	2.28×10^{-11}	3.13×10^{-8}	1.68×10^{-9}	3.89×10^{-14}	4.19×10^{-7}	6.25×10^{-16}
F17	Mean	3.9789×10^{-1}	3.9789×10^{-1}	4.0217×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}	3.9789×10^{-1}
	Std	5.94×10^{-12}	5.40×10^{-9}	1.52×10^{-2}	8.10×10^{-7}	6.71×10^{-6}	7.99×10^{-15}	1.27×10^{-7}	0.00×10^{0}
F18	Mean	3.0000×10^{0}	3.0000×10^{0}	4.8000×10^{0}	5.7000×10^{0}	3.0001×10^{0}	3.0000×10^{0}	3.0000×10^{0}	3.0000×10^{0}
	Std	9.42×10^{-11}	1.17×10^{-9}	6.85×10^{0}	1.48×10^{1}	8.14×10^{-5}	2.13×10^{-13}	3.49×10^{-6}	1.79×10^{-15}
F19	Mean	-3.8627×10^{0}	-3.8628×10^{0}	-3.8627×10^{0}	-3.8605×10^{0}	-3.8572×10^{0}	-3.8628×10^{0}	-3.8628×10^{0}	-3.8628×10^{0}
	Std	7.91×10^{-5}	1.58×10^{-7}	2.62×10^{-4}	4.08×10^{-3}	1.02×10^{-2}	1.17×10^{-12}	7.73×10^{-6}	2.58×10^{-15}
F20	Mean	-3.286×10^{0}	-3.2503×10^{0}	-3.2942×10^{0}	-3.2339×10^{0}	-3.2225×10^{0}	-3.2255×10^{0}	-3.2454×10^{0}	-3.2402×10^{0}
	Std	5.59×10^{-2}	5.95×10^{-2}	5.12×10^{-2}	7.30×10^{-2}	1.10×10^{-1}	5.45×10^{-2}	5.93×10^{-2}	8.13×10^{-2}
F21	Mean	-1.01531×10^{1}	-1.01531×10^{1}	-7.8781×10^{0}	-8.8066×10^{0}	-8.4438×10^{0}	-6.9755×10^{0}	-7.048×10^{0}	-6.3883×10^{0}
	Std	1.42×10^{-4}	1.05×10^{-4}	2.68×10^{0}	2.54×10^{0}	2.44×10^{0}	3.35×10^{0}	3.28×10^{0}	3.27×10^{0}
F22	Mean	-1.04028×10^{1}	-1.04028×10^{1}	-7.2814×10^{0}	-1.02239×10^{1}	-7.0271×10^{0}	-8.938×10^{0}	-9.0327×10^{0}	-8.71250×10^{0}
	Std	7.38×10^{-5}	1.82×10^{-4}	3.52×10^{0}	9.70×10^{-1}	3.08×10^{0}	2.99×10^{0}	2.83×10^{0}	2.91×10^{0}
F23	Mean	-1.05363×10^{1}	-1.05363×10^{1}	-6.6743×10^{0}	-1.05349×10^{1}	-7.7815×10^{0}	-8.1138×10^{0}	-8.5201×10^{0}	-9.1233×10^{0}
	Std	8.26×10^{-5}	9.71×10^{-5}	3.31×10^{0}	8.48×10^{-4}	3.28×10^{0}	3.51×10^{0}	3.20×10^{0}	2.93×10^{0}

Table 6. *p*-values of the Wilcoxon signed-rank test between DESMAOA and other competitor algorithms.

Function	DESMAOA vs. SMA	DESMAOA vs. AOA	DESMAOA vs. GWO	DESMAOA vs. WOA	DESMAOA vs. SSA	DESMAOA vs. MVO	DESMAOA vs. PSO
F1	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F2	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F3	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F4	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F5	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F6	1.22×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	8.54×10^{-4}
F7	2.52×10^{-1}	1.88×10^{-1}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F8	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F9	1.00×10^{0}	1.22×10^{-4}	6.10×10^{-5}	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F10	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	9.77×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F11	1.00×10^{0}	6.10×10^{-5}	2.50×10^{-1}	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F12	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	8.36×10^{-3}
F13	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	1.22×10^{-4}
F14	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	8.14×10^{-2}	6.10×10^{-5}	7.93×10^{-3}
F15	3.30×10^{-1}	5.54×10^{-2}	4.54×10^{-1}	5.54×10^{-2}	6.10×10^{-5}	1.16×10^{-3}	1.22×10^{-4}
F16	2.56×10^{-2}	3.36×10^{-3}	6.10×10^{-5}	2.52×10^{-1}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F17	6.10×10^{-4}	6.39×10^{-1}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F18	1.81×10^{-2}	7.62×10^{-1}	8.36×10^{-3}	8.36×10^{-3}	6.10×10^{-5}	8.36×10^{-3}	6.10×10^{-5}
F19	6.10×10^{-5}	1.03×10^{-2}	2.56×10^{-2}	6.10×10^{-5}	6.10×10^{-5}	4.27×10^{-3}	6.10×10^{-5}
F20	6.39×10^{-1}	1.81×10^{-2}	2.52×10^{-1}	2.52×10^{-1}	1.51×10^{-2}	5.99×10^{-1}	2.08×10^{-1}
F21	3.05×10^{-4}	3.02×10^{-2}	6.10×10^{-5}	1.22×10^{-4}	1.88×10^{-1}	1.53×10^{-3}	8.04×10^{-1}
F22	6.10×10^{-5}	4.27×10^{-3}	6.10×10^{-4}	1.22×10^{-4}	8.04×10^{-1}	3.03×10^{-1}	7.62×10^{-1}
F23	6.10×10^{-4}	1.21×10^{-1}	1.22×10^{-4}	6.10×10^{-5}	3.30×10^{-1}	6.79×10^{-1}	6.79×10^{-1}

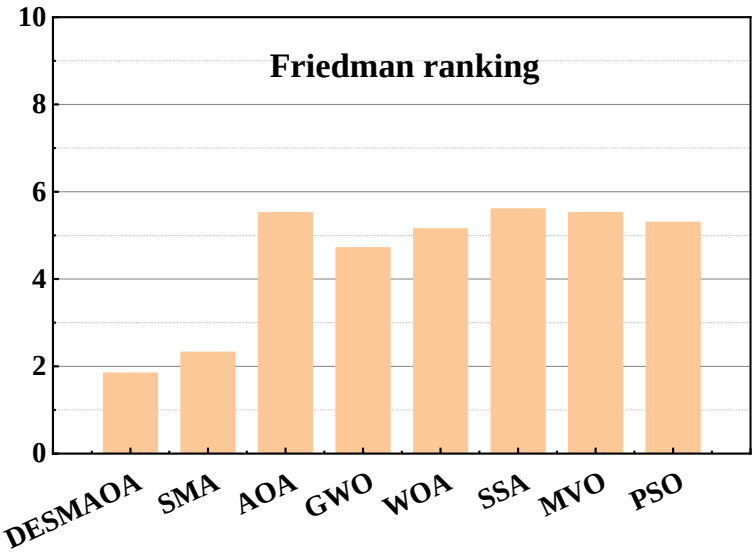

Figure 2. Average Friedman ranking values of DESMAOA and other comparative algorithms on 30 dimensions.

4.2.2. Qualitative Analysis

Figure 3 shows the qualitative results of proposed algorithm in F4, F5, F6, F8, F12, F13, F15, and F21. From the scatter plot of the search history, we were able to see that search agents were distributed in the whole search space in the early stage. During the iteration progresses, they concentrated in a quick time. The density of distribution for different functions indicates that DESMAOA had balanced performance between exploration and exploitation. Moreover, some sudden changes in the amplitude were observed clearly

in the trajectory of the first search agent, which revealed that DESMAOA had strong exploration capability over the course of iterations when handing these test functions. The drastic fluctuation of average fitness also showed that DESMAOA can jump out of local optima and explore more spaces when dealing with different types of optimization issues. Hence, the local optimal solution can be avoided effectively. Finally, the DESMAOA was able to find better solutions in most of the functions compared with SMA and AOA, which demonstrates the effectiveness of the proposed method.

