algorithms

Article

Improved Slime Mold Algorithm with Dynamic Quantum Rotation Gate and Opposition-Based Learning for Global Optimization and Engineering Design Problems

Yunyang Zhang [1], Shiyu Du [2,*] and Quan Zhang [1]

[1] College of Information Science and Engineering, Ningbo University, Ningbo 315211, China
[2] Engineering Laboratory of Advanced Energy Materials, Ningbo Institute of Materials Technology and Engineering, Ningbo 315211, China
* Correspondence: dushiyu@nimte.ac.cn

Abstract: The slime mold algorithm (SMA) is a swarm-based metaheuristic algorithm inspired by the natural oscillatory patterns of slime molds. Compared with other algorithms, the SMA is competitive but still suffers from unbalanced development and exploration and the tendency to fall into local optima. To overcome these drawbacks, an improved SMA with a dynamic quantum rotation gate and opposition-based learning (DQOBLSMA) is proposed in this paper. Specifically, for the first time, two mechanisms are used simultaneously to improve the robustness of the original SMA: the dynamic quantum rotation gate and opposition-based learning. The dynamic quantum rotation gate proposes an adaptive parameter control strategy based on the fitness to achieve a balance between exploitation and exploration compared to the original quantum rotation gate. The opposition-based learning strategy enhances population diversity and avoids falling into the local optima. Twenty-three benchmark test functions verify the superiority of the DQOBLSMA. Three typical engineering design problems demonstrate the ability of the DQOBLSMA to solve practical problems. Experimental results show that the proposed algorithm outperforms other comparative algorithms in convergence speed, convergence accuracy, and reliability.

Keywords: slime mold algorithm; metaheuristics algorithm; engineering design problem; dynamic quantum rotation gate; opposition-based learning

Citation: Zhang, Y.; Du, S.; Zhang, Q. Improved Slime Mold Algorithm with Dynamic Quantum Rotation Gate and Opposition-Based Learning for Global Optimization and Engineering Design Problems. *Algorithms* **2022**, *15*, 317. https://doi.org/10.3390/a15090317

Academic Editor: Günther Raidl

Received: 19 July 2022
Accepted: 31 August 2022
Published: 4 September 2022

1. Introduction

In the optimization field, solving an optimization problem usually means finding the optimal value to maximize or minimize a set of objective functions without violating constraints [1]. Optimization methods can be divided into two main categories: exact algorithms and metaheuristics [2]. While exact algorithms can provide global optima precisely, they have exponentially increasing execution times in proportion to the number of variables and are considered less suitable and practical [3]. In contrast, metaheuristic algorithms can identify the best or near-optimal solution in a reasonable amount of time [4]. During the last two decades, metaheuristic algorithms have gained much attention, and much development and work there have been on them due to their flexibility, simplicity, and global optimization. Thus, they are widely used for solving optimization problems in almost every domain, such as big data text clustering [5], tuning of fuzzy control systems [6,7], path planning [8,9], feature selection [10–12], training neural networks [13], parameter estimation for photovoltaic cells [14–16], image segmentation [17,18], tomography analysis [19], and permutation flowshop scheduling [20,21].

Metaheuristic algorithms simulate natural phenomena or laws of physics and are usually classified into three categories: evolutionary algorithms, physical and chemical algorithms, and swarm-based algorithms. Evolutionary algorithms are a class of algorithms that simulate the laws of evolution in nature. The best known is the genetic algorithm

(GA) [22], which was developed from Darwin's theory of superiority and inferiority. There are other algorithms, such as differential evolution (DE) [23], which simulates the crossover and variation mechanisms of inheritance, evolutionary programming (EP) [24], and evolutionary strategies (ES) [25]. Physical and chemical algorithms search for the optimum by simulating the universe's chemical laws or physical phenomena. Algorithms in this category include simulated annealing (SA) [26], electromagnetic field optimization (EFO) [27], equilibrium optimizer (EO) [28], and Archimedes' optimization algorithm (ArchOA) [29]. Swarm-based algorithms simulate the behavior of social groups of animals or humans. Examples of such algorithms include the whale optimization algorithm (WOA) [30], salp swarm algorithm (SSA) [31], moth search algorithm (MSA) [32], aquila optimizer (AO) [33], grey wolf optimizer (GWO) [34], harris hawks optimization (HHO) [35], and particle swarm optimization (PSO) [36].

However, the no free lunch (NFL) theorem [37] proves that no single algorithm can solve all optimization problems well. If an algorithm is particularly effective for a particular class of problems, it may not be able to solve other classes of optimization problems. This motivates us to propose new algorithms or improve the existing ones. The slime mold algorithm (SMA) [38] is a new meta-heuristic algorithm proposed by Li et al. in 2020. The basic idea of the SMA is based on the foraging behavior of slime mold, which has different feedback aspects according to the food quality. Different search mechanisms have been introduced into the SMA to solve various optimization problems. For example, Zhao et al. [39] introduced a diffusion mechanism and association strategy into the SMA and applied the proposed algorithm to the image segmentation of CT images. Salah L. et al. [40] applied the slime mold algorithm to optimize an artificial neural network model for predicting monthly stochastic urban water demand. Wang et al. [41] developed a parallel slime mold algorithm for the distribution network reconfiguration problem with distributed generation. Tang et al. [42] introduced chaotic opposition-based learning and spiral search strategies into the SMA and proposed two adaptive parameter control strategies. The simulation results show that the proposed algorithms outperform other similar algorithms. Örnek et al. [43] proposed an enhanced SMA that combines the sine cosine algorithm with the position update of the SMA. Experimental results show that the proposed hybrid algorithm has a better ability to jump out of local optima with faster convergence.

Although the SMA, as a new algorithm, is competitive with other algorithms, it also suffers from some shortcomings. The SMA, similarly to many other swarm-based metaheuristic algorithms, suffers from slow convergence and premature convergence to a local optimum solution [44]. In addition, the update strategy of SMA reduces exploration capabilities and reduces population diversity. To improve the above problems, an improved algorithm based on SMA, called the dynamic-quantum-rotation-gate- and opposition-based learning SMA (DQOBLSMA), is proposed. In this paper, we introduce two mechanisms, the dynamic quantum rotation gate (DQGR) and opposition-based learning (OBL), into the SMA simultaneously. Both mechanisms improve the shortcomings of the original algorithm in terms of slow convergence and the tendency to fall into local optima. First, DQGR rotates the search individuals to the direction of the optimum, improving the diversity of the population and enhancing the global exploration capability of the algorithm. At the same time, OBL explores the partial solution in the opposite direction, improving the algorithm's ability to jump out of local optima. The performance of the DQOBLSMA was evaluated by comparing it with the original SMA algorithm and with other advanced algorithms. In addition, three different constraint engineering problems were used to verify the performance of the DQOBLSMA further: the welded beam design problem, the tension/compression spring design problem, and pressure vessel design.

The main contributions of this paper are summarized as follows:

1. DQRG and OBL strategies were introduced into SMA to improve the exploration capabilities of SMA.
2. The DQRG strategy is proposed in order to balance the exploration and exploitation phases.

3. By comparing five well-known metaheuristic algorithms, experiments show that the proposed DQOBLSMA is more robust and effective.
4. Experiments on three engineering design optimization problems show that the DQOBLSMA can be effectively applied to practical engineering problems.

This paper is organized as follows. Section 2 describes the slime mold algorithm, quantum rotation gate, and opposition-based learning. Section 3 presents the proposed improved slime mold algorithm. Sections 4 show the experimental study and discussion using benchmark functions. The DQOBLSMA is applied to solve the three engineering problems in Section 5. Finally, the conclusion and future work are given in Section 6.

2. Materials and Methods

2.1. Slime Mold Algorithm

The slime mold algorithm (SMA) [38] is a swarm-based metaheuristic algorithm recently developed by Li et al. The algorithm simulates a range of behaviors for foraging by the slime mold. The negative and positive feedbacks of the slime mold using a biological oscillator to propagate waves during foraging for a food source are simulated by the SMA using adaptive weights. Three special behaviors of the slime mold are mathematically formulated in the SMA: approaching food, wrapping food, and grabbing food. The process of approaching food can be expressed as

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

where t is the number of current iterations, $X_i(t+1)$ is the newly generated position, $Xb(t)$ denotes the best position found by the 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 is a random value in $[0,1]$.

vb and $vs.c$ are the coefficients that simulate the oscillation and contraction mode of slime mold, respectively, and $vs.c$ 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

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

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

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

$$SmellIndex(i) = Sort(S(i)) \tag{4}$$

where N is the size of the population, i represents the $i-th$ individual in the population, $i \in 1, 2 \ldots N$, $rand$ denotes the random value in the interval of $[0,1]$, bF denotes the optimal fitness obtained in the current iterative process, wF denotes the worst fitness value obtained in the iterative process currently, $S(i)$ represents the fitness of X, $SmellIndex$ denotes the sequence of fitness values sorted.

$$p = \tanh|S(i) - DF| \tag{5}$$

where DF denotes the best fitness obtained in all iterations.

Finally, when the slime mold has found the food , it still has a certain chance z to search other new food, which is formulated as

$$X(t+1) = \text{rand} \cdot (UB - LB) + LB, r_2 < z \tag{6}$$

where UB and LB are the upper and lower limits, respectively, and r_2 implies a random value in the region $[0,1]$. z is set to 0.03 in original SMA.

Finally, the pseudo-code of SMA is given in Algorithm 1.

Algorithm 1: Pseudo-code of the slime mold algorithm (SMA).

```
Input: Population size $N$, Maximum number of iteration $MaxIt$.
Output: The best location $X_b$, the best fitness value $bestFitness$ .
1  Initialize the parameters popsize($N$);
2  Initialize the positions of slime mold $X_i (i = 1, 2, 3..., N)$ ;
3  while $t < MaxIt$ do
4      Calculate the fitness of all slime molds;
5      Update $bestFitness$,$X_b$;
6      Calculate the $W$ by Equation (3);
7      foreach each slime mold do
8          if $r_2$<$z$ then
9              update the position using Equation (6);
10         else
11             Update $p$, $vb$, and $vc$;
12             Update position by Equation (1);
13         end
14     end
15     $t = t + 1$;
16 end
17 return $bestFitness$,$X_b$
```

2.2. Description of the Quantum Rotation Gate

2.2.1. Quantum Bit

The fundamental storage unit is a quantum bit in quantum computer systems, communication systems, and other quantum information systems [45]. The difference between quantum bits and classical bits is that quantum bits can be in a superposition of two states simultaneously, whereas classical bits can be in only one state at a period of time, which is defined as Equation (7).

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{7}$$

where α and β represent the probability amplitudes of the two superposition states. $|\alpha|^2$ and $|\beta|^2$ are the e probabilities that the qubit is in two different states of "0" and "1", and the relationship between them is shown in Equation (8).

$$|\alpha|^2 + |\beta|^2 = 1 \tag{8}$$

Thus, a quantum bit can represent one state or be in both states at the same time.

2.2.2. Quantum Rotation Gate

In the DQOBLSMA, the QRG strategy is introduced to update the position of some search individuals to enhance the exploitation of the algorithm. In the physical discipline of quantum computing, the quantum rotation gate is used as a state processing technique. Quantum bits are binary, and the position information generated by the swarm-based algorithm is floating-point data. In order to process the position information, the discrete data of the quantum bits need to be turned into the algorithm's continuous data. The information of each dimension of the search agent is rotated in couples and updated by a quantum rotation gate. The update process and adjustment operation of QRG are as follows. Equation (9) shows that the 2×2 matrix represents the quantum rotation gate.

$$U(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \tag{9}$$

The updating process is as follows:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = U(\theta_i)\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{10}$$

where$(\alpha_i, \beta_i)^T$ shows the state of the quantum bit of the ith quantum bit of the chromosome before the update of the quantum rotation gate, and $(\alpha_i', \beta_i')^T$ indicates the state of the quantum bit after the update. θ_i denotes the rotation angle of the ith quantum bit, the size and sign of which have been pre-set, and its adjustment strategy is shown in Table 1.

