applied sciences

MDPI

Article

A Hybrid Golden Jackal Optimization and Golden Sine Algorithm with Dynamic Lens-Imaging Learning for Global Optimization Problems

Panliang Yuan [1], Taihua Zhang [2,*], Liguo Yao [2], Yao Lu [2] and Weibin Zhuang [1]

[1] School of Mechanical and Electrical Engineering, Guizhou Normal University, Guiyang 550025, China
[2] Technical Engineering Center of Manufacturing Service and Knowledge Engineering, Guizhou Normal University, Guiyang 550025, China
* Correspondence: zhangth542@163.com; Tel.: +86-135-1851-5198

Abstract: Golden jackal optimization (GJO) is an effective metaheuristic algorithm that imitates the cooperative hunting behavior of the golden jackal. However, since the update of the prey's position often depends on the male golden jackal and there is insufficient diversity of golden jackals in some cases, it is prone to falling into a local optimal optimum. In order to address these drawbacks of GJO, this paper proposes an improved algorithm, called a hybrid GJO and golden sine (S) algorithm (Gold-SA) with dynamic lens-imaging (L) learning (LSGJO). First, this paper proposes novel dual golden spiral update rules inspired by Gold-SA. These rules give GJO the ability to think like a human (Gold-SA), making the golden jackal more intelligent in the process of preying, and improving the ability and efficiency of optimization. Second, a novel nonlinear dynamic decreasing scaling factor is introduced into the lens-imaging learning operator to maintain the population diversity. The performance of LSGJO is verified through 23 classical benchmark functions and 3 complex design problems in real scenarios. The experimental results show that LSGJO converges faster and more accurately than 11 state-of-the-art optimization algorithms, the global and local search ability has improved significantly, and the proposed algorithm has shown superior performance in solving constrained problems.

Keywords: hybrid metaheuristics; golden jackal algorithm; lens-imaging learning; golden sine algorithm; global optimization problems

check for updates

Citation: Yuan, P.; Zhang, T.; Yao, L.; Lu, Y.; Zhuang, W. A Hybrid Golden Jackal Optimization and Golden Sine Algorithm with Dynamic Lens-Imaging Learning for Global Optimization Problems. *Appl. Sci.* **2022**, *12*, 9709. https://doi.org/10.3390/app12199709

Academic Editors: Antonio J. Nebro and José Manuel García-Nieto

Received: 1 September 2022
Accepted: 23 September 2022
Published: 27 September 2022

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

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Natural science and social economy optimization problems are a research hotspot in computer science, management and decision-making, artificial intelligence, and other fields. The search for high-precision solutions to such optimization problems has attracted many researchers. However, the traditional optimization methods based on mathematical theory, such as Newton's downhill method and the gradient descent method, have been unable to solve these problems effectively [1,2], so many scholars favor metaheuristic algorithms.

Metaheuristic algorithms are used to find the optimal solution or satisfactory solution to complex optimization problems [3–5], and they are inspired by the phenomenon of a biological population, physical phenomena, evolutionary law, etc. For example, the whale optimization algorithm (WOA) is inspired by the humpback whales' foraging behavior in nature [6]. The salp swarm algorithm (SSA) is inspired by the swarming behavior of salps when navigating and foraging in oceans [7]. The Harris's hawk optimization (HHO) is inspired by the different mechanisms of the Harris's hawk's strategy for capturing prey [8]. Biological populations inspire these algorithms. For example, the equilibrium optimizer (EO) is inspired by control volume mass balance models that estimate both dynamic and equilibrium states [9]. The lightning attachment procedure optimization (LAPO) is inspired by the natural process of connecting the upward-facing and downward-facing leads of

lightning [10]. The turbulent flow of water-based optimization (TFWO) is inspired by whirlpools created in the turbulent flow of water [11]. Physical phenomena inspire these algorithms. For example, the inspiration for the genetic algorithm (GA) comes from the survival of the fittest under the action of genetics, selection, and variation in organisms to achieve evolution and development [12]. The immune algorithm (IA) inspiration comes from the immune mechanism of biology, combined with the evolutionary mechanism of genes [13]. The laws of evolution inspire these algorithms. Metaheuristic algorithms are widely used in signal processing [14], image processing [15,16], fault detection [17], production scheduling [18], feature selection [19], path planning [20], numerical optimization [21], engineering design [22–24], etc.

Moreover, the no-free-lunch theory (NFL) shows that no one algorithm can be applied to all optimization problems [25]. This theory has prompted many researchers to improve existing algorithms. Alkayem et al. proposed the self-adaptive quasi-oppositional stochastic fractal search (SA-QSFS) by employing triple modal-based objective function combination and quasi-oppositional learning [26]. Tian and Shi proposed MPSO using chaos initialization and sigmoid-like inertia weight based on the PSO algorithm [27]. Dhargupta et al. proposed SOGWO by combining opposition-based learning (OBL) with a grey wolf optimizer [28]. Alkayem et al. proposed the social engineering particle swarm optimization algorithm (SEPSO) by combining the PSO population-based elitist-solution mechanism and the SEO two-solution attacker–defender paradigm [29].