4.2.3. Analysis of Convergence Behavior

It is important to study the convergence behavior of optimization algorithms when they are searching for the optimal solution. In general, fast convergence speed is required in the early exploration, which implies the algorithm has powerful exploration capability. On the other hand, local optima also should be avoided, which can be seen from the convergence curve. Figure 4 shows the convergence curves of DESMAOA and other compared algorithms on 30 dimensions. Some benchmark functions are used for analysis including F4, F5, F6, F8, F12, F13, F15, and F21. From these functions, it can be seen that the initial convergence speed of DESMAOA is the fastest in most cases. In Figure 5, step-like or cliff-like declines in the convergence curves of DESMAOA can be observed. This suggests that the DESMAOA has a prominent exploration capability. From F5, F6, F12, and F13, the precision of solutions for DESMAOA is further improved with the help of SAS during the iteration. In sum, DESMAOA achieved the best solutions in these functions.

4.2.4. Scalability Test

The performance fluctuations of optimization algorithms can be revealed according to the scalability test. In this work, the performance of DESMAOA in different dimensions (D = 50, 200, 1000) were also tested. It is easy to understand that the higher dimension will make it harder for the algorithm to find the global optimal solution. Note that only F1–F13 in the 23 benchmark functions were selected for this test. As mentioned previously, F1–F7 are single-mode functions that only have one locally optimal solution. In contrast, F8–F13 are multimode functions that have many locally optimal solutions. Moreover, the experimental parameters were kept the same as previous experiments. Tables 7 and 8 show the results of DESMAOA and other algorithms in different dimensions.

Both results of unimodal and multimode functions indicated that DESMAOA had excellent performance in the conditions of high dimensions. Compared with SMA, AOA, and other well-known algorithms, DESMAOA was the first in all functions except F7. In F7, AOA became better and had more stable results in different dimensions. It is also noted that these comparative algorithms (GWO, WOA, SSA, MVO, and PSO) presented poor optimization capability in some cases, especially in higher dimensions. Furthermore, the Wilcoxon signed-rank test and Friedman ranking test were utilized to analyze the differences between DESMAOA and other algorithms, as listed in Tables 9–12. From Tables 9–11, it can be seen that DESMAOA had significant differences compared with these comparative algorithms. Moreover, in Table 12, the proposed DESMAOA ranked first compared to other algorithms in different dimensions. It is noted that the distance between the first and second was evident. In summary, the proposed DESMAOA had better optimization behavior and stability in dealing with high-dimensional problems.

4.3. The IEEE CEC2021 Standard Test Functions

This section describes the IEEE CEC2021 test functions that were employed to further analyze the performance of proposed DESMAOA on solving global optimization problems. The comparative algorithms included the SMA, AOA, GWO, WOA, SSA, MVO, and PSO. To achieve the statistical results, 30 repeated independent tests were conducted for each function. The experimental results are given in Table 13.

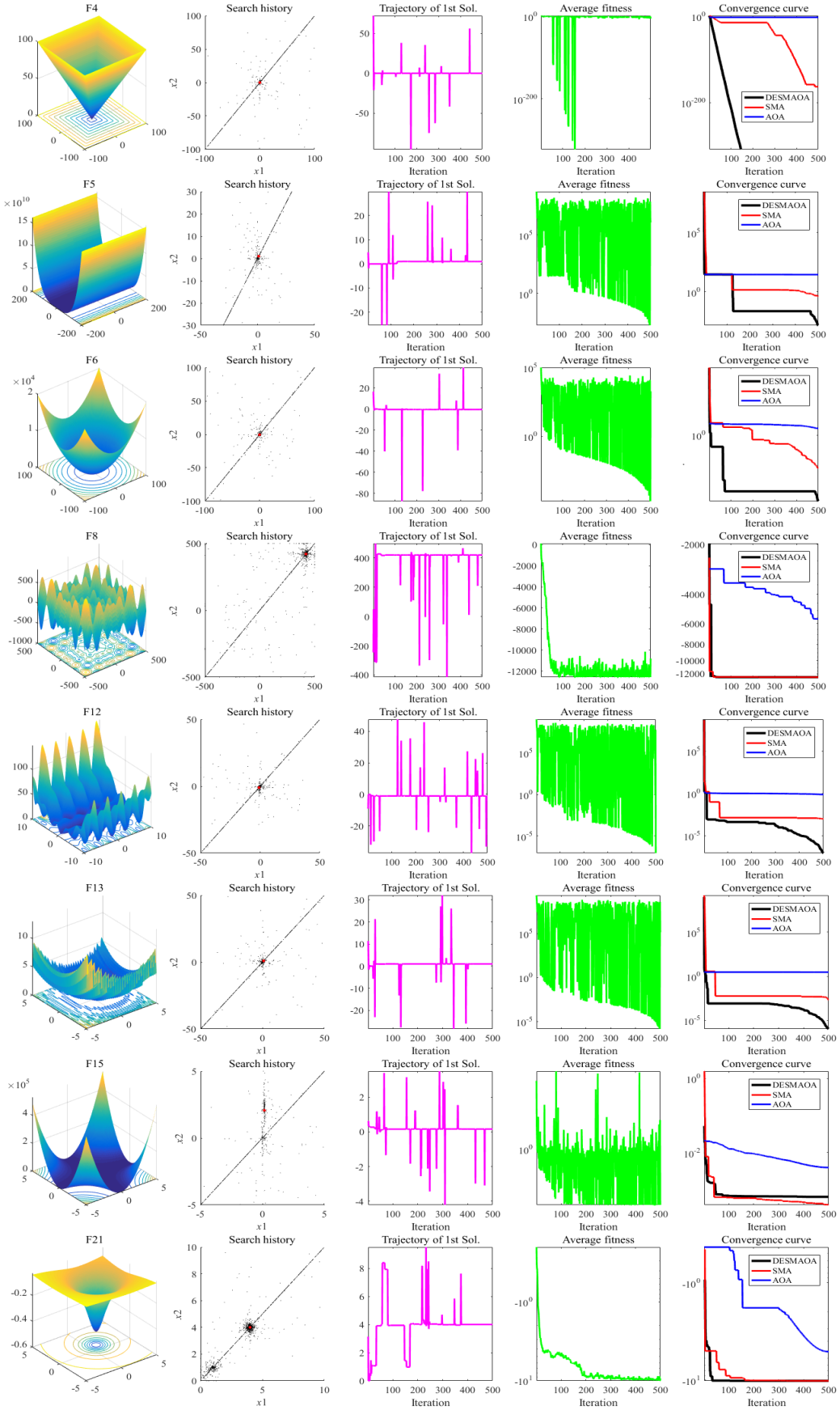

Figure 3. Qualitative results for the benchmark functions F4, F5, F6, F8, F12, F13, F15, and F21.