Table 1. Strategies for specifying rotation angle in QRG.

Situation	$\Delta\theta_i$	$s(\alpha_i, \beta_i)$			
		$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\alpha_i\beta_i > 0$	$\beta_i = 0$
$f(x_i) = best\_fitness$	δ	0	0	0	0
$f(x_i) > best\_fitness$	δ	-1	± 1	$+1$	0
$f(x_i) < best\_fitness$	δ	$+1$	0	-1	± 1

Table 1 shows the rotation angle is labeled by $\theta_i = \Delta\theta_i \cdot s(\alpha_i, \beta_i)$, where $s(\alpha_i, \beta_i)$ denotes the rotation of the target direction. $\Delta\theta_i$ represents the rotation angle of the i-th rotation, where the position state of the i-th search agent in the population is α_i, and the position state of the optimal search agent in the whole population is β_i. By comparing the fitness values of the current target and the optimal target, the direction of the target with higher fitness is selected to rotate the individual, thereby expanding the search space. If $f(x_i) > best\_fitness$, then the algorithm evolves toward the current target. Conversely, let the quantum bit state vector transform towards the direction where the optimal individual exists [46]. Figure 1 shows the quantum bit state vector transformation process.

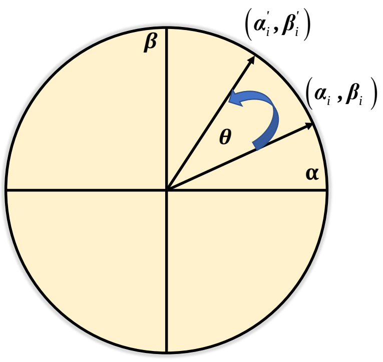


Figure 1. The process of updating the state of a quantum bit.

2.3. Opposition-Based Learning (OBL)

Tizhoosh proposed OBL in 2005 [47]. This technique can increase the convergence speeds of metaheuristic algorithms by replacing a solution in the population by searching for a potentially better solution in the opposite direction of the current one. With this approach, a population with better solutions could be generated after each iteration and accelerate convergence speed. The OBL strategy has been successfully used in various metaheuristic algorithms to improve the ability of local optima stagnation avoidance [48], and the mathematical expression is as follows:

$$X_{\mathrm{OBL}}(t) = LB + UB - X(t) \tag{11}$$

In opposition-based learning, for the original solution $X(t)$ and the reverse solution $X_{\mathrm{OBL}}(t)$, according to their fitness, save the better solution among them. Finally, the slime mold position for the next iteration is updated as follows in the minimization problem:

$$X_{\mathrm{OBL}}(t+1) = \begin{cases} X_{\mathrm{OBL}}(t) & \text{if } f(X_{\mathrm{OBL}}(t)) < f(X(t)) \\ X(t) & \text{if } f(X_{\mathrm{OBL}}(t)) \geq f(X(t)) \end{cases} \tag{12}$$

3. Proposed Method

3.1. Improved Quantum Rotation Gate

The magnitude of the rotation angle of the QRG significantly affects the convergence speed. A relatively large amplitude leads to premature convergence. Conversely, smaller angles lead to slower convergence. In particular, the rotation angle of the original quantum rotation gate is fixed, which is not conducive to the balance between exploration and exploitation. Based on this, we propose a new dynamic adaptation strategy to adjust the rotation angle of the quantum rotation gate. In the early exploration stage, the value of θ should be increased when the current individual is far from the best. In the exploitation stage, the value of θ should be decreased. This method allows the search process to adapt to different solutions and is more conducive to searching for the global optimum. In detail, this improved method determines the value of the rotation angle by the difference between the current individual's fitness and the best fitness that has been obtained so far. The rotation angle θ is defined as

$$\Delta\theta = \theta_{\min} + \gamma_i \cdot (\theta_{\max} - \theta_{\min}) \tag{13}$$

where $\theta_{\max}$ and $\theta_{\min}$ are the maximum and minimum values of the range of $\Delta\theta$, respectively. The maximum and minimum values take 0.035π and 0.001π, respectively. γ is defined as:

$$\gamma_i = 1 - e^{-4\cdot(\frac{bF - S(i)}{bF - wF})^2} \tag{14}$$

The pseudo-code of DQRG (Algorithm 2) is as follows:

Algorithm 2: Pseudo-code of the quantum rotation gate (DQRG).

```
Input: position $X_i$, fitness values of $X_i$ $fitness(i)$, the best fitness value $bF$, $dim$
Output: updated position $X$.
1  Initialize the parameters $\alpha$, $\beta$, $s$;
2  while $i < dim$ do
3      Update $\alpha$, $\beta$;
4      Compare the $fitness(i)$ and $bF$;
5      Update $s$ according to Table 1;
6      Update $\Delta\theta$ by Equation (13);
7      Perform DQRG by Equation (10);
8      $i = i + 1$;
9  end
10 return $X$
```

3.2. OBL

In this work, an improved method to obtain the opposite solution is proposed further. Specifically, instead of using only lower and upper bounds to find the opposite point, the impact of the current better solution, including the optimal, suboptimal, and third optimal solutions, is added to the opposite point's calculation procedure. The new formula of the opposite point is expressed as follows:

$$X_m = \frac{X_{os} + X_{ss} + X_{ts}}{3} \tag{15}$$

where X_m is the average of three better solutions, X_{os} is the current best solution, X_{ss} is the suboptimal solution, and X_{ts} is the third optimal solution.

$$X_{\mathrm{OBL}}(t+1) = LB + UB - X_m(t) + rand \cdot (X_m(t) - X(t)) \tag{16}$$

where $X_{\mathrm{OBL}}(t+1)$ is the improved opposite solution, $rand$ denotes the random value in the interval of $[0,1]$, and UB and LB are the upper and lower limits.

3.3. Improved SMA

To explore the solution space of complex optimization problems more efficiently, we propose two strategies based on the original SMA algorithm: the DQRG and OBL strategies. In the proposed method, two main conditions are considered to execute the proposed policy procedures. The first condition is the execution of SMA or two other strategies. If $r_2 < 0.8$, then SMA is executed to update the position. Otherwise, the second condition is checked to determine the strategy to adopt. If $r_3 < 0.5$ in the second condition, the solution will be updated using the DQRG; otherwise, OBL will be executed for the searched individual. The pseudo-code of the DQOBLSMA is shown as Algorithm 3:

Algorithm 3: Pseudo-code of the DQOBLSMA.

```
Input: Population size $N$, Maximum number of iteration $MaxIt$.
Output: The best location $X_b$, the best fitness value $bF$ .
1  Initialize the parameters popsize($N$), $Max_iteraition$;
2  Initialize the positions of slime mold $X_i (i = 1, 2, 3..., N)$ ;
3  while $t < MaxIt$ do
4      Calculate the fitness of all slime mold;
5      Update $bF$, $X_b$;
6      Calculate the $W$ by Equation (3);
7      foreach slime mold do
8          if $r_1 < 0.8$ then
9              if $r_2 < z$ then
10                 update the position using Equation (6);
11             else
12                 Update $p$, $vb$, and $vc$;
13                 Update position by Equation (1);
14             end
15         else
16             if $r_3 < 0.5$ then
17                 Operate Dynamic quantum rotation gate by Algorithm 2;
18             else
19                 Calculate opposition solution $X_{OBL}$ of individual $X$ by Equation
                     (16);
20             end
21         end
22     end
23     $t = t + 1$;
24 end
25 return bestFitness,$X_b$
```

3.4. Computational Complexity Analysis

The computational complexity of the DQOBLSMA depends on the population size (N), dimension size (D), and maximum iterations (T). First, the DQOBLSMA produces the search agents randomly in the search space, so the computational complexity is O($N \times D$). Second, the computational complexity of calculating the fitness of all agents is O(N). The quick-sort of all search agents is O($N \times \log N$). Moreover, updating the positions of agents in the original SMA is ($N \times D$). Therefore, the total computational complexity of original SMA is O($N \times D + N \times T \times (1 + D + \log N)$).

Updating the positions through the DQRG is O($N \times D$) (maximum), and the OBL is O(N) (maximum). Updating the position using DQRG and the original SMA will not be done simultaneously. In summary, the final time complexity is O(DQOBLSMA) = O($N \times D + N \times T \times (1 + D + \log N)$)(maximum). In summary, the improved strategy proposed in this paper does not increase the computational complexity when compared with the original SMA.

4. Experiments and Discussion

We conducted a series of experiments to verify the performance of the DQOBLSMA. The classical benchmark functions are introduced in Section 4.1. In the experiments of test functions, the impacts of two mechanisms were analyzed; see Section 4.2. In Section 4.3, the DQOBLSMA is compared with several advanced algorithms. In Section 4.4, the convergence of the algorithms is analyzed.

The performance of the DQOBLSMA was investigated using the mean result (Mean) and standard deviation (Std). In order to accurately make statistically reasonable conclusions, the results of the benchmark test functions were ranked using the Friedman test. In addition, the Wilcoxon's rank-sum test was used to assess the average performances of the algorithms in a statistical sense. In this study, it was used to test whether there was a difference in the effect of the DQOBLSMA compared with those of other algorithms in pairwise comparisons. When the p-value is less than 0.05, the result is significantly different from the other methods. The symbols "+," "−," and "=" indicate if the DQOBLSMA is better than, inferior to, or equal to the other algorithms, respectively.

4.1. Benchmark Function Validation and Parameter Settings

In this study, the test set for the DQOBLSMA comparison experiment was the 23 classical test functions that had been used in the literature [34]. The details are shown in Table 2. These classical test functions are divided into unimodal functions, multimodal functions, and fixed-dimension multimodal functions. The unimodal functions (F1–F7) have only one local solution and one optimal global solution and are usually used to evaluate the local exploitation ability of the algorithm. Multimodal functions (F8–F13) are often used to test the exploration ability of the algorithm. F14–F23 are fixed-dimensional multimodal functions with many local optimal points and low dimensionality, which can be used to evaluate the stability of the algorithm.

Table 2. The classic benchmark functions.

Function Type	Function	Name	Dimension	Range	Theoretical Value
Unimodal test functions	F1	Sphere	30	$[-100, 100]$	0
	F2	Schwefel 2.22	30	$[-10, 10]$	0
	F3	Schwefel 1.2	30	$[-100, 100]$	0
	F4	Schwefel 2.21	30	$[-100, 100]$	0
	F5	Rosenbrock	30	$[-30, 30]$	0
	F6	Step	30	$[-100, 100]$	0
Multimodal test functions	F7	Quartic	30	$[-1.28, 1.28]$	0
	F8	Schwefel 2.26	30	$[-500, 500]$	$-418.9829 \times D$
	F9	Rastrigin	30	$[-5.12, 5.12]$	0
	F10	Ackley	30	$[-32, 32]$	0
	F11	Griewank	30	$[-600, 600]$	0
	F12	Penalized	30	$[-50, 50]$	0
	F13	Penalized2	30	$[-50, 50]$	0
Fixed-dimension multimodal test functions	F14	Foxholes	2	$[-65, 65]$	0.998004
	F15	Kowalik	4	$[-5, 5]$	0.0003075
	F16	Six-Hump Camel Back	2	$[-5, 5]$	-1.03163
	F17	Branin	2	$[-5, 5]$	0.398
	F18	Goldstein Price	2	$[-2, 2]$	3
	F19	Hartman 3	3	$[-1, 2]$	-3.8628
	F20	Hartman 6	6	$[0, 1]$	-3.322
	F21	Shekel 5	4	$[0, 10]$	-10.1532
	F22	Shekel 7	4	$[0, 10]$	-10.4028
	F23	Shekel 10	4	$[0, 10]$	-10.5363

The DQOBLSMA has been compared to the original SMA and five other algorithms: the slime mold algorithm improved by opposition-based learning and Levy flight distribution (OBLSMAL) [48], the equilibrium slime mold algorithm (ESMA) [49], the equilibrium optimizer with a mutation strategy (MEO) [50], the adaptive differential evolution with an optional external archive (JADE) [51], and the gray wolf optimizer based on random walk (RWGWO) [52]. The parameter settings of each algorithm are shown in Table 3, and the experimental parameters for all optimization algorithms were chosen to be the same as those reported in the original works.

Table 3. Parameter settings for the comparative algorithms.

Algorithm	Parameter
OBLSMAL	$z = 0.03, p_1 = 0.5, p_2 = 0.5$
ESMA	$z = 0.03$
MEO	$a_1 = 2, a_2 = 1, GP = 0.5$
JADE	$\mu F = 0.5, \mu CR = 0.5, p = 0.1, c = 0.1$
RWGWO	Control parameter a, b decrease linearly from 2 to 0
SMA	$z = 0.03$

In order to maintain a fair comparison, each algorithm was independently run 30 times. The population size (N) and the maximum function evaluation times (FEs) of all experimental methods were fixed at 30 and 15,000, respectively. The comparative experiment was run under the same test conditions to keep the experimental conditions consistent. The proposed method was coded in Python3.8 and tested on a PC with an AMD R5-4600 Hz, 3.00 GHz of memory, 16 GB of RAM, and the Windows 11 operating system.

4.2. Impacts of Components

In this section, different versions of the improvement are investigated. The proposed DQOBLSMA adds two different mechanisms to the original SMA. To verify their respective effects, they are compared when separated. Different combinations between SMA and two mechanisms are listed below:

- SMA combined with DQRG and OBL (DQOBLSMA);
- SMA combined with DQRG (DQSMA);
- SMA combined with OBL(OBLSMA);
- Original SMA;

Table 4 gives the comparison results between the original SMA and the improved algorithm after adding the mechanism. The ranking of the four algorithms is given at the end of the table, and it can be seen that the first-ranked algorithm is the DQOBLSMA. This ranking was obtained using the Friedman ranking test [53] and reveals the overall performance rankings of the compared algorithms against the tested functions. In these cases, the ranking from best to worst was roughly as follows: DQOBLSMA > OBLSMA > SMA > DQSMA. With the addition of both mechanisms, the performance of the DQOBLSMA is more stable, and the global search capability is much improved. When comparing DQSMA with OBLSMA, we can see that OBLSMA is much stronger than DQSMA, indicating that the contribution of OBL to the performance of SMA is more significant than the contribution of DQRG to the performance of SMA. When comparing DQSMA with SMA, we can see that DQSMA becomes worse on unimodal functions but stronger on most multimodal and fixed-dimensional multimodal functions than the original SMA in terms of optimization.

Wilcoxon's rank-sum test was used to verify the significance of the DQOBLSMA against the original SMA and SMA with the addition of one mechanism. The results are shown in Table 5. Based on these results and those in Table 4, the DQOBLSMA outperformed SMA on 13 benchmark functions, DQSMA on 17 benchmark functions, and OBLSMA on 8 benchmark functions. Thus, the DQOBLSMA algorithm proposed in this paper combines DQRG with OBL. Although DQSMA and OBLSMA can both find the solutions, there are more benefits to be gained by combining the two strategies. In conclusion, the DQOBLSMA offers better optimization performance and is significantly better than SMA, DQSMA, and OBLSMA.

Table 4. Search results (comparisons of the DQOBLSMA, DQSMA, OBLSMA, SMA).

Function	DQOBLSMA		DQSMA		OBLSMA		SMA	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F1	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	1.0891×10^{-02}	4.3266×10^{-03}	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$
F2	2.9368×10^{-231}	$0.0000\times10^{+00}$	5.1658×10^{-02}	1.7908×10^{-02}	2.7971×10^{-244}	$0.0000\times10^{+00}$	7.2130×10^{-164}	$0.0000\times10^{+00}$
F3	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	3.8217×10^{-02}	5.9525×10^{-02}	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$
F4	1.4919×10^{-224}	$0.0000\times10^{+00}$	1.5620×10^{-02}	8.3506×10^{-03}	3.1204×10^{-229}	$0.0000\times10^{+00}$	5.3508×10^{-168}	$0.0000\times10^{+00}$
F5	1.4718×10^{-01}	1.5834×10^{-01}	$5.1129\times10^{+00}$	$1.1125\times10^{+01}$	$6.4059\times10^{+00}$	$1.1204\times10^{+01}$	$2.8202\times10^{+01}$	2.6986×10^{-01}
F6	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$
F7	8.8202×10^{-05}	7.2479×10^{-05}	5.8003×10^{-04}	3.1965×10^{-04}	1.3372×10^{-04}	8.9724×10^{-05}	2.3852×10^{-04}	2.0182×10^{-04}
F8	$-1.2569\times10^{+04}$	1.0234×10^{-01}	$-1.1726\times10^{+04}$	$1.0829\times10^{+03}$	$-1.2569\times10^{+04}$	5.6297×10^{-02}	$-9.1620\times10^{+03}$	$7.0236\times10^{+02}$
F9	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$
F10	4.4409×10^{-16}	$0.0000\times10^{+00}$	4.4409×10^{-16}	$0.0000\times10^{+00}$	4.4409×10^{-16}	$0.0000\times10^{+00}$	4.4409×10^{-16}	$0.0000\times10^{+00}$
F11	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	2.2635×10^{-02}	1.0138×10^{-02}	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$
F12	8.9207×10^{-04}	1.0608×10^{-03}	5.0820×10^{-03}	1.4513×10^{-02}	3.5431×10^{-03}	9.2857×10^{-03}	2.4763×10^{-02}	9.4810×10^{-03}
F13	1.4921×10^{-03}	3.6813×10^{-03}	4.2575×10^{-02}	7.6342×10^{-02}	2.2321×10^{-03}	8.3069×10^{-03}	5.0605×10^{-02}	3.4525×10^{-02}
F14	9.9800×10^{-01}	3.4807×10^{-13}	$1.1634\times10^{+00}$	5.9405×10^{-01}	9.9800×10^{-01}	2.0372×10^{-13}	9.9800×10^{-01}	2.0923×10^{-12}
F15	3.8029×10^{-04}	9.0820×10^{-05}	4.4014×10^{-04}	1.0909×10^{-04}	4.7568×10^{-04}	1.7299×10^{-04}	5.3389×10^{-04}	2.7098×10^{-04}
F16	$-1.0316\times10^{+00}$	4.1555×10^{-10}	$-1.0316\times10^{+00}$	5.1500×10^{-06}	$-1.0316\times10^{+00}$	8.8268×10^{-10}	$-1.0316\times10^{+00}$	1.4953×10^{-09}
F17	3.9789×10^{-01}	3.2851×10^{-08}	3.9794×10^{-01}	1.2457×10^{-04}	3.9789×10^{-01}	1.3247×10^{-07}	3.9789×10^{-01}	3.4597×10^{-07}
F18	$3.0000\times10^{+00}$	4.3415×10^{-07}	$3.0006\times10^{+00}$	5.3604×10^{-04}	$3.0000\times10^{+00}$	5.3937×10^{-08}	$3.0000\times10^{+00}$	3.3742×10^{-08}
F19	$-3.8628\times10^{+00}$	6.0439×10^{-07}	$-3.8628\times10^{+00}$	2.6271×10^{-05}	$-3.8627\times10^{+00}$	3.4507×10^{-04}	$-3.8628\times10^{+00}$	3.3028×10^{-07}
F20	$-3.2821\times10^{+00}$	5.7002×10^{-02}	$-3.2375\times10^{+00}$	6.5585×10^{-02}	$-3.2615\times10^{+00}$	6.0657×10^{-02}	$-3.2582\times10^{+00}$	5.9773×10^{-02}
F21	$-1.0153\times10^{+01}$	2.1496×10^{-04}	$-1.0152\times10^{+01}$	1.8979×10^{-03}	$-1.0153\times10^{+01}$	8.5453×10^{-05}	$-8.7668\times10^{+00}$	$2.7426\times10^{+00}$
F22	$-1.0403\times10^{+01}$	1.8317×10^{-04}	$-1.0402\times10^{+01}$	1.0712×10^{-03}	$-1.0403\times10^{+01}$	1.2865×10^{-04}	$-8.5645\times10^{+00}$	$2.8449\times10^{+00}$
F23	$-1.0536\times10^{+01}$	2.0415×10^{-04}	$-1.0534\times10^{+01}$	3.6030×10^{-03}	$-1.0536\times10^{+01}$	1.2450×10^{-04}	$-8.5593\times10^{+00}$	$2.8800\times10^{+00}$
Friedman test average rank	1.74		3.33		1.91		3.02	

Table 5. Test statistical results of Wilcoxon's rank-sum test.

Benchmark	DQOBLSMA vs. DQSMA		DQOBLSMA vs. OBLSMA		DQOBLSMA vs. SMA	
	p-Value	Winner	*p*-Value	Winner	*p*-Value	Winner
F1	2.87×10^{-11}	+	NaN	=	NaN	=
F2	2.87×10^{-11}	+	5.22×10^{-09}	−	1.94×10^{-09}	+
F3	2.87×10^{-11}	+	NaN	=	NaN	=
F4	2.87×10^{-11}	+	5.22×10^{-09}	−	1.48×10^{-09}	+
F5	NaN	+	6.24×10^{-03}	+	2.87×10^{-11}	+
F6	NaN	=	NaN	=	NaN	=
F7	1.63×10^{-08}	+	2.37×10^{-02}	+	1.73×10^{-04}	+
F8	2.87×10^{-11}	+	5.96×10^{-03}	=	2.87×10^{-11}	+
F9	2.87×10^{-11}	=	NaN	=	NaN	=
F10	2.87×10^{-11}	=	NaN	=	2.87×10^{-11}	=
F11	2.87×10^{-11}	+	NaN	=	NaN	=
F12	NaN	+	4.59×10^{-02}	+	2.87×10^{-11}	+
F13	7.90×10^{-05}	+	NaN	+	3.88×10^{-11}	+
F14	2.87×10^{-11}	+	NaN	=	2.87×10^{-11}	=
F15	6.8×10^{-3}	+	3.09×10^{-02}	+	2.82×10^{-03}	+
F16	2.87×10^{-11}	=	NaN	=	5.10×10^{-05}	+
F17	2.87×10^{-11}	+	NaN	=	6.37×10^{-04}	=
F18	2.87×10^{-11}	=	NaN	+	NaN	=
F19	1.31×10^{-07}	=	5.12×10^{-04}	+	3.50×10^{-08}	=
F20	1.15×10^{-06}	+	NaN	+	3.76×10^{-03}	+
F21	6.81×10^{-09}	+	1.41×10^{-03}	=	2.33×10^{-09}	+
F22	8.12×10^{-09}	+	4.44×10^{-02}	=	6.26×10^{-08}	+
F23	1.54×10^{-10}	+	1.72×10^{-03}	=	1.55×10^{-06}	+
+/−/=	17/0/6		8/2/13		f13/0/10	

4.3. Benchmark Function Experiments

As seen from Table 6, on unimodal benchmark functions (F1–F7), the DQOBLSMA can achieve better results than other optimization algorithms. For F1, F3, and F6, the DQOBLSMA could find the theoretical optimal value. For all unimodal functions, the DQOBLSMA obtained the smallest mean values and standard deviations compared to other algorithms, showing the best accuracy and stability.