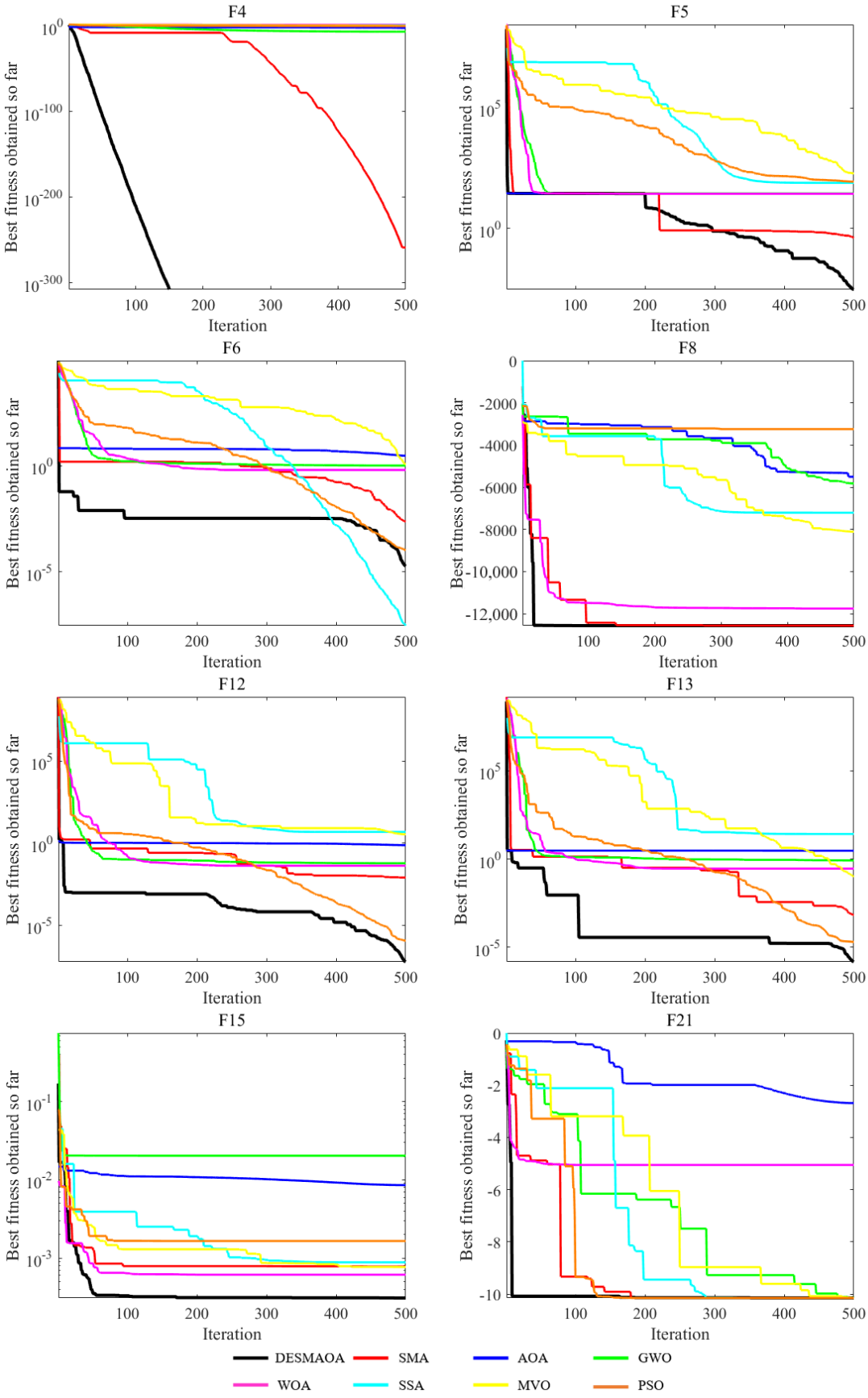

Figure 4. The convergence curves of F4, F5, F6, F8, F12, F13, F15, and F21.

Table 7. Unimodal benchmark function result statistics of the DESMAOA and competitor algorithms in different dimensions.

Function	*D*	Metric	DESMAOA	SMA	AOA	GWO	WOA	SSA	MVO	PSO
F1	50	Mean	0.00×10^{0}	3.94×10^{-310}	4.05×10^{-5}	5.96×10^{-20}	3.97×10^{-71}	6.27×10^{-1}	9.07×10^{0}	2.05×10^{-1}
		Std	0.00×10^{0}	0.00×10^{0}	1.33×10^{-5}	5.86×10^{-20}	2.17×10^{-70}	5.32×10^{-1}	2.48×10^{0}	1.82×10^{-1}
	200	Mean	0.00×10^{0}	5.15×10^{-244}	4.63×10^{-2}	1.09×10^{-7}	2.33×10^{-71}	1.76×10^{4}	2.84×10^{3}	3.31×10^{2}
		Std	0.00×10^{0}	0.00×10^{0}	1.24×10^{-2}	7.08×10^{-8}	9.24×10^{-71}	1.60×10^{3}	3.15×10^{2}	4.11×10^{1}
	1000	Mean	0.00×10^{0}	2.20×10^{-246}	1.50×10^{0}	2.53×10^{-1}	3.57×10^{-68}	2.29×10^{5}	7.94×10^{5}	4.11×10^{4}
		Std	0.00×10^{0}	0.00×10^{0}	4.79×10^{-2}	5.65×10^{-2}	1.95×10^{-67}	1.19×10^{4}	2.70×10^{4}	2.32×10^{3}
F2	50	Mean	0.00×10^{0}	1.50×10^{-145}	6.70×10^{-3}	2.60×10^{-12}	1.56×10^{-49}	9.29×10^{0}	3.50×10^{3}	2.58×10^{1}
		Std	0.00×10^{0}	8.24×10^{-145}	3.05×10^{-3}	1.52×10^{-12}	7.19×10^{-49}	3.59×10^{0}	1.73×10^{4}	1.89×10^{1}
	200	Mean	0.00×10^{0}	7.90×10^{-138}	7.23×10^{-2}	3.25×10^{-5}	2.93×10^{-48}	1.55×10^{2}	5.08×10^{77}	4.66×10^{2}
		Std	0.00×10^{0}	3.91×10^{-137}	1.18×10^{-2}	7.69×10^{-6}	1.17×10^{-47}	1.44×10^{1}	2.73×10^{78}	6.40×10^{1}
	1000	Mean	0.00×10^{0}	5.92×10^{-1}	1.58×10^{0}	6.78×10^{-1}	1.44×10^{-47}	1.19×10^{3}	3.59×10^{278}	1.41×10^{3}
		Std	0.00×10^{0}	2.92×10^{0}	1.08×10^{-1}	5.77×10^{-1}	7.74×10^{-47}	2.48×10^{1}	Inf	6.51×10^{1}
F3	50	Mean	0.00×10^{0}	1.03×10^{-293}	1.81×10^{-2}	3.84×10^{-1}	2.01×10^{5}	9.10×10^{3}	6.50×10^{3}	1.48×10^{3}
		Std	0.00×10^{0}	0.00×10^{0}	8.60×10^{-3}	1.01×10^{0}	4.50×10^{4}	4.69×10^{3}	1.95×10^{3}	4.64×10^{2}
	200	Mean	0.00×10^{0}	3.94×10^{-219}	7.32×10^{-1}	1.98×10^{4}	4.55×10^{6}	2.05×10^{5}	3.16×10^{5}	8.34×10^{4}
		Std	0.00×10^{0}	0.00×10^{0}	1.86×10^{-1}	9.23×10^{3}	1.51×10^{6}	7.25×10^{4}	3.10×10^{4}	2.24×10^{4}
	1000	Mean	0.00×10^{0}	4.81×10^{-125}	3.35×10^{1}	1.53×10^{6}	1.36×10^{8}	5.19×10^{6}	7.98×10^{6}	2.27×10^{6}
		Std	0.00×10^{0}	2.64×10^{-124}	6.49×10^{0}	3.16×10^{5}	6.32×10^{7}	2.46×10^{6}	8.76×10^{5}	4.56×10^{5}
F4	50	Mean	0.00×10^{0}	3.80×10^{-158}	3.56×10^{-2}	7.25×10^{-4}	7.35×10^{1}	1.94×10^{1}	1.67×10^{1}	3.74×10^{0}
		Std	0.00×10^{0}	2.06×10^{-157}	7.14×10^{-3}	1.42×10^{-3}	1.99×10^{1}	3.14×10^{0}	4.24×10^{0}	7.18×10^{-1}
	200	Mean	0.00×10^{0}	3.11×10^{-114}	9.10×10^{-2}	2.39×10^{1}	8.41×10^{1}	3.52×10^{1}	8.32×10^{1}	1.93×10^{1}
		Std	0.00×10^{0}	1.19×10^{-113}	1.18×10^{-2}	5.51×10^{0}	1.89×10^{1}	3.49×10^{0}	3.80×10^{0}	1.45×10^{0}
	1000	Mean	0.00×10^{0}	3.86×10^{-101}	1.54×10^{-1}	7.88×10^{1}	7.94×10^{1}	4.43×10^{1}	9.81×10^{1}	3.31×10^{1}
		Std	0.00×10^{0}	1.90×10^{-100}	7.55×10^{-3}	3.25×10^{0}	2.09×10^{1}	3.19×10^{0}	6.42×10^{-1}	1.52×10^{0}
F5	50	Mean	4.84×10^{0}	1.89×10^{1}	4.83×10^{1}	4.72×10^{1}	4.83×10^{1}	1.64×10^{3}	7.66×10^{2}	4.21×10^{2}
		Std	1.47×10^{1}	1.93×10^{1}	1.40×10^{-1}	7.35×10^{-1}	4.05×10^{-1}	3.66×10^{3}	7.47×10^{2}	2.18×10^{2}
	200	Mean	1.29×10^{1}	6.23×10^{1}	1.98×10^{2}	1.98×10^{2}	1.98×10^{2}	3.79×10^{6}	3.91×10^{5}	5.98×10^{5}
		Std	3.68×10^{1}	7.20×10^{1}	7.40×10^{-2}	4.19×10^{-1}	1.65×10^{-1}	9.40×10^{5}	1.21×10^{5}	1.04×10^{5}
	1000	Mean	1.11×10^{2}	4.01×10^{2}	1.00×10^{3}	1.05×10^{3}	9.94×10^{2}	1.21×10^{8}	2.33×10^{9}	2.95×10^{8}
		Std	2.50×10^{2}	4.15×10^{2}	2.71×10^{-1}	2.55×10^{1}	1.03×10^{0}	1.12×10^{7}	1.85×10^{8}	4.28×10^{7}
F6	50	Mean	1.66×10^{-4}	8.78×10^{-2}	7.29×10^{0}	2.57×10^{0}	1.20×10^{0}	8.30×10^{-1}	9.61×10^{0}	1.97×10^{-1}
		Std	5.60×10^{-5}	6.54×10^{-2}	4.04×10^{-1}	4.02×10^{-1}	5.39×10^{-1}	7.25×10^{-1}	1.88×10^{0}	1.74×10^{-1}
	200	Mean	3.73×10^{-2}	8.26×10^{0}	3.59×10^{1}	2.91×10^{1}	1.11×10^{1}	1.76×10^{4}	2.99×10^{3}	3.21×10^{2}
		Std	3.21×10^{-2}	8.04×10^{0}	1.19×10^{0}	1.28×10^{0}	2.90×10^{0}	2.45×10^{3}	4.10×10^{2}	4.66×10^{1}
	1000	Mean	3.34×10^{0}	6.80×10^{1}	2.42×10^{2}	2.02×10^{2}	6.68×10^{1}	2.37×10^{5}	8.04×10^{5}	4.02×10^{4}
		Std	4.80×10^{0}	8.81×10^{1}	1.23×10^{0}	2.57×10^{0}	1.52×10^{1}	1.11×10^{4}	2.79×10^{4}	2.17×10^{3}
F7	50	Mean	1.36×10^{-4}	1.96×10^{-4}	6.63×10^{-5}	3.54×10^{-3}	3.97×10^{-3}	4.86×10^{-1}	1.07×10^{-1}	3.97×10^{1}
		Std	1.18×10^{-4}	1.69×10^{-4}	5.90×10^{-5}	1.90×10^{-3}	4.79×10^{-3}	1.53×10^{-1}	2.32×10^{-2}	2.76×10^{1}
	200	Mean	1.29×10^{-4}	4.32×10^{-4}	5.35×10^{-5}	1.63×10^{-2}	4.14×10^{-3}	1.72×10^{1}	5.40×10^{0}	2.95×10^{3}
		Std	1.52×10^{-4}	3.04×10^{-4}	5.09×10^{-5}	5.34×10^{-3}	4.22×10^{-3}	4.14×10^{0}	7.17×10^{-1}	4.68×10^{2}
	1000	Mean	1.17×10^{-4}	6.93×10^{-4}	8.25×10^{-5}	1.55×10^{-1}	3.29×10^{-3}	1.74×10^{3}	2.88×10^{4}	2.39×10^{5}
		Std	1.28×10^{-4}	4.85×10^{-4}	6.96×10^{-5}	3.32×10^{-2}	3.73×10^{-3}	1.75×10^{2}	2.72×10^{3}	7.66×10^{3}

Table 8. Unimodal benchmark function result statistics of the DESMAOA and competitor algorithms in different dimensions.

Function	*D*	Metric	DESMAOA	SMA	AOA	GWO	WOA	SSA	MVO	PSO
F8	50	Mean	-2.0949×10^{4}	-2.0947×10^{4}	-8.3989×10^{3}	-9.1468×10^{3}	-1.8030×10^{4}	-1.2107×10^{4}	-1.2350×10^{4}	-7.6172×10^{3}
		Std	1.54×10^{-4}	2.27×10^{0}	5.06×10^{2}	1.59×10^{3}	2.78×10^{3}	1.00×10^{3}	1.11×10^{3}	2.18×10^{3}
	200	Mean	-8.3796×10^{4}	-8.3757×10^{4}	-2.1657×10^{4}	-2.7533×10^{4}	-7.1281×10^{4}	-3.4381×10^{4}	-4.0399×10^{4}	-1.5591×10^{4}
		Std	6.62×10^{-1}	6.42×10^{1}	1.27×10^{3}	5.65×10^{3}	1.28×10^{4}	2.25×10^{3}	2.30×10^{3}	6.41×10^{3}
	1000	Mean	-4.1892×10^{5}	-4.1862×10^{5}	-5.4566×10^{4}	-8.4602×10^{4}	-3.5941×10^{5}	-8.8084×10^{4}	-1.1041×10^{5}	-3.3236×10^{4}
		Std	1.25×10^{2}	5.75×10^{2}	2.29×10^{3}	2.28×10^{4}	5.92×10^{4}	7.25×10^{3}	3.94×10^{3}	1.51×10^{4}
F9	50	Mean	0.00×10^{0}	0.00×10^{0}	1.61×10^{-5}	5.24×10^{0}	1.89×10^{-15}	8.82×10^{1}	2.54×10^{2}	2.84×10^{2}
		Std	0.00×10^{0}	0.00×10^{0}	4.42×10^{-6}	7.71×10^{0}	1.04×10^{-14}	2.32×10^{1}	5.65×10^{1}	5.01×10^{1}
	200	Mean	0.00×10^{0}	0.00×10^{0}	1.34×10^{-3}	2.41×10^{1}	7.58×10^{-15}	8.27×10^{2}	1.90×10^{3}	2.02×10^{3}
		Std	0.00×10^{0}	0.00×10^{0}	1.82×10^{-4}	9.14×10^{0}	4.15×10^{-14}	8.74×10^{1}	1.30×10^{2}	1.25×10^{2}
	1000	Mean	0.00×10^{0}	0.00×10^{0}	3.79×10^{-2}	2.06×10^{2}	0.00×10^{0}	7.63×10^{3}	1.46×10^{4}	1.41×10^{4}
		Std	0.00×10^{0}	0.00×10^{0}	1.94×10^{-3}	5.67×10^{1}	0.00×10^{0}	2.12×10^{2}	2.44×10^{2}	2.98×10^{2}
F10	50	Mean	8.8818×10^{-16}	8.8818×10^{-16}	1.14×10^{-3}	4.3720×10^{-11}	4.3225×10^{-15}	4.83×10^{0}	3.56×10^{0}	1.69×10^{0}
		Std	0.00×10^{0}	0.00×10^{0}	1.93×10^{-4}	2.44×10^{-11}	2.38×10^{-15}	1.23×10^{0}	3.13×10^{0}	5.70×10^{-1}
	200	Mean	8.8818×10^{-16}	8.8818×10^{-16}	1.06×10^{-2}	2.18×10^{-5}	4.09×10^{-15}	1.30×10^{1}	2.04×10^{1}	6.61×10^{0}
		Std	0.00×10^{0}	0.00×10^{0}	1.01×10^{-3}	6.01×10^{-6}	2.70×10^{-15}	4.61×10^{-1}	2.15×10^{-1}	3.38×10^{-1}
	1000	Mean	8.8818×10^{-16}	8.8818×10^{-16}	3.32×10^{-2}	1.89×10^{-2}	4.91×10^{-15}	1.45×10^{1}	2.10×10^{1}	1.60×10^{1}
		Std	0.00×10^{0}	0.00×10^{0}	7.69×10^{-4}	3.23×10^{-3}	2.42×10^{-15}	1.94×10^{-1}	3.27×10^{-2}	2.33×10^{-1}
F11	50	Mean	0.00×10^{0}	0.00×10^{0}	7.70×10^{-3}	2.94×10^{-3}	1.34×10^{-2}	5.55×10^{-1}	1.09×10^{0}	1.62×10^{-2}
		Std	0.00×10^{0}	0.00×10^{0}	1.91×10^{-2}	6.06×10^{-3}	5.11×10^{-2}	2.74×10^{-1}	2.30×10^{-2}	1.19×10^{-2}
	200	Mean	0.00×10^{0}	0.00×10^{0}	7.85×10^{0}	6.27×10^{-3}	0.00×10^{0}	1.46×10^{2}	2.73×10^{1}	2.28×10^{0}
		Std	0.00×10^{0}	0.00×10^{0}	1.19×10^{1}	1.48×10^{-2}	0.00×10^{0}	1.88×10^{1}	3.08×10^{0}	2.70×10^{0}
	1000	Mean	0.00×10^{0}	0.00×10^{0}	1.33×10^{4}	2.37×10^{-2}	0.00×10^{0}	2.12×10^{3}	7.25×10^{3}	2.74×10^{2}
		Std	0.00×10^{0}	0.00×10^{0}	2.64×10^{3}	3.67×10^{-2}	0.00×10^{0}	8.35×10^{1}	3.01×10^{2}	1.86×10^{1}
F12	50	Mean	6.67×10^{-7}	6.02×10^{-3}	9.06×10^{-1}	1.22×10^{-1}	3.46×10^{-2}	1.26×10^{1}	5.51×10^{0}	8.03×10^{-2}
		Std	6.28×10^{-7}	1.22×10^{-2}	2.39×10^{-2}	7.35×10^{-2}	1.86×10^{-2}	4.57×10^{0}	1.37×10^{0}	1.53×10^{-1}
	200	Mean	2.01×10^{-5}	5.76×10^{-3}	8.41×10^{-1}	5.42×10^{-1}	7.03×10^{-2}	7.55×10^{3}	2.30×10^{3}	4.84×10^{1}
		Std	1.57×10^{-5}	8.11×10^{-3}	5.60×10^{-2}	6.68×10^{-2}	3.52×10^{-2}	1.01×10^{4}	2.88×10^{3}	3.71×10^{1}
	1000	Mean	1.53×10^{-4}	9.67×10^{-3}	1.04×10^{0}	1.26×10^{0}	1.05×10^{-1}	1.16×10^{7}	4.19×10^{9}	9.21×10^{6}
		Std	2.46×10^{-4}	1.70×10^{-2}	1.12×10^{-2}	2.99×10^{-1}	5.30×10^{-2}	4.67×10^{6}	4.67×10^{8}	2.30×10^{6}
F13	50	Mean	1.08×10^{-3}	2.52×10^{-2}	4.94×10^{0}	2.03×10^{0}	1.14×10^{0}	8.07×10^{1}	7.29×10^{0}	1.84×10^{-1}
		Std	4.27×10^{-3}	3.02×10^{-2}	6.56×10^{-4}	2.80×10^{-1}	4.84×10^{-1}	1.61×10^{1}	1.15×10^{1}	1.16×10^{-1}
	200	Mean	1.85×10^{-3}	4.73×10^{-1}	1.97×10^{1}	1.67×10^{1}	6.18×10^{0}	1.61×10^{6}	1.11×10^{5}	5.27×10^{3}
		Std	1.29×10^{-3}	7.17×10^{-1}	9.97×10^{-2}	4.32×10^{-1}	1.75×10^{0}	7.60×10^{5}	1.03×10^{5}	2.58×10^{3}
	1000	Mean	6.06×10^{-2}	3.82×10^{0}	1.00×10^{2}	1.21×10^{2}	4.02×10^{1}	1.47×10^{8}	9.13×10^{9}	8.26×10^{7}
		Std	8.56×10^{-2}	3.59×10^{0}	3.49×10^{-1}	7.98×10^{0}	1.12×10^{1}	2.91×10^{7}	8.64×10^{8}	1.28×10^{7}