From the results shown in Tables 7 and 8, the DQOBLSMA outperformed the other algorithms for most of the multimodal and fixed-dimensional multimodal functions. For the multimodal functions F8–F13, the DQOBLSMA obtained almost all the best mean and standard deviation values, and obtainedthe global optimal solution for four functions (F8–F11). As shown in Table 8, the DQOBLSMA obtained theoretically optimal values in 8 of the 10 fixed-dimensional multimodal functions (F14–F23). Although the DQOBLSMA did not outperform JADE in F14–F23, it exceeded ESMA and OBLSMAL in overall performance. These results show that the DQOBLSMA also provides powerful and robust exploitation capabilities.

In addition, Table 9 presents Wilcoxon's rank-sum test results to verify the significant differences between the DQOBLSMA and the other five algorithms. It is worth noting that *p*-values less than 0.05 mean significant differences between the respective pairs of compared algorithms. The DQOBLSMA outperformed all other algorithms to varying degrees, and outperformed OBLSMAL, ESMA, MEO, JADE, and RWGWO, on 14, 15, 16, 15, and 18 benchmark functions, respectively. Table 10 shows the statistical results of the Friedman test, where the DQOBLSMA ranked first in F1–F7 and F8–F13 and second after JADE by a small margin in F14–F23. The DQOBLSMA received the best ranking overall. In summary, the DQOBLSMA provided better results on almost all benchmark functions than the other algorithms.

Table 6. Results of unimodal benchmark test functions.

Func	Criteria	DQOBLSMA	OBLSMAL	ESMA	MEO	JADE	RWGWO
F1	Best	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	1.0936×10^{-54}	7.1160×10^{-14}	7.2435×10^{-73}
	Mean	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	1.3473×10^{-51}	1.3924×10^{-12}	9.9351×10^{-65}
	Worst	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	1.1718×10^{-50}	8.3623×10^{-12}	2.8903×10^{-63}
	Std	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	3.4983×10^{-51}	2.2303×10^{-12}	6.9913×10^{-64}
F2	Best	1.6860×10^{-280}	1.8971×10^{-126}	1.2829×10^{-179}	1.2944×10^{-32}	8.7945×10^{-08}	8.4261×10^{-52}
	Mean	2.9368×10^{-231}	7.4709×10^{-113}	4.1210×10^{-175}	6.2425×10^{-31}	4.5037×10^{-06}	1.2077×10^{-47}
	Worst	8.8104×10^{-230}	2.1489×10^{-111}	8.3686×10^{-174}	2.2747×10^{-30}	7.1303×10^{-05}	7.7841×10^{-47}
	Std	$0.0000\times10^{+00}$	5.1975×10^{-112}	$0.0000\times10^{+00}$	7.6873×10^{-31}	1.7166×10^{-05}	2.3314×10^{-47}
F3	Best	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	1.3923×10^{-278}	3.5006×10^{-21}	$3.4830\times10^{+00}$	$2.2232\times10^{+03}$
	Mean	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	7.5255×10^{-205}	1.7481×10^{-17}	$2.1172\times10^{+01}$	$6.0553\times10^{+03}$
	Worst	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	2.2576×10^{-203}	1.1762×10^{-16}	$5.6794\times10^{+01}$	$1.1356\times10^{+04}$
	Std	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	3.4991×10^{-17}	$1.6585\times10^{+01}$	$2.4311\times10^{+03}$
F4	Best	2.2279×10^{-273}	9.2369×10^{-122}	8.7764×10^{-173}	6.6498×10^{-15}	1.2412×10^{-01}	8.3027×10^{-07}
	Mean	1.4919×10^{-224}	1.3337×10^{-106}	1.4870×10^{-162}	5.1881×10^{-13}	6.6358×10^{-01}	$2.1528\times10^{+00}$
	Worst	4.4756×10^{-223}	3.9116×10^{-105}	4.2072×10^{-161}	5.2753×10^{-12}	$1.7608\times10^{+00}$	$2.9139\times10^{+01}$
	Std	$0.0000\times10^{+00}$	9.4644×10^{-106}	1.0186×10^{-161}	1.2870×10^{-12}	4.2471×10^{-01}	$7.2432\times10^{+00}$
F5	Best	4.8100×10^{-04}	$2.6149\times10^{+01}$	$2.3534\times10^{+01}$	$2.5670\times10^{+01}$	$1.5204\times10^{+01}$	$2.8626\times10^{+01}$
	Mean	1.4718×10^{-01}	$2.7476\times10^{+01}$	$2.7593\times10^{+01}$	$2.6755\times10^{+01}$	$3.4093\times10^{+01}$	$2.8807\times10^{+01}$
	Worst	5.6207×10^{-01}	$2.8866\times10^{+01}$	$2.8973\times10^{+01}$	$2.8759\times10^{+01}$	$9.3404\times10^{+01}$	$2.8898\times10^{+01}$
	Std	1.5834×10^{-01}	8.0237×10^{-01}	$1.5217\times10^{+00}$	7.3628×10^{-01}	$2.4394\times10^{+01}$	6.2979×10^{-02}
F6	Best	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$
	Mean	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	6.6667×10^{-02}	$0.0000\times10^{+00}$
	Worst	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$1.0000\times10^{+00}$	$0.0000\times10^{+00}$
	Std	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	2.9152×10^{-01}	$0.0000\times10^{+00}$
F7	Best	3.2865×10^{-07}	5.4366×10^{-07}	3.6036×10^{-05}	8.8161×10^{-06}	9.3734×10^{-03}	2.1106×10^{-05}
	Mean	8.8202×10^{-05}	2.1700×10^{-04}	2.0721×10^{-04}	3.7390×10^{-04}	1.8194×10^{-02}	1.7522×10^{-02}
	Worst	2.6899×10^{-04}	1.0933×10^{-03}	6.4167×10^{-04}	1.5364×10^{-03}	2.6649×10^{-02}	1.8120×10^{-01}
	Std	7.2479×10^{-05}	2.6392×10^{-04}	1.7316×10^{-04}	4.1133×10^{-04}	4.8117×10^{-03}	4.3922×10^{-02}

Table 7. Results of multi-modal benchmark functions.

Func	Criteria	DQOBLSMA	OBLSMAL	ESMA	MEO	JADE	RWGWO
F8	Best	$-1.2569\times10^{+04}$	$-8.8602\times10^{+03}$	$-9.8908\times10^{+03}$	$-5.4647\times10^{+03}$	$-1.1856\times10^{+04}$	$-9.3674\times10^{+03}$
	Mean	$-1.2569\times10^{+04}$	$-7.0233\times10^{+03}$	$-8.5070\times10^{+03}$	$-3.7623\times10^{+03}$	$-1.0905\times10^{+04}$	$-8.8801\times10^{+03}$
	Worst	$-1.2569\times10^{+04}$	$-5.4879\times10^{+03}$	$-6.4963\times10^{+03}$	$-3.0199\times10^{+03}$	$-6.8045\times10^{+03}$	$-8.0571\times10^{+03}$
	Std	1.0234×10^{-01}	$7.7253\times10^{+02}$	$8.5477\times10^{+02}$	$5.5378\times10^{+02}$	$1.6276\times10^{+03}$	$3.2772\times10^{+02}$
F9	Best	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$
	Mean	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	9.4739×10^{-15}	$0.0000\times10^{+00}$
	Worst	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	1.4744×10^{-13}	$0.0000\times10^{+00}$
	Std	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	3.7368×10^{-14}	$0.0000\times10^{+00}$
F10	Best	4.4409×10^{-16}	4.4409×10^{-16}	4.4409×10^{-16}	4.4409×10^{-16}	7.7624×10^{-08}	4.4409×10^{-16}
	Mean	4.4409×10^{-16}	4.4409×10^{-16}	4.4409×10^{-16}	4.4409×10^{-16}	3.8505×10^{-02}	3.5231×10^{-15}
	Worst	4.4409×10^{-16}	4.4409×10^{-16}	4.4409×10^{-16}	4.4409×10^{-16}	$1.1551\times10^{+00}$	3.9968×10^{-15}
	Std	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	2.7968×10^{-01}	1.2900×10^{-15}
F11	Best	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	7.2387×10^{-14}	$0.0000\times10^{+00}$
	Mean	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	4.3486×10^{-03}	$0.0000\times10^{+00}$
	Worst	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	3.6770×10^{-02}	$0.0000\times10^{+00}$
	Std	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	$0.0000\times10^{+00}$	9.9297×10^{-03}	$0.0000\times10^{+00}$
F12	Best	5.8098×10^{-08}	1.7590×10^{-02}	2.8002×10^{-02}	1.1540×10^{-02}	4.4220×10^{-14}	2.9591×10^{-02}
	Mean	8.9207×10^{-04}	4.3823×10^{-02}	9.0114×10^{-02}	4.6612×10^{-02}	4.4934×10^{-02}	1.0348×10^{-01}
	Worst	3.6337×10^{-03}	1.2371×10^{-01}	4.2696×10^{-01}	8.6796×10^{-02}	4.1469×10^{-01}	7.3880×10^{-01}
	Std	1.0608×10^{-03}	2.4564×10^{-02}	1.0394×10^{-01}	2.0035×10^{-02}	1.2955×10^{-01}	1.6264×10^{-01}
F13	Best	7.3229×10^{-06}	2.4407×10^{-01}	2.5338×10^{-01}	4.5254×10^{-01}	4.1497×10^{-14}	5.6485×10^{-01}
	Mean	1.4921×10^{-03}	$1.0518\times10^{+00}$	7.9118×10^{-01}	8.9529×10^{-01}	2.3516×10^{-10}	$1.1565\times10^{+00}$
	Worst	1.1660×10^{-02}	$2.6596\times10^{+00}$	$1.4767\times10^{+00}$	$1.2682\times10^{+00}$	2.9604×10^{-09}	$2.3763\times10^{+00}$
	Std	3.6813×10^{-03}	6.9507×10^{-01}	3.3716×10^{-01}	2.2110×10^{-01}	7.8166×10^{-10}	4.1169×10^{-01}

Table 8. Results of fixed-dimension multi-modal benchmark functions.