Table 9. *p*-values of the Wilcoxon signed-rank test between DESMAOA and other competitor algorithms on 50 dimensions.

Function	DESMAOA vs. SMA	DESMAOA vs. AOA	DESMAOA vs. GWO	DESMAOA vs. WOA	DESMAOA vs. SSA	DESMAOA vs. MVO	DESMAOA vs. PSO
F1	5.00×10^{-1}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F2	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F3	5.00×10^{-1}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F4	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F5	5.37×10^{-3}	6.10×10^{-5}	1.22×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F6	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F7	8.33×10^{-2}	7.30×10^{-2}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F8	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F9	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F10	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	2.44×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F11	1.00×10^{0}	6.10×10^{-5}	2.50×10^{-1}	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F12	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F13	2.01×10^{-3}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}

Table 10. *p*-Values of the Wilcoxon signed-rank test between DESMAOA and other competitor algorithms on 200 dimensions.

Function	DESMAOA vs. SMA	DESMAOA vs. AOA	DESMAOA vs. GWO	DESMAOA vs. WOA	DESMAOA vs. SSA	DESMAOA vs. MVO	DESMAOA vs. PSO
F1	2.50×10^{-1}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F2	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F3	6.10×10^{-5}	6.10×10^{-5}	9.77×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F4	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F5	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	8.54×10^{-4}	6.10×10^{-5}	6.10×10^{-5}
F6	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	1.22×10^{-4}	6.10×10^{-5}
F7	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F8	2.56×10^{-2}	6.10×10^{-5}	1.22×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F9	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F10	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F11	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	1.00×10^{0}	6.10×10^{-5}	4.88×10^{-4}	6.10×10^{-5}
F12	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	1.22×10^{-4}
F13	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}

Table 11. *p*-Values of the Wilcoxon signed-rank test between DESMAOA and other competitor algorithms on 1000 dimensions.

Function	DESMAOA vs. SMA	DESMAOA vs. AOA	DESMAOA vs. GWO	DESMAOA vs. WOA	DESMAOA vs. SSA	DESMAOA vs. MVO	DESMAOA vs. PSO
F1	3.91×10^{-3}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F2	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F3	6.10×10^{-5}	6.10×10^{-5}	1.22×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F4	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F5	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-4}	6.10×10^{-5}	6.10×10^{-5}
F6	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	1.22×10^{-4}	6.10×10^{-5}
F7	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	8.36×10^{-3}
F8	4.21×10^{-1}	6.10×10^{-5}	1.22×10^{-4}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F9	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F10	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}
F11	1.00×10^{0}	6.10×10^{-5}	6.10×10^{-5}	1.00×10^{0}	6.10×10^{-5}	9.77×10^{-4}	6.10×10^{-5}
F12	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	1.16×10^{-3}
F13	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}	6.10×10^{-5}

Table 12. Experimental results of Friedman test on 50, 200, and 1000 dimensions.

Algorithm	D = 50		D = 200		D = 1000	
	Mean	Rank	Mean		Mean	Rank
DESMAOA	1.1923	1	1.2308	1	1.2692	1
SMA	1.9615	2	2.0000	2	2.1923	2
AOA	4.7692	5	4.5385	5	4.4615	4
GWO	4.3077	3	4.3846	4	4.6923	5
WOA	4.3846	4	3.7692	3	3.4615	3
SSA	6.7692	7	7.0000	8	6.1538	7
MVO	6.8462	8	6.7692	7	7.4615	8
PSO	5.7692	6	6.3077	6	6.3077	6

Table 13. The result statistics of CEC2021 test functions for the DESMAOA and competitor algorithms.

Function	Metric	DESMAOA	SMA	AOA	GWO	WOA	SSA	MVO	PSO
CEC_01	Mean	3.4126×10^{3}	8.4790×10^{3}	1.4400×10^{10}	8.5800×10^{7}	8.8200×10^{7}	3.0497×10^{3}	2.0700×10^{4}	2.6509×10^{3}
	Std	3.2484×10^{3}	4.3942×10^{3}	5.4800×10^{9}	2.0600×10^{8}	1.0800×10^{8}	2.6586×10^{3}	1.2300×10^{4}	2.8147×10^{3}
CEC_02	Mean	1.6460×10^{3}	1.7230×10^{3}	2.3794×10^{3}	1.7650×10^{3}	2.3408×10^{3}	1.9202×10^{3}	1.7883×10^{3}	2.0353×10^{3}
	Std	1.8064×10^{2}	2.0356×10^{2}	2.5500×10^{2}	4.0002×10^{2}	2.8907×10^{2}	3.0299×10^{2}	2.8914×10^{2}	3.3248×10^{2}
CEC_03	Mean	7.4197×10^{2}	7.3268×10^{2}	8.0255×10^{2}	7.3186×10^{2}	7.9430×10^{2}	7.4133×10^{2}	7.3257×10^{2}	7.3096×10^{2}
	Std	1.3713×10^{1}	9.9242×10^{0}	8.5999×10^{0}	1.0358×10^{1}	2.9899×10^{1}	1.5608×10^{1}	9.4278×10^{0}	1.1527×10^{1}
CEC_04	Mean	1.9024×10^{3}	1.9015×10^{3}	3.9200×10^{5}	1.9028×10^{3}	1.9116×10^{3}	1.9015×10^{3}	1.9014×10^{3}	1.9011×10^{3}
	Std	1.5132×10^{0}	4.7478×10^{-1}	2.0400×10^{5}	1.1137×10^{0}	8.0854×10^{0}	4.6367×10^{-1}	6.4272×10^{-1}	6.8410×10^{-1}
CEC_05	Mean	4.7672×10^{3}	2.0900×10^{4}	4.6200×10^{5}	1.1600×10^{5}	5.8000×10^{5}	3.2100×10^{4}	6.7593×10^{3}	5.0581×10^{3}
	Std	4.2784×10^{3}	4.6800×10^{4}	1.1100×10^{5}	1.9300×10^{5}	8.5200×10^{5}	7.2900×10^{4}	4.2953×10^{3}	3.3481×10^{3}
CEC_06	Mean	1.7312×10^{3}	1.7684×10^{3}	2.2222×10^{3}	1.7776×10^{3}	1.8431×10^{3}	1.7573×10^{3}	1.7587×10^{3}	1.8632×10^{3}
	Std	1.2285×10^{2}	9.1820×10^{1}	2.0670×10^{2}	1.1171×10^{2}	1.0223×10^{2}	8.6052×10^{1}	1.0387×10^{2}	1.0503×10^{2}
CEC_07	Mean	7.0311×10^{3}	6.5603×10^{3}	2.9700×10^{6}	1.8000×10^{4}	3.4500×10^{5}	7.3733×10^{3}	7.5164×10^{3}	6.0549×10^{3}
	Std	7.7063×10^{3}	6.4319×10^{3}	3.8500×10^{6}	3.7100×10^{4}	5.3700×10^{5}	4.9714×10^{3}	6.1260×10^{3}	2.6469×10^{3}
CEC_08	Mean	2.2974×10^{3}	2.4032×10^{3}	3.5700×10^{3}	2.3384×10^{3}	2.4289×10^{3}	2.3012×10^{3}	2.3861×10^{3}	2.4193×10^{3}
	Std	2.2368×10^{1}	3.1043×10^{2}	3.7790×10^{2}	9.1745×10^{1}	3.3946×10^{2}	1.4399×10^{1}	2.6287×10^{2}	3.7983×10^{2}
CEC_09	Mean	2.7018×10^{3}	2.7599×10^{3}	2.9038×10^{3}	2.7449×10^{3}	2.7752×10^{3}	2.7330×10^{3}	2.7514×10^{3}	2.7922×10^{3}
	Std	1.0302×10^{2}	1.0211×10^{1}	9.9242×10^{1}	4.4570×10^{1}	6.2479×10^{1}	6.4081×10^{1}	9.5351×10^{0}	1.0509×10^{2}
CEC_10	Mean	2.9231×10^{3}	2.9323×10^{3}	3.6576×10^{3}	2.9366×10^{3}	2.9545×10^{3}	2.9289×10^{3}	2.9290×10^{3}	2.9234×10^{3}
	Std	2.2799×10^{1}	3.1577×10^{1}	4.0387×10^{2}	2.4483×10^{1}	6.9125×10^{1}	2.4243×10^{1}	2.9085×10^{1}	2.3865×10^{1}
Average rank		2.3	3.9	7.9	4.6	6.9	3.3	3.7	3.4
Rank		1	5	8	6	7	2	4	3

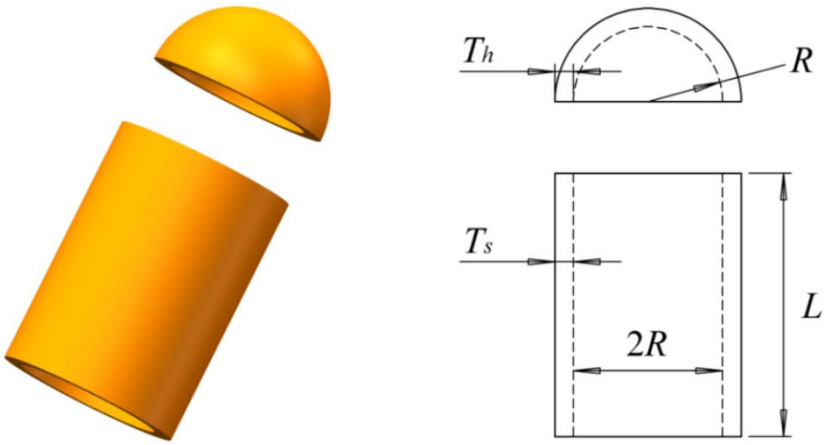

Figure 5. Pressure vessel design problem: model diagram (**left**) and structure parameters (**right**).

From Table 13, it can be observed that the DESMAOA was able to obtain the best results in six functions: CEC_02, CEC_05, CEC_06, CEC_08, CEC_09, and CEC_10. Thus, we can find that the DESMAOA has good performance in hybrid and composition test functions. By comparing it with other optimization algorithms, we found that DESMAOA showed very competitive performance for these CEC2021 test functions. Moreover, the Friedman's ranking test was also used to evaluate the performance of DESMAOA. The

average rank and rank were also given in Table 13. It can be seen that DESMAOA obtained the best statistical ranking result among these algorithms.

Therefore, the results of CEC2021 test functions also showed the high performance for solving optimization problems.

5. Applicability for Solving Engineering Design Problems

This section reports the three classical engineering design problems we employed to evaluate the capability of DESMAOA to solve practical problems, which were the pressure vessel design problem, three-bar truss design problem, and tension/compression spring design problem. In the same way, 30 search agents and 500 iterations were utilized in the design procedure of engineering problems for a fair comparison. Meanwhile, other related results of optimization algorithms proposed by scholars are also given and compared with proposed algorithm here. Detailed descriptions are shown below.

5.1. Pressure Vessel Design

The design of the pressure vessel is an optimization problem with four variables and four constraints in the industrial field [43]. The lowest cost of pressure vessel was the ultimate goal. The structure of pressure vessel is shown in Figure 5. The four design variables were the thickness of the shell (T_s), thickness of the head (T_h), inner radius (R), and length of the cylindrical section (L). Table 14 lists the comparison between DESMAOA and other competitor algorithms. From Table 14, we can see that DESMAOA was capable of finding the optimal solution with the lowest cost.

Table 14. Optimal results for comparative algorithms on the pressure vessel design problem.