Func	Criteria	DQOBLSMA	OBLSMAL	ESMA	MEO	JADE	RWGWO
F14	Best	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}	$1.0937\times10^{+00}$	9.9800×10^{-01}	9.9800×10^{-01}
	Mean	9.9800×10^{-01}	$1.1304\times10^{+00}$	$1.0641\times10^{+00}$	$5.7783\times10^{+00}$	9.9800×10^{-01}	$1.7229\times10^{+00}$
	Worst	9.9800×10^{-01}	$2.9821\times10^{+00}$	$2.9821\times10^{+00}$	$1.2671\times10^{+01}$	9.9800×10^{-01}	$5.9288\times10^{+00}$
	Std	3.4807×10^{-13}	5.2272×10^{-01}	4.8038×10^{-01}	$3.9055\times10^{+00}$	2.7756×10^{-17}	$1.6192\times10^{+00}$
F15	Best	3.0958×10^{-04}	3.0772×10^{-04}	5.8084×10^{-04}	3.0894×10^{-04}	3.0749×10^{-04}	4.1151×10^{-04}
	Mean	3.8029×10^{-04}	8.3277×10^{-04}	8.3114×10^{-04}	3.4423×10^{-03}	1.7361×10^{-03}	1.1214×10^{-03}
	Worst	6.3781×10^{-04}	1.2548×10^{-03}	1.2249×10^{-03}	2.0363×10^{-02}	2.0363×10^{-02}	2.6665×10^{-03}
	Std	9.0820×10^{-05}	3.3167×10^{-04}	2.1318×10^{-04}	7.2217×10^{-03}	5.8251×10^{-03}	5.9580×10^{-04}
F16	Best	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$
	Mean	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$
	Worst	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0316\times10^{+00}$	$-1.0307\times10^{+00}$
	Std	4.1555×10^{-10}	3.1460×10^{-08}	2.6995×10^{-10}	1.7352×10^{-10}	6.5564×10^{-16}	2.3316×10^{-04}
F17	Best	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}
	Mean	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9792×10^{-01}
	Worst	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9803×10^{-01}
	Std	3.2851×10^{-08}	1.4951×10^{-07}	3.3940×10^{-08}	5.0177×10^{-09}	$0.0000\times10^{+00}$	4.2346×10^{-05}
F18	Best	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$
	Mean	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0029\times10^{+00}$
	Worst	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0000\times10^{+00}$	$3.0315\times10^{+00}$
	Std	4.3415×10^{-07}	7.5551×10^{-07}	4.5784×10^{-11}	1.0614×10^{-05}	1.5740×10^{-15}	8.2802×10^{-03}
F19	Best	$-3.8628\times10^{+00}$	$-3.8628\times10^{+00}$	$-3.8628\times10^{+00}$	$-3.8626\times10^{+00}$	$-3.8628\times10^{+00}$	$-3.8628\times10^{+00}$
	Mean	$-3.8628\times10^{+00}$	$-3.8628\times10^{+00}$	$-3.8627\times10^{+00}$	$-3.8589\times10^{+00}$	$-3.8628\times10^{+00}$	$-3.8520\times10^{+00}$
	Worst	$-3.8628\times10^{+00}$	$-3.8628\times10^{+00}$	$-3.8616\times10^{+00}$	$-3.8549\times10^{+00}$	$-3.8628\times10^{+00}$	$-3.7967\times10^{+00}$
	Std	6.0439×10^{-07}	7.0025×10^{-06}	2.8826×10^{-04}	2.7955×10^{-03}	2.6226×10^{-15}	1.7232×10^{-02}
F20	Best	$-3.3220\times10^{+00}$	$-3.3220\times10^{+00}$	$-3.3220\times10^{+00}$	$-3.3220\times10^{+00}$	$-3.3220\times10^{+00}$	$-3.2948\times10^{+00}$
	Mean	$-3.2821\times10^{+00}$	$-3.2220\times10^{+00}$	$-3.2313\times10^{+00}$	$-3.2590\times10^{+00}$	$-3.2903\times10^{+00}$	$-3.1655\times10^{+00}$
	Worst	$-3.1999\times10^{+00}$	$-3.1985\times10^{+00}$	$-3.0851\times10^{+00}$	$-3.0633\times10^{+00}$	$-3.2031\times10^{+00}$	$-2.9487\times10^{+00}$
	Std	5.7002×10^{-02}	4.6895×10^{-02}	6.8225×10^{-02}	9.1478×10^{-02}	5.3456×10^{-02}	1.0946×10^{-01}
F21	Best	$-1.0153\times10^{+01}$	$-1.0153\times10^{+01}$	$-1.0153\times10^{+01}$	$-5.1609\times10^{+00}$	$-1.0153\times10^{+01}$	$-1.0148\times10^{+01}$
	Mean	$-1.0153\times10^{+01}$	$-9.9934\times10^{+00}$	$-9.3978\times10^{+00}$	$-5.0587\times10^{+00}$	$-9.3166\times10^{+00}$	$-7.0389\times10^{+00}$
	Worst	$-1.0152\times10^{+01}$	$-7.5756\times10^{+00}$	$-2.6300\times10^{+00}$	$-5.0552\times10^{+00}$	$-2.6305\times10^{+00}$	$-5.0064\times10^{+00}$
	Std	2.1496×10^{-04}	6.7461×10^{-01}	$2.4872\times10^{+00}$	2.5592×10^{-02}	$2.4162\times10^{+00}$	$2.4611\times10^{+00}$
F22	Best	$-1.0403\times10^{+01}$	$-1.0403\times10^{+01}$	$-1.0403\times10^{+01}$	$-8.1136\times10^{+00}$	$-1.0403\times10^{+01}$	$-1.0391\times10^{+01}$
	Mean	$-1.0403\times10^{+01}$	$-9.1689\times10^{+00}$	$-9.3977\times10^{+00}$	$-5.2039\times10^{+00}$	$-9.7170\times10^{+00}$	$-7.0943\times10^{+00}$
	Worst	$-1.0402\times10^{+01}$	$-2.7484\times10^{+00}$	$-2.7495\times10^{+00}$	$-2.5429\times10^{+00}$	$-2.7496\times10^{+00}$	$-2.7426\times10^{+00}$
	Std	1.8317×10^{-04}	$2.8605\times10^{+00}$	$2.6374\times10^{+00}$	$1.0562\times10^{+00}$	$2.3627\times10^{+00}$	$2.9401\times10^{+00}$
F23	Best	$-1.0536\times10^{+01}$	$-1.0536\times10^{+01}$	$-1.0536\times10^{+01}$	$-1.0536\times10^{+01}$	$-1.0536\times10^{+01}$	$-1.0536\times10^{+01}$
	Mean	$-1.0536\times10^{+01}$	$-8.8867\times10^{+00}$	$-9.2412\times10^{+00}$	$-7.4582\times10^{+00}$	$-1.0536\times10^{+01}$	$-6.4373\times10^{+00}$
	Worst	$-1.0536\times10^{+01}$	$-2.4177\times10^{+00}$	$-2.4216\times10^{+00}$	$-5.1285\times10^{+00}$	$-1.0536\times10^{+01}$	$-2.4270\times10^{+00}$
	Std	2.0415×10^{-04}	$3.1383\times10^{+00}$	$3.0570\times10^{+00}$	$2.5657\times10^{+00}$	1.9610×10^{-15}	$2.5968\times10^{+00}$

Table 9. Test statistical results of Wilcoxon's rank-sum test.

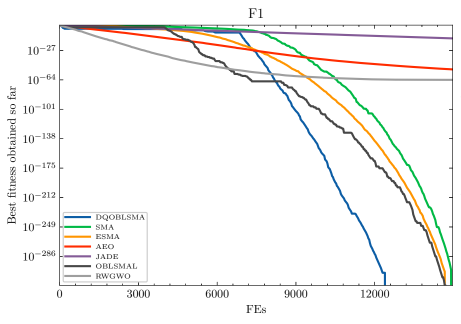
Benchmark	DQOBLSMA vs. OBLSMAL		DQOBLSMA vs. ESMA		DQOBLSMA vs. MEO		DQOBLSMA vs. JADE		DQOBLSMA vs. RWGWO	
	p-Value	Winner	*p*-Value	Winner	*p*-Value	Winner	*p*-Value	Winner	*p*-Value	Winner
F1	NaN	=	NaN	=	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+
F2	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+
F3	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+
F4	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+
F5	1.73×10^{-06}	+	1.73×10^{-06}	+	7.04×10^{-01}	+	1.73×10^{-06}	+	1.73×10^{-06}	+
F6	NaN	=	NaN	=	NaN	=	NaN	+	NaN	=
F7	2.41×10^{-03}	+	4.20×10^{-04}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	9.32×10^{-06}	+
F8	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+	1.73×10^{-06}	+
F9	NaN	=	NaN	=	NaN	=	1.20×10^{-02}	+	NaN	=
F10	NaN	=	NaN	=	NaN	=	1.73×10^{-06}	+	6.39×10^{-07}	+
F11	NaN	=	NaN	=	NaN	=	1.73×10^{-06}	+	NaN	=
F12	1.73×10^{-06}	+	1.73×10^{-06}	+	2.61×10^{-04}	+	NaN	+	1.73×10^{-06}	+
F13	1.73×10^{-06}	+	1.73×10^{-06}	+	2.07×10^{-02}	+	1.73×10^{-06}	−	1.73×10^{-06}	+
F14	2.61×10^{-04}	+	2.71×10^{-01}	+	1.73×10^{-06}	+	1.73×10^{-06}	−	1.73×10^{-06}	+
F15	6.34×10^{-06}	+	1.73×10^{-06}	+	3.00×10^{-02}	+	NaN	+	1.73×10^{-06}	+
F16	4.29×10^{-06}	=	1.17×10^{-02}	−	4.73×10^{-06}	−	1.73×10^{-06}	=	1.73×10^{-06}	=
F17	7.51×10^{-05}	=	NaN	=	4.86×10^{-05}	−	1.73×10^{-06}	=	1.73×10^{-06}	−
F18	3.88×10^{-04}	=	9.32×10^{-06}	−	5.75×10^{-06}	=	1.73×10^{-06}	−	1.73×10^{-06}	+
F19	7.71×10^{-04}	=	1.74×10^{-04}	+	1.73×10^{-06}	+	1.73×10^{-06}	−	1.73×10^{-06}	+
F20	1.89×10^{-04}	+	8.94×10^{-04}	+	1.73×10^{-06}	+	1.75×10^{-02}	−	1.74×10^{-04}	+
F21	1.73×10^{-06}	+	4.29×10^{-06}	+	1.73×10^{-06}	+	1.48×10^{-02}	+	1.73×10^{-06}	+
F22	5.22×10^{-06}	+	1.02×10^{-05}	+	1.73×10^{-06}	+	2.77×10^{-03}	+	1.73×10^{-06}	+
F23	5.22×10^{-06}	+	1.38×10^{-03}	+	1.73×10^{-06}	+	1.73×10^{-06}	−	1.73×10^{-06}	+
+/−/=	14/0/9		15/2/6		16/2/5		15/6/2		18/1/4	

Table 10. Test statistical results of the Friedman test.

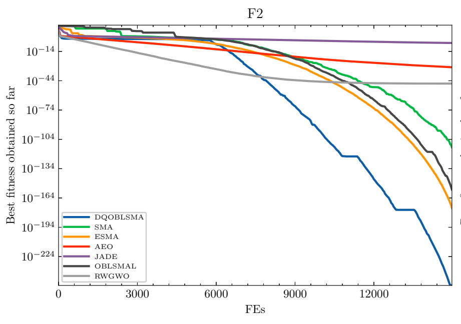
Func	DQOBLSMA	OBLSMAL	ESMA	MEO	JADE	RWGWO
F1–F7	1.36	3	2.36	3.86	5.71	4.71
F8–F13	2.08	3.42	3.42	3.75	4	4.33
F14–23	2.45	3.7	3	4.5	1.85	5.5
F1–F23	2.02	3.41	2.91	4.11	3.59	4.96

4.4. Convergence Analysis

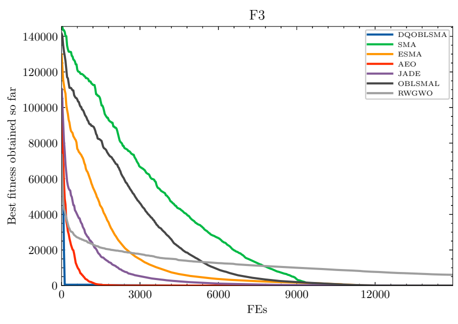
To demonstrate the effectiveness of the proposed DQOBLSMA, Figure 2 shows the convergence curves of the DQOBLSMA, SMA, ESMA, AEO, JADE, and RWGWO for the classical benchmark functions. The convergence curves show that the initial convergence of the DQOBLSMA was the fastest in most cases, except for $F6, F9, F10$, and $F11$; and RWGWO had faster initial convergence for these functions. For $F16$–$F20$, all comparison algorithms converged quickly to the global optimum, and the DQOBLSMA did not show a significant advantage. In Figure 2, a step or cliff drop in the DQOBLSMA's convergence curve can be observed, which indicates outstanding exploration capability. In almost all test cases, the DQOBLSMA had a better convergence rate than SMA and SMA variants, indicating that the SMA's convergence results can be significantly improved when applying the proposed search strategies. In conclusion, the DQOBLSMA is not only robust and effective at producing the best results, but also has a higher convergence speed than the other algorithms.



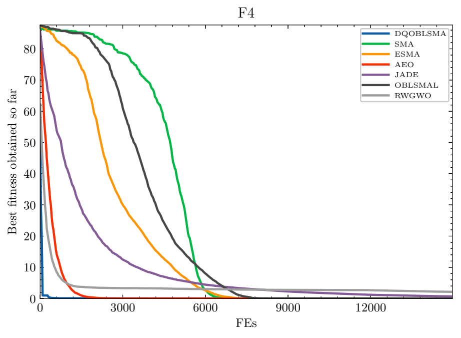
(**a**) Convergence curve of F1



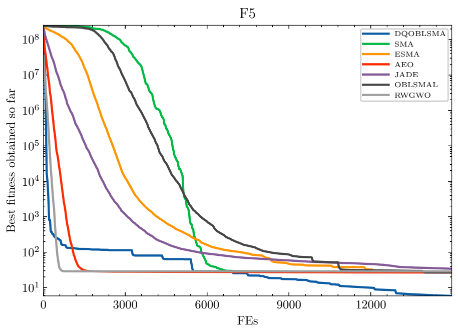
(**b**) Convergence curve of F2



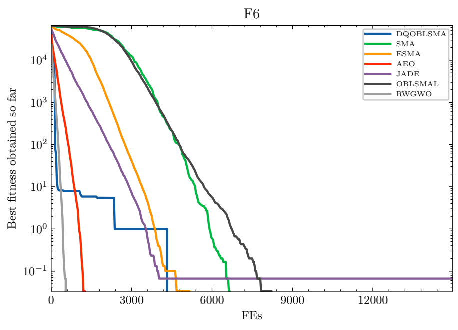
(**c**) Convergence curve of F3



(**d**) Convergence curve of F4

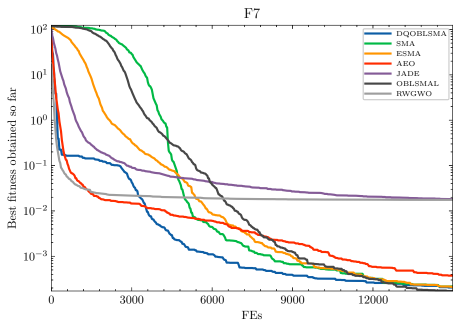


(**e**) Convergence curve of F5

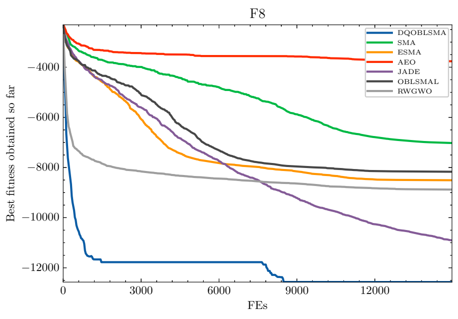


(**f**) Convergence curve of F6

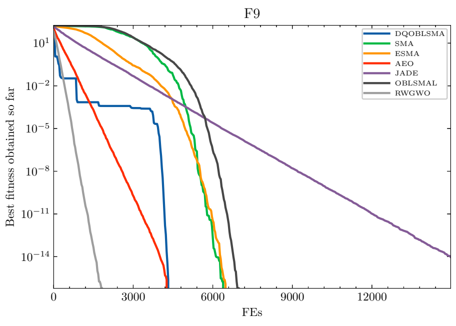
Figure 2. *Cont.*



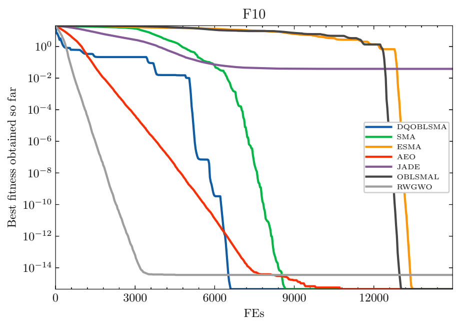
(**g**) Convergence curve of F7



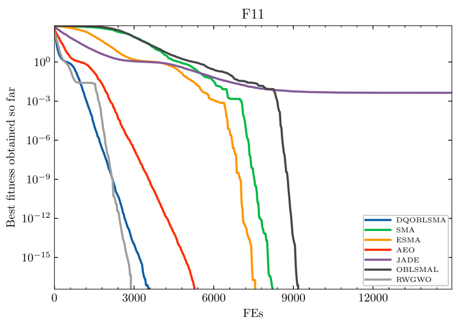
(**h**) Convergence curve of F8



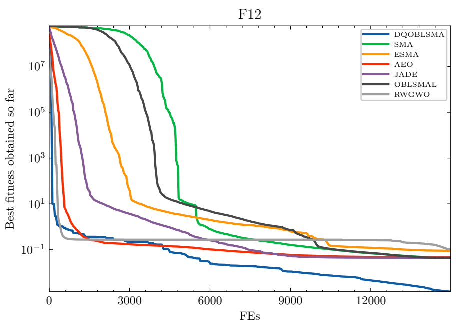
(**i**) Convergence curve of F9



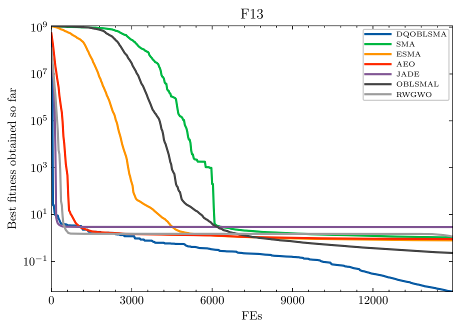
(**j**) Convergence curve of F10



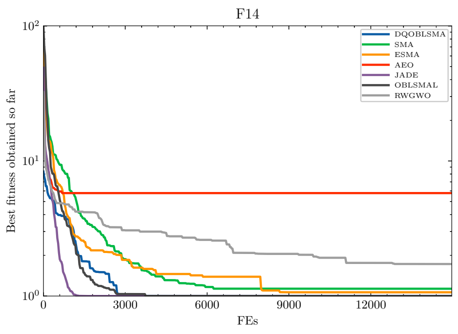
(**k**) Convergence curve of F11



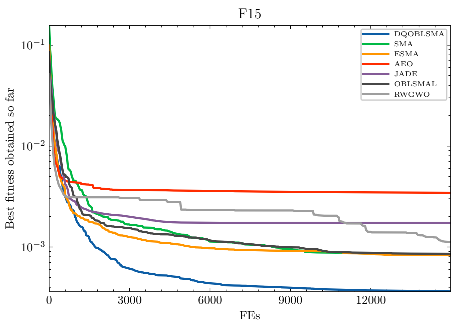
(**l**) Convergence curve of F12



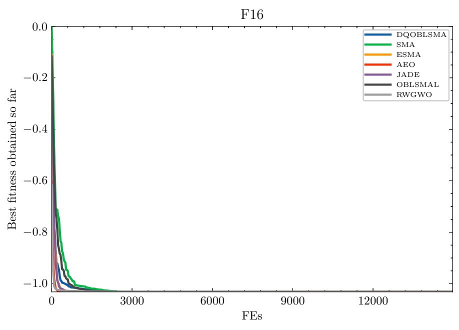
(**m**) Convergence curve of F13



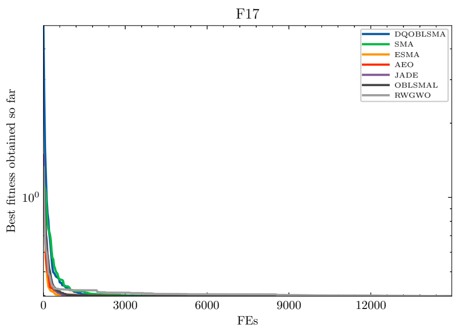
(**n**) Convergence curve of F14



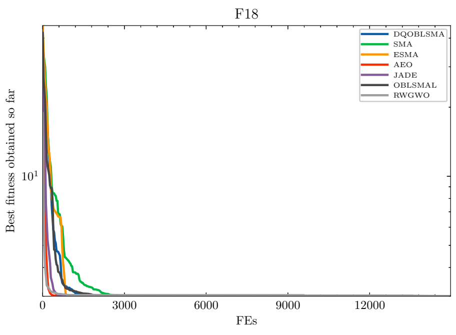
(**o**) Convergence curve of F15



(**p**) Convergence curve of F16

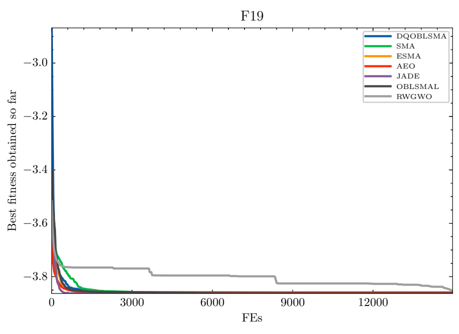


(**q**) Convergence curve of F17

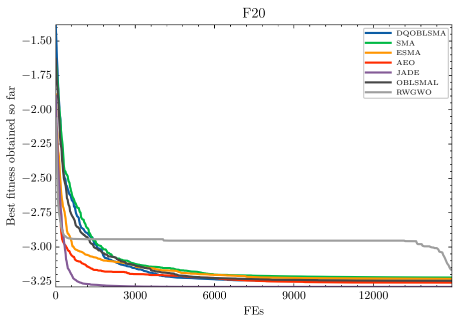


(**r**) Convergence curve of F18

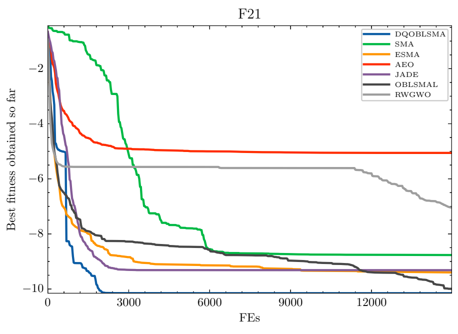
Figure 2. *Cont.*



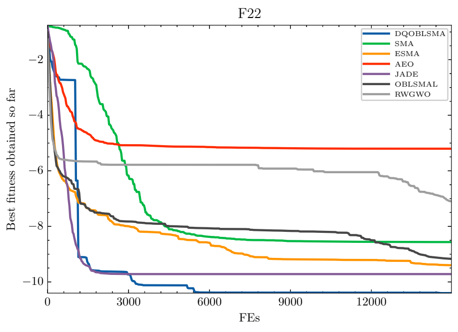
(**s**) Convergence curve of F19



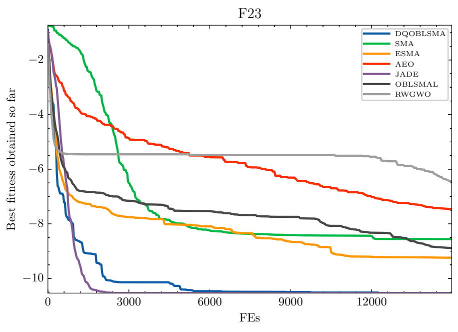
(**t**) Convergence curve of F20



(**u**) Convergence curve of F21



(**v**) Convergence curve of F22



(**w**) Convergence curve of F23

Figure 2. Convergence figures on test functions F1–F23.

5. Engineering Design Problems

In this section, the DQOBLSMA is evaluated using three engineering design problems: the welded beam design problem, tension/compression springs, and the pressure vessel design problem. These engineering problems are well known and have been widely used to verify the effectiveness of methods for solving complex real-world problems [54]. The proposed method is compared with the state-of-the-art algorithms: OBLSMAL, ESMA, MEO, JADE, and RWGWO. The population size (N) and the maximum number of iterations were fixed at 30 and 500 for all comparison algorithms.

5.1. Welded Beam Design Problem

The design diagram for the structural problem of a welded beam [55] is shown in Figure 3. The objective of structural design optimization of welded beams is to minimize the total cost, subject to certain constraints, which are the shear stress τ, the bending stress σ on the beam, the buckling load P_c, and the deflection δ of beam. Four variables are considered in this problem, welded thickness (h), the bar length (l), bar height (t), and the thickness of the bar (b).