Algorithm	Optimal Values for Variables				Optimal Cost
	T_s	T_h	R	L	
DESMAOA	7.943124×10^{-1}	3.927124×10^{-1}	4.288001×10^{1}	1.671866×10^{2}	5.8363262×10^{3}
SMA [33]	7.931×10^{-1}	3.932×10^{-1}	4.06711×10^{1}	1.962178×10^{2}	5.9941857×10^{3}
AOA [34]	8.303737×10^{-1}	4.162057×10^{-1}	4.275127×10^{1}	1.693454×10^{2}	6.0487844×10^{3}
MVO [5]	8.125×10^{-1}	4.375×10^{-1}	4.2090738×10^{1}	1.7673869×10^{2}	6.0608066×10^{3}
WOA [39]	8.12500×10^{-1}	4.37500×10^{-1}	4.2098209×10^{1}	1.76638998×10^{2}	6.0597410×10^{3}
MFO [44]	8.125×10^{-1}	4.375×10^{-1}	4.2098445×10^{1}	1.76636596×10^{2}	6.0597143×10^{3}
GWO [38]	8.125×10^{-1}	4.345×10^{-1}	4.20892×10^{1}	1.767587×10^{2}	6.0515639×10^{3}
MOSCA [45]	7.781909×10^{-1}	3.830476×10^{-1}	4.03207539×10^{1}	$1.999841994 \times 10^{2}$	5.88071150×10^{3}
LWOA [46]	7.78858×10^{-1}	3.85321×10^{-1}	4.032609×10^{1}	2.00×10^{2}	5.893339×10^{3}
IMFO [47]	7.781948×10^{-1}	3.846621×10^{-1}	4.032097×10^{1}	1.999812×10^{2}	5.8853778×10^{3}

5.2. Three-Bar Truss Design

The aim of the three-bar truss design is to achieve the lowest weight of three-bar truss with the constraints of stress, deflection, and buckling, which belongs to the field of civil engineering [48]. In this design problem, two parameters x_1 (or A1) and x_2 (or A2) were involved, as shown in Figure 6. The solutions obtained by the DESMAOA and other representative algorithms are listed in Table 15. It can be seen that the proposed hybrid method apparently outperformed other approaches. Moreover, 30 repeated tests were also performed to evaluate the robustness of the proposed algorithm. The worst value, mean value, best value, and stand deviation were 2.639079×10^{2}, 2.638562×10^{2}, 2.638523×10^{2}, and 1.0451×10^{-2}. Hence, the statistical results revealed that the proposed algorithm had very stable and superior performance in solving this design problem.

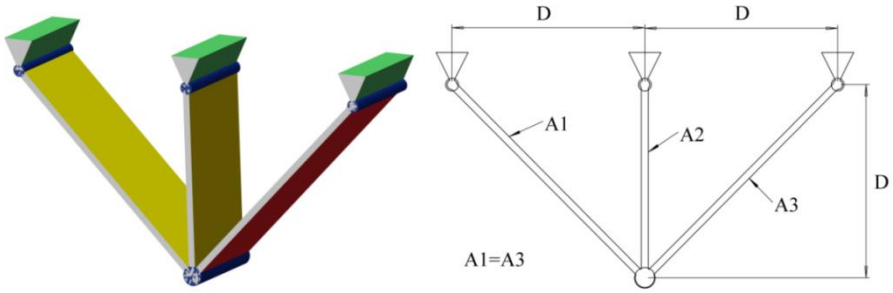

Figure 6. Three-bar truss design problem: model diagram (**left**) and structure parameters (**right**).

Table 15. Optimal results for comparative algorithms on the three-bar truss design problem.

Algorithm	Optimal Values for Variables		Optimal Weight
	x_1	x_2	
DESMAOA	7.882549×10^{-1}	4.085642×10^{-1}	$2.638523657 \times 10^{2}$
SMA [33]	7.729316×10^{-1}	4.718874×10^{-1}	$2.658067955 \times 10^{2}$
AOA [34]	7.9369×10^{-1}	3.9426×10^{-1}	2.639154×10^{2}
MBA [48]	7.885650×10^{-1}	4.085597×10^{-1}	$2.638958522 \times 10^{2}$
SSA [40]	$7.88665414 \times 10^{-1}$	$4.08275784 \times 10^{-1}$	$2.638958434 \times 10^{2}$
MFO [44]	$7.88244771 \times 10^{-1}$	$4.09466906 \times 10^{-1}$	$2.638959797 \times 10^{2}$
PSO-DE [49]	7.886751×10^{-1}	4.082482×10^{-1}	$2.638958433 \times 10^{2}$
HSCAHS [50]	7.885721×10^{-1}	4.084012×10^{-1}	2.63881992×10^{2}

5.3. Tension/Compression Spring Design

In the design of a tension/compression spring [51], the objective is to obtain the minimum optimal weight under three constraints: (1) shear stress, (2) surge frequency, and (3) deflection. As shown in Figure 7, there were three variables that needed to be considered. They were the wire diameter (d), mean coil diameter (D), and the number of active coils (N). The results of DESMAOA and other comparative algorithms are listed in Table 16. By comparison, the proposed DESMAOA achieved the best solution for this problem, which was 5.44827×10^{-2}, 4.83109×10^{-1}, and 5.746128×10^{0} for d, D, and N, respectively. Moreover, the optimal weight was 1.11083×10^{-2}.

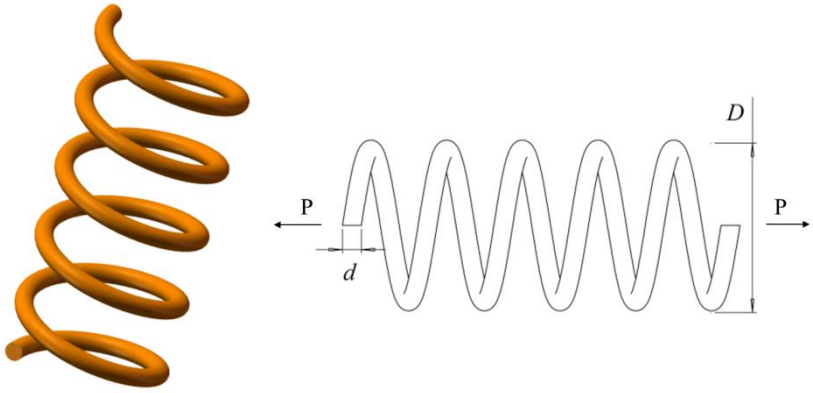

Figure 7. Tension/compression spring design problem: model diagram (**left**) and structure parameters (**right**).

Table 16. Optimal results for comparative algorithms on the tension/compression spring design problem.

Algorithm	Optimal Values for Variables			Optimal Weight
	d	***D***	***p***	
DESMAOA	5.44827×10^{-2}	4.83109×10^{-1}	5.746128×10^{0}	1.11083×10^{-2}
SMA [33]	5.8992×10^{-2}	6.23402×10^{-1}	3.590304×10^{0}	1.2128×10^{-2}
AOA [34]	5.00×10^{-2}	3.49809×10^{-1}	1.18637×10^{1}	1.2124×10^{-2}
MVO [5]	5.251×10^{-2}	3.7602×10^{-1}	1.033513×10^{1}	1.2790×10^{-2}
AO [14]	5.02439×10^{-2}	3.5262×10^{-1}	1.05425×10^{1}	1.1165×10^{-2}
SSA [40]	5.1207×10^{-2}	3.45215×10^{-1}	1.2004032×10^{1}	1.26763×10^{-2}
GWO [38]	5.169×10^{-2}	3.56737×10^{-1}	1.128885×10^{1}	1.2666×10^{-2}
GSA [6]	5.0276×10^{-2}	3.23680×10^{-1}	1.3525410×10^{1}	1.27022×10^{-2}
WSA [51]	5.168626×10^{-2}	3.5665047×10^{-1}	$1.129291654 \times 10^{1}$	1.267061×10^{-2}

6. Conclusions and Future Works

To overcome the shortcomings of basic meta-heuristic algorithms, this paper presents an effective deep ensemble method of two very new optimization algorithms, i.e., the SMA and AOA. A preliminary hybrid of these two algorithms was firstly conducted to enhance the capability of exploration. Then, two strategies were integrated to the hybridized algorithm to assist it to jump out of the local minima and improve the accuracy of the solution. The performance of proposed DESMAOA was extensively analyzed by using 23 classical test functions.

First, different combinations of SMAOA and two strategies were analyzed and discussed. The results revealed the effectiveness of proposed strategies. Then, the results of DESMAOA were compared with SMA, AOA, and five well-known algorithms. The results showed that the proposed method had the advantages of both SMA and AOA and that it also was evidently better than other comparison algorithms. Afterward, experimental tests in high dimensional environments (50, 200, and 1000) were also investigated among these comparative algorithms, and the results of scalability test also confirmed the superior performance of the proposed method. Finally, the proposed DESMAOA was employed to deal with three engineering design problems. The results show that the proposed method was good at solving these problems, and in particular it was very stable when solving the three-bar truss design problem.

As future perspectives, the DESMAOA can be utilized to solve more optimization problems in other disciplines, such as the feature selection, training of multi-layer perceptron neural network, and image processing. Another investigation is to consider the implementation of this hybrid method on other optimization algorithms for better optimization performance.