The mathematical equations of this problem are shown below:

Consider:

$$\mathbf{x} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b];$$

minimize:

$$f(\mathbf{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2);$$

subject to:

$$g_1(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} - \tau_{max} \leq 0;$$
$$g_2(\mathbf{x}) = \frac{6PL}{x_3^2 x_4} - \sigma_{max} \leq 0;$$
$$g_3(\mathbf{x}) = x_1 - x_4 \leq 0;$$
$$g_4(\mathbf{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0;$$
$$g_5(\mathbf{x}) = 0.125 - x_1 \leq 0;$$
$$g_6(\mathbf{x}) = \frac{4PL^3}{Ex_3^3 x_4} - \delta_{max} \leq 0;$$
$$g_7(\mathbf{x}) = P - \frac{4.013Ex_3x_4^3}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \leq 0;$$

where:

$$\tau' = \frac{P}{2x_1x_2}, \tau'' = MRJ, M = P(L + \frac{x_2}{2}),$$
$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$
$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, P = 6000lb,$$
$$L = 14in, E = 30 \times 10^6 psi, G = 12 \times 10^6 psi,$$
$$\tau_{max} = 13600psi, \sigma_{max} = 30000psi,$$
$$\delta_{max} = 0.25in;$$

range of variables:

$$0.1 \leq x_1, x_4 \leq 2.0 \; and \; 0.1 \leq x_2, x_3 \leq 10.0$$

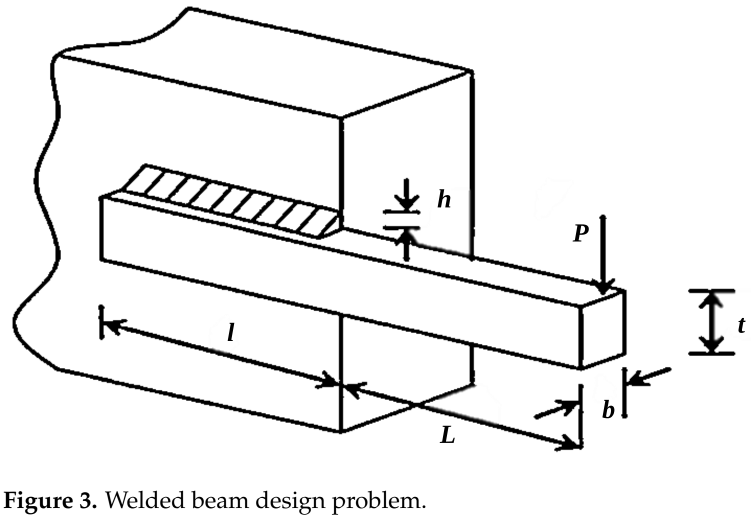


Figure 3. Welded beam design problem.

In Table 11, the results of the proposed DQOBLSMA and other well-known comparative optimization algorithms are given. It is clear from Table 11 that the proposed DQOBLSMA provides promising results for the optimal variables compared to other well-known optimization algorithms. The DQOBLSMA obtained a minimum cost of 1.695436 when h = 0.205598, l = 3.255605 , t = 9.036367, and b = 0.205741.

Table 11. Comparison in welded beam design.

Algorithm	**Optimal Values for Variables**				**Optimal Cost**
	h	***l***	***t***	***b***	
DQOBLSMA	0.205598	3.255605	9.036367	0.205741	1.695436
OBLSMAL	0.253062	1.842203	8.270240	0.253229	1.726511
ESMA	0.201567	3.357515	8.983361	0.208407	1.712227
SMA	0.197433	3.407377	9.036868	0.205729	1.703704
MEO	0.194411	3.487386	9.040436	0.205984	1.712024
JADE	0.205734	3.253036	9.036624	0.205730	1.695245
RWGWO	0.247585	3.000055	8.090046	0.256700	1.901643

5.2. Tension/Compression Spring Design

The design goal for extension/compression springs [56] is to obtain the minimum optimum weight under four constraints: deviation (g_1), shear stress (g_2), surge frequency (g_3), and deflection (g_4). As shown in Figure 4, three variables need to be considered. They are the wire diameter (d), the mean coil diameter (D), and the number of active coils (N). The mathematical description of this problem is given below:

Consider:

$$\mathbf{x} = [x_1\ x_2\ x_3] = [d\ D\ N];$$

minimize:

$$f(\mathbf{x}) = x_1^2 x_2 (2 + x_3);$$

subject to:

$$\begin{aligned} g_1(\mathbf{x}) &= 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0; \\ g_2(\mathbf{x}) &= \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} \leq 0; \\ g_3(\mathbf{x}) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0; \\ g_4(\mathbf{x}) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0; \end{aligned}$$

range of variables:

$$0.05 \leq x_1 \leq 2.0, 0.25 \leq x_2 \leq 1.3, \textit{and}\ 2.0 \leq x_3 \leq 15.0.$$

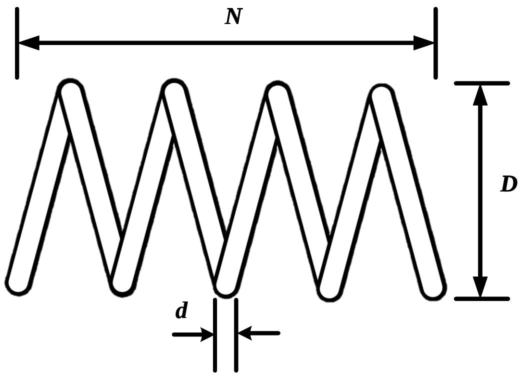


Figure 4. Tension/compression spring design problem.

The results of the DQOBLSMA and other comparative algorithms are presented in Table 12. The proposed DQOBLSMA achieved the best solution to the problem. The DQOBLSMA obtained a minimum cost of 0.012719 when d = 0.050000, D = 0.317425, and N = 14.028013.

Table 12. Comparison for the tension/compression spring design problem.

Algorithm	Optimal Values for Variables			Optimal Cost
	d	***D***	***N***	
DQOBLSMA	0.050000	0.317425	14.028013	0.012719
OBLSMAL	0.050000	0.317409	14.030650	0.012721
ESMA	0.051458	0.353086	12.050995	0.012739
SMA	0.050000	0.317317	14.042338	0.012726
MEO	0.057203	0.514683	7.661607	0.014002
JADE	0.055015	0.442128	7.613118	0.012864
RWGWO	0.056389	0.480684	6.712235	0.013316

5.3. Pressure Vessel Design

The pressure vessel design problem is a four-variable, four-constraint problem in the industry field that aims to reduce the total cost of a given cylindrical pressure vessel [57]. The four variables studied include the width of the shell (Ts), the width of the head (Th), the inner radius (R), and the length of the cylindrical section (L), as shown in Figure 5. The formulation of objective functions and four optimization constraints can be described as follows:

Consider:

$$\mathbf{x} = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L];$$

minimize:

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3;$$

subject to:

$$\begin{aligned} g_1(x) &= -x_1 + 0.0193x_3 \le 0 \\ g_2(x) &= -x_3 + 0.00954x_3 \le 0 \\ g_3(x) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\ g_4(x) &= x_4 - 240 \le 0; \end{aligned}$$

range of variables:

$$0 \le x_1 \le 99, 0 \le x_2 \le 99, 10 \le x_3 \le 200, 10 \le x_4 \le 200$$

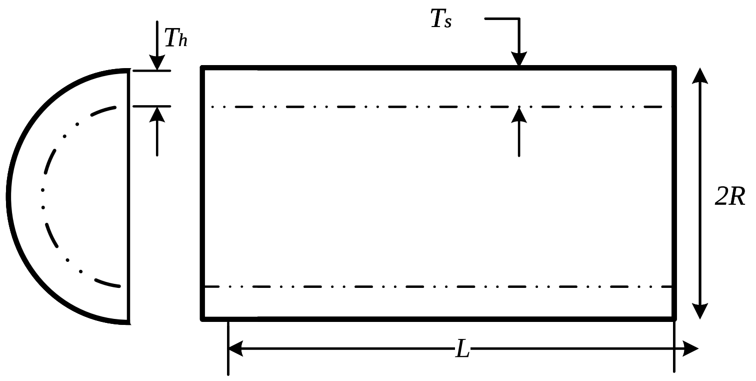


Figure 5. Pressure vessel design problem.

Table 13 shows how the DQOBLSMA compares with other competitor algorithms. The results shows the DQOBLSMA is able to find the optimal solution at the lowest cost, obtaining an optimal spend of 5885.623524 when *Ts*= 0.778246, *Th* = 0.384708, *R* = 40.323469, and *L* = 199.950065.

Table 13. Comparison in pressure vessel design.

Algorithm	**Optimal Values for Variables**				**Optimal Cost**
	Ts	***Th***	***R***	***L***	
DQOBLSMA	0.778246	0.384708	40.323469	199.950065	5885.623524
OBLSMAL	0.865273	0.427877	44.832637	145.769573	6060.212044
ESMA	0.974581	0.481740	50.496415	112.689545	6417.418230
SMA	0.814081	0.402437	42.180339	175.629283	5949.827184
MEO	0.850407	0.425437	44.051816	154.133369	6046.777664
JADE	0.788821	0.389961	40.870447	192.471633	5904.076066
RWGWO	0.877511	0.432390	45.308765	140.703767	6095.405916

6. Conclusions

In this paper, an enhanced SMA (DQOBLSMA) was proposed by introducing two mechanisms, DQRG and OBL, into the original SMA. In the DQOBLSMA, these two strategies further enhance the global search capability of the original SMA: DQRG enhances the exploration capability of the original SMA, and OBL increases the population diversity. The DQOBLSMA overcomes the weaknesses of the original search method and avoids premature convergence. The performance of the proposed DQOBLSMA was analyzed by using 23 classical mathematical benchmark functions.

First, the DQOBLSMA and the individual combinations of these two strategies were analyzed and discussed. The results showed that the proposed strategies are effective, and SMA achieved the best performance with the combination of the two mechanisms. Secondly, the results of the DQOBLSMA were compared with five state-of-the-art algorithms ESMA, AEO, JADE, OBLSMAL, and RWGWO. The results show that the DQOBLSMA is competitive with other advanced metaheuristic algorithms. To further validate the superiority of the DQOBLSMA, it was applied to three industrial engineering design problems. The experimental results show that the DQOBLSMA also achieves better results when solving engineering problems and significantly improves the original solutions.

As a future perspective, a multi-objective version of the DQOBLSMA will be considered. The proposed algorithm has promising applications in scheduling problems, image segmentation, parameter estimation, multi-objective engineering problems, text clustering, feature selection, test classification, and web applications.