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

Funding: This work was supported by the Sanming University introduces high-level talents to start scientific research funding support project (21YG01, 20YG14), Fujian Natural Science Foundation Project (2021J011128), Guiding science and technology projects in Sanming City (2021-S-8), Educational research projects of young and middle-aged teachers in Fujian Province (JAT200618), Scientific research and development fund of Sanming University (B202009), and Funded By Open Research Fund Program of Fujian Provincial Key Laboratory of Agriculture Internet of Things Application (ZD2101).

Data Availability Statement: Not applicable.

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

References

1. Abualigah, L.; Diaba, A. Advances in sine cosine algorithm: A comprehensive survey. *Artif. Intell. Rev.* **2021**, *54*, 2567–2608. [CrossRef]
2. Abualigah, L.; Diaba, A. A comprehensive survey of the Grasshopper optimization algorithm: Results, variants, and applications. *Neural Comput. Appl.* **2020**, *32*, 15533–15556. [CrossRef]
3. Jia, H.; Lang, C.; Oliva, D.; Song, W.; Peng, X. Dynamic Harris Hawks Optimization with Mutation Mechanism for Satellite Image Segmentation. *Remote Sens.* **2019**, *11*, 1421. [CrossRef]
4. Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115665. [CrossRef]
5. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
6. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
7. Kaveh, A.; Dadras, A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Adv. Eng. Softw.* **2017**, *110*, 69–84. [CrossRef]
8. Asef, F.; Majidnezhad, V.; Feizi-Derakhshi, M.R.; Parsa, S. Heat transfer relation-based optimization algorithm (HTOA). *Soft. Comput.* **2021**, *25*, 8129–8158. [CrossRef]
9. Corriveau, G.; Guilbault, R.; Tahan, A.; Sabourin, R. Bayesian network as an adaptive parameter setting approach for genetic algorithms. *Complex Intell. Syst.* **2016**, *2*, 1–22. [CrossRef]
10. Storn, R.; Price, K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **2010**, *23*, 689–694.
11. Yao, X.; Liu, Y.; Lin, G. Evolutionary Programming Made Faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102.
12. Chen, G.; Yu, J. Particle swarm optimization algorithm. *Inf. Control* **2005**, *186*, 454–458.
13. Gaurav, D.; Vijay, K. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. *Knowl. Based Syst.* **2018**, *159*, 20–50.
14. Abualigah, L.; Yousri, D.; Elaziz, M.A.; Ewees, A.A.; Al-qaness, M.A.A.; Gandomi, A.H. Aquila optimizer: A novel meta-heuristic optimization Algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
15. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine predators algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
16. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
17. Satapathy, S.; Naik, A. Social group optimization (SGO): A new population evolutionary optimization technique. *Complex Intell. Syst.* **2016**, *2*, 173–203. [CrossRef]
18. Al-Betar, M.A. β-hill climbing: An exploratory local search. *Neural Comput. Appl.* **2017**, *28*, 153–168. [CrossRef]
19. Martínez-Lvarez, F.; Asencio-Cortés, G.; Torres, J.F.; Gutiérrez-Avilés, D.; Melgar-García, L.; Pérez-Chacón, R.; Rubio-Escudero, C.; Riquelme, J.C.; Troncoso, A. Coronavirus Optimization Algorithm: A bioinspired metaheuristic based on the COVID-19 propagation model. *Big Data* **2020**, *8*, 308–322. [CrossRef] [PubMed]
20. Hussain, A.; Muhammad, Y.S. Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. *Complex Intell. Syst.* **2019**, *6*, 1–14. [CrossRef]
21. Wolpert, D.H.; Macready, W.G. No Free Lunch Theorems for Optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
22. Shehadeh, H.A. A hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) for global optimization. *Neural Comput. Appl.* **2021**, *33*, 11739–11752.
23. Kaveh, A.; Rahmani, P.; Eslamlou, A.D. An efficient hybrid approach based on Harris Hawks optimization and imperialist competitive algorithm for structural optimization. *Eng. Comput.* **2021**, *277*, 1–29.
24. Dhiman, G.; Kaur, A. A hybrid algorithm based on particle swarm and spotted hyena optimizer for global optimization. In *Soft Computing for Problem Solving*; Springer: Singapore, 2019; Volume 1, pp. 599–615.
25. Dhiman, G. SSC: A hybrid nature-inspired meta-heuristic optimization algorithm for engineering applications. *Knowl. Based Syst.* **2021**, *222*, 106926. [CrossRef]
26. Banaie-Dezfouli, M.; Nadimi-Shahraki, M.H.; Beheshti, Z. R-GWO: Representative-based grey wolf optimizer for solving engineering problems. *Appl. Soft Comput.* **2021**, *106*, 107328. [CrossRef]
27. Zhang, H.; Wang, Z.; Chen, W.; Heidari, A.A.; Wang, M.; Zhao, X.; Liang, G.; Chen, H.; Zhang, X. Ensemble mutation-driven salp swarm algorithm with restart mechanism: Framework and fundamental analysis. *Expert Syst. Appl.* **2021**, *165*, 113897. [CrossRef]
28. Yu, C.; Heidari, A.A.; Xue, X.; Zhang, L.; Chen, H.; Chen, W. Boosting quantum rotation gate embedded slime mould algorithm. *Expert Syst. Appl.* **2021**, *181*, 115082. [CrossRef]
29. Zhang, H.; Cai, Z.; Ye, X.; Wang, M.; Kuang, F.; Chen, H.; Li, C.; Li, Y. A multi-strategy enhanced salp swarm algorithm for global optimization. *Eng. Comput.* **2020**. [CrossRef]
30. Che, Y.; He, D. A Hybrid Whale Optimization with Seagull Algorithm for Global Optimization Problems. *Math. Probl. Eng.* **2021**, *2021*, 1–31.
31. Hassan, B.A. CSCF: A chaotic sine cosine firefly algorithm for practical application problems. *Neural Comput. Appl.* **2021**, *33*, 7011–7030. [CrossRef]

32. Yue, S.; Zhang, H. A hybrid grasshopper optimization algorithm with bat algorithm for global optimization. *Multimed. Tools Appl.* **2021**, *80*, 3863–3884. [CrossRef]
33. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime Mould Algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [CrossRef]
34. 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]
35. Mirjalili, S. SCA: A Sine Cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
36. Molga, M.; Smutnicki, C. Test Functions for Optimization Needs. 2005. Available online: http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf (accessed on 1 October 2021).
37. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K.; Agrawal, P.; Kumar, A.; Suganthan, P.N. Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Cairo University. *Tech. Rep.* 2020. Available online: http://home.elka.pw.edu.pl/~{}ewarchul/cec2021-specification.pdf (accessed on 1 October 2021).
38. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
39. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
40. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
41. Demsar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
42. Garcia, S.; Fernandez, A.; Luengo, J.; Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inform. Sci.* **2010**, *180*, 2044–2064. [CrossRef]
43. Kannan, B.; Kramer, S.N. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *J. Mech. Des.* **1994**, *116*, 405–411. [CrossRef]
44. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [CrossRef]
45. Rizk-Allah, R.M. Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems. *J. Comput. Des. Eng.* **2018**, *5*, 249–273. [CrossRef]
46. Zhou, Y.; Ling, Y.; Luo, Q. Lévy flight trajectory-based whale optimization algorithm for engineering optimization. *Eng. Comput.* **2018**, *35*, 2406–2428. [CrossRef]
47. 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]
48. Sadollah, A.; Bahreininejad, A.; Eskandar, H.; Hamdi, M. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Appl. Soft Comput.* **2013**, *13*, 2592–2612. [CrossRef]
49. Liu, H.; Cai, Z.; Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl. Soft Comput.* **2010**, *10*, 629–640. [CrossRef]
50. Singh, N.; Kaur, J. Hybridizing sine-cosine algorithm with harmony search strategy for optimization design problems. *Soft. Comput.* **2021**. [CrossRef]
51. Baykasoğlu, A.; Akpinar, S. Weighted superposition attraction (WSA): A swarm intelligence algorithm for optimization problems–part2: Constrained optimization. *Appl. Soft Comput.* **2015**, *37*, 396–415. [CrossRef]