Author Contributions: Conceptualization, S.D. and Y.Z.; software, Y.Z.; validation, S.D. and Q.Z.; formal analysis, S.D. and Y.Z.; investigation, S.D. and Y.Z.; resources, S.D.; writing—original draft preparation, Y.Z.; writing—review and editing, S.D. and Y.Z.; visualization, Y.Z.; funding acquisition, S.D. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the support of the Key R & D Projects of Zhejiang Province (No. 2022C01236, 2019C01060), the National Natural Science Foundations of China (Grant Nos. 21875271, U20B2021, 21707147, 51372046, 51479037, 91226202, and 91426304), the Entrepreneuship Program of Foshan National Hi-tech Industrial Development Zone, the Major Project of the Ministry of Science and Technology of China (Grant No. 2015ZX06004-001), Ningbo Natural Science Foundations (Grant Nos. 2014A610006, 2016A610273, and 2019A610106).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
2. Jamil, M.; Yang, X.S. A literature survey of benchmark functions for global optimization problems. *IJMMNO* **2013**, *4*, 150, https://doi.org/10.1504/IJMMNO.2013.055204.
3. Katebi, J.; Shoaei-parchin, M.; Shariati, M.; Trung, N.T.; Khorami, M. Developed comparative analysis of metaheuristic optimization algorithms for optimal active control of structures. *Eng. Comput.* **2020**, *36*, 1539–1558. https://doi.org/10.1007/s00366-019-00780-7.
4. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S. An improved grey wolf optimizer for solving engineering problems. *Expert Syst. Appl.* **2021**, *166*, 113917. https://doi.org/10.1016/j.eswa.2020.113917.
5. Abualigah, L.; Gandomi, A.H.; Elaziz, M.A.; Hamad, H.A.; Omari, M.; Alshinwan, M.; Khasawneh, A.M. Advances in meta-heuristic optimization algorithms in big data text clustering. *Electronics* **2021**, *10*, 101. https://doi.org/10.3390/electronics10020101.
6. Marinaki, M.; Marinakis, Y.; Stavroulakis, G.E. Fuzzy control optimized by PSO for vibration suppression of beams. *Control. Eng. Pract.* **2010**, *18*, 618–629. https://doi.org/10.1016/j.conengprac.2010.03.001.
7. David, R.C.; Precup, R.E.; Petriu, E.M.; Rădac, M.B.; Preitl, S. Gravitational search algorithm-based design of fuzzy control systems with a reduced parametric sensitivity. *Inf. Sci.* **2013**, *247*, 154–173.
8. Tang, A.D.; Han, T.; Zhou, H.; Xie, L. An improved equilibrium optimizer with application in unmanned aerial vehicle path planning. *Sensors* **2021**, *21*, 1814. https://doi.org/10.3390/s21051814.
9. Fu, J.; Lv, T.; Li, B. Underwater Submarine Path Planning Based on Artificial Potential Field Ant Colony Algorithm and Velocity Obstacle Method. *Sensors* **2022**, *22*, 3652. https://doi.org/10.3390/s22103652.
10. Alweshah, M.; Khalaileh, S.A.; Gupta, B.B.; Almomani, A.; Hammouri, A.I.; Al-Betar, M.A. The monarch butterfly optimization algorithm for solving feature selection problems. *Neural. Comput. Appl.* **2020**, *34*, 11267–11281. https://doi.org/10.1007/s00521-020-05210-0.
11. Alweshah, M. Solving feature selection problems by combining mutation and crossover operations with the monarch butterfly optimization algorithm. *Appl. Intell.* **2021**, *51*, 4058–4081. https://doi.org/10.1007/s10489-020-01981-0.
12. Almomani, O. A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms. *Symmetry* **2020**, *12*, 1046. https://doi.org/10.3390/sym12061046.
13. Moayedi, H.; Nguyen, H.; Kok Foong, L. Nonlinear evolutionary swarm intelligence of grasshopper optimization algorithm and gray wolf optimization for weight adjustment of neural network. *Eng. Comput.* **2021**, *37*, 1265–1275. https://doi.org/10.1007/s00366-019-00882-2.
14. Wunnava, A.; Naik, M.K.; Panda, R.; Jena, B.; Abraham, A. A novel interdependence based multilevel thresholding technique using adaptive equilibrium optimizer. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103836. https://doi.org/10.1016/j.engappai.2020.103836.
15. Kundu, R.; Chattopadhyay, S.; Cuevas, E.; Sarkar, R. AltWOA: Altruistic Whale Optimization Algorithm for feature selection on microarray datasets. *Comput. Biol. Med.* **2022**, *144*, 105349. https://doi.org/10.1016/j.compbiomed.2022.105349.
16. Abdel-Basset, M.; Mohamed, R.; Chakrabortty, R.K.; Sallam, K.; Ryan, M.J. An efficient teaching-learning-based optimization algorithm for parameters identification of photovoltaic models: Analysis and validations. *Energy Convers. Manag.* **2021**, *227*, 113614. https://doi.org/10.1016/j.enconman.2020.113614.

17. Abd Elaziz, M.; Yousri, D.; Al-qaness, M.A.A.; AbdelAty, A.M.; Radwan, A.G.; Ewees, A.A. A Grunwald–Letnikov based Manta ray foraging optimizer for global optimization and image segmentation. *Eng. Appl. Artif. Intell.* **2021**, *98*, 104105. https://doi.org/10.1016/j.engappai.2020.104105.
18. Naik, M.K.; Panda, R.; Abraham, A. An opposition equilibrium optimizer for context-sensitive entropy dependency based multi-level thresholding of remote sensing images. *Swarm Evol. Comput.* **2021**, *65*, 100907. https://doi.org/10.1016/j.swevo.2021.100907.
19. Yang, Y.; Tao, L.; Yang, H.; Iglauer, S.; Wang, X.; Askari, R.; Yao, J.; Zhang, K.; Zhang, L.; Sun, H. Stress sensitivity of fractured and vuggy carbonate: An X-Ray computed tomography analysis. *J. Geophys. Res. Solid Earth* **2020**, *125*, e2019JB018759.
20. Lin, S.W.; Cheng, C.Y.; Pourhejazy, P.; Ying, K.C. Multi-temperature simulated annealing for optimizing mixed-blocking permutation flowshop scheduling problems. *Expert Syst. Appl.* **2021**, *165*, 113837. https://doi.org/10.1016/j.eswa.2020.113837.
21. Hernández-Ramírez, L.; Frausto-Solís, J.; Castilla-Valdez, G.; González-Barbosa, J.; Sánchez Hernández, J.P. Three Hybrid Scatter Search Algorithms for Multi-Objective Job Shop Scheduling Problem. *Axioms* **2022**, *11*, 61. https://doi.org/10.3390/axioms11020061.
22. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992.
23. Rocca, P.; Oliveri, G.; Massa, A. Differential evolution as applied to electromagnetics. *IEEE Antennas Propag. Mag.* **2011**, *53*, 38–49. https://doi.org/10.1109/MAP.2011.5773566.
24. Juste, K.; Kita, H.; Tanaka, E.; Hasegawa, J. An evolutionary programming solution to the unit commitment problem. *IEEE Trans. Power Syst.* **1999**, *14*, 1452–1459. https://doi.org/10.1109/59.801925.
25. Beyer, H.G.; Schwefel, H.P. Evolution strategies–a comprehensive introduction. *Nat. Comput.* **2002**, *1*, 3–52. https://doi.org/10.1023/A:1015059928466.
26. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680.
27. Abedinpourshotorban, H.; Mariyam Shamsuddin, S.; Beheshti, Z.; Jawawi, D.N.A. Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. *Swarm Evol. Comput.* **2016**, *26*, 8–22. https://doi.org/10.1016/j.swevo.2015.07.002.
28. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl. Based Syst.* **2020**, *191*, 105190. https://doi.org/10.1016/j.knosys.2019.105190.
29. Hashim, F.A.; Hussain, K.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Appl. Intell.* **2021**, *51*, 1531–1551. https://doi.org/10.1007/s10489-020-01893-z.
30. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008.
31. 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. https://doi.org/10.1016/j.advengsoft.2017.07.002.
32. Wang, G.G. Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. https://doi.org/10.1007/s12293-016-0212-3.
33. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; 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. https://doi.org/10.1016/j.cie.2021.107250.
34. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.
35. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. https://doi.org/10.1016/j.future.2019.02.028.
36. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
37. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput* **1997**, *1*, 67–82.
38. 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. https://doi.org/10.1016/j.future.2020.03.055.
39. Zhao, S.; Wang, P.; Heidari, A.A.; Chen, H.; Turabieh, H.; Mafarja, M.; Li, C. Multilevel threshold image segmentation with diffusion association slime mould algorithm and Renyi's entropy for chronic obstructive pulmonary disease. *Comput. Biol. Med.* **2021**, *134*, 104427, https://doi.org/10.1016/j.compbiomed.2021.104427.
40. Zubaidi, S.L.; Abdulkareem, I.H.; Hashim, K.S.; Al-Bugharbee, H.; Ridha, H.M.; Gharghan, S.K.; Al-Qaim, F.F.; Muradov, M.; Kot, P.; Al-Khaddar, R. Hybridised artificial neural network model with slime mould algorithm: A novel methodology for prediction of urban stochastic water demand. *Water* **2020**, *12*, 2692. https://doi.org/10.3390/w12102692.
41. Wang, H.J.; Pan, J.S.; Nguyen, T.T.; Weng, S. Distribution network reconfiguration with distributed generation based on parallel slime mould algorithm. *Energy* **2022**, *244*, 123011. https://doi.org/10.1016/j.energy.2021.123011.
42. Tang, A.D.; Tang, S.Q.; Han, T.; Zhou, H.; Xie, L. A modified slime mould algorithm for global optimization. *Comput. Intell. Neurosci.* **2021**, *2021*, 2298215, https://doi.org/10.1155/2021/2298215.
43. Örnek, B.N.; Aydemir, S.B.; Düzenli, T.; Özak, B. A novel version of slime mould algorithm for global optimization and real world engineering problems: Enhanced slime mould algorithm. *Math. Comput. Simul.* **2022**, *198*, 253–288. https://doi.org/10.1016/j.matcom.2022.02.030.

44. Kaveh, A.; Biabani Hamedani, K.; Kamalinejad, M. Improved slime mould algorithm with elitist strategy and its application to structural optimization with natural frequency constraints. *Comput. Struct.* **2022**, *264*, 106760. https://doi.org/10.1016/j.compstruc.2022.106760.
45. Pfaff, W.; Hensen, B.J.; Bernien, H.; van Dam, S.B.; Blok, M.S.; Taminiau, T.H.; Tiggelman, M.J.; Schouten, R.N.; Markham, M.; Twitchen, D.J.; et al. Unconditional quantum teleportation between distant solid-state quantum bits. *Science* **2014**, *345*, 532–535. https://doi.org/10.1126/science.1253512.
46. Xu, B.; Heidari, A.A.; Kuang, F.; Zhang, S.; Chen, H.; Cai, Z. Quantum Nelder-Mead Hunger Games Search for optimizing photovoltaic solar cells. *Int. J. Energy Res.* **2022**, *46*, 12417–12466. https://doi.org/10.1002/er.8011.
47. Tizhoosh, H. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005; Volume 1, pp. 695–701. https://doi.org/10.1109/CIMCA.2005.1631345.
48. Abualigah, L.; Diabat, A.; Elaziz, M.A. Improved slime mould algorithm by opposition-based learning and Levy flight distribution for global optimization and advances in real-world engineering problems. *J. Ambient. Intell. Humaniz. Comput.* **2021**, pp. 1-40 1–40. https://doi.org/10.1007/s12652-021-03372-w.
49. Naik, M.K.; Panda, R.; Abraham, A. An entropy minimization based multilevel colour thresholding technique for analysis of breast thermograms using equilibrium slime mould algorithm. *Appl. Soft Comput.* **2021**, *113*, 107955.
50. Gupta, S.; Deep, K.; Mirjalili, S. An efficient equilibrium optimizer with mutation strategy for numerical optimization. *Appl. Soft Comput.* **2020**, *96*, 106542.
51. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958.
52. Gupta, S.; Deep, K. A novel random walk grey wolf optimizer. *Swarm Evol. Comput.* **2019**, *44*, 101–112.
53. García, S.; Fernández, 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. *Inf. Sci.* **2010**, *180*, 2044–2064.
54. Wang, Z.; Luo, Q.; Zhou, Y. Hybrid metaheuristic algorithm using butterfly and flower pollination base on mutualism mechanism for global optimization problems. *Eng. Comput.* **2021**, *37*, 3665–3698.
55. Chen, H.; Heidari, A.A.; Zhao, X.; Zhang, L.; Chen, H. Advanced orthogonal learning-driven multi-swarm sine cosine optimization: Framework and case studies. *Expert Syst. Appl.* **2020**, *144*, 113113. https://doi.org/10.1016/j.eswa.2019.113113.
56. 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. https://doi.org/10.3390/pr9091551.
57. 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.