Our team has also improved some algorithms and achieved good results in numerical optimization and engineering applications: Wei et al. proposed a new unbalanced fault diagnosis framework using MFO to optimize γ parameters in LS-SVM [30]. Fan et al. proposed BGWO by combining the beetle antenna strategy with the gray wolf algorithm and adopting the nonlinear control step size strategy [31]. Fan et al. proposed m-EO based on EO with reverse learning, new update rules, and chaos strategy [32]. Fan et al. proposed MMPA based on WOA with a new position update strategy, a logistic opposition-based learning mechanism, inertia weight coefficient, and a nonlinear control step size strategy [33]. Wei et al. proposed NI-MWMOTE based on MWMOTE with an adaptive noise processing scheme and an aggregative hierarchical clustering method [34].

The golden jackal optimization algorithm (GJO) is a recently proposed biological swarm intelligence algorithm [35], inspired by the collaborative hunting behavior of golden jackals. Although GJO has the advantages of easy implementation, high stability, and few adjustment parameters, its exploration and exploitation capabilities are unbalanced, which can easily lead to excessive exploitation and fall into local optima. In the process of iteration, the position of prey is always located in the middle of two golden jackals. However, the position of male golden jackals is not necessarily the optimal solution, which easily leads to slow convergence in the late iteration, poor convergence accuracy, and easily falling into a locally optimal solution. Aiming at the shortcomings of GJO, this paper improves the original algorithm and proposes LSGJO. The new algorithm adds the dynamic lens-imaging learning strategy and the novel position updating strategy to make it have better global search ability and local search ability.

In order to prove its superior result, LSGJO is compared with several well-known and recent algorithms on 23 benchmark functions. Among the 23 benchmark functions, the functions F1-F13 have three different dimensions (30, 100, 500), and F14-F23 are fixed-dimension functions. The experimental results show that the algorithm proposed in this paper has a fast convergence speed and high accuracy. In addition, the experimental results of LSGJO on constrained optimization problems in three mechanical fields also show that the proposed algorithm can solve practical problems.

The highlights and contributions of this paper are summarized as follows:

(1) LSGJO is proposed.
(2) Wilcoxon rank sum test and Friedman test are used to analyze the statistical data. Observing the convergence curve and comparing it with other algorithms proves that LSGJO has tremendous advantages.

(3) LSGJO is applied to solve three constrained optimization problems in mechanical fields and compared with many advanced algorithms.

The remaining sections of this paper are as follows: Section 2 briefly summarizes the conventional golden jackal algorithm. Section 3 proposes LSGJO and analyzes its time complexity. The benchmark functions are tested, and the results are analyzed in Section 4. LSGJO is used to solve three constrained optimization problems in mechanical fields in Section 5. Section 6 discusses the challenges, recommendations, and limitations related to the proposed algorithm. Finally, Section 7 concludes the paper and proposes future studies.

2. Golden Jackal Algorithm

The golden jackal algorithm is a swarm intelligence algorithm proposed by Nitish Chopra and Muhammad Mohsin Ansari; it mimics the hunting behavior of golden jackals in nature. Golden jackals usually hunt with males and females. The hunting behavior of the golden jackal is divided into three steps: (1) searching for and moving towards the prey; (2) enclosing and irritating the prey until it stops moving; and (3) pouncing towards the prey.

During the initialization phase, a randomly distributed set of prey position matrices is generated by Equation (1):

$$\begin{bmatrix} Y_{1,1} & \cdots & Y_{1,j} & \cdots & Y_{1,n} \\ Y_{2,1} & \cdots & Y_{2,j} & \cdots & Y_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{N-1,1} & \cdots & Y_{N-1,j} & \cdots & Y_{N-1,n} \\ Y_{N,1} & \cdots & Y_{N,j} & \cdots & Y_{N,n} \end{bmatrix} \tag{1}$$

where N denotes the number of prey populations and n denotes dimensions.

The mathematical model of the golden jackal's hunt is as follows ($|E| > 1$):

$$Y_1(t) = Y_M(t) - E \cdot |Y_M(t) - rl \cdot Prey(t)| \tag{2}$$

$$Y_2(t) = Y_{FM}(t) - E \cdot |Y_{FM}(t) - rl \cdot Prey(t)| \tag{3}$$

where t is the current iteration, $Y_M(t)$ indicates the position of the male golden jackal, $Y_{FM}(t)$ indicates the position of the female, and $Prey(t)$ is the position vector of the prey. $Y_1(t)$ and $Y_2(t)$ are the updated positions of the male and female golden jackals.

E is the evading energy of prey and is calculated as follows:

$$E = E_1 \cdot E_0 \tag{4}$$

$$E_1 = c_1 \cdot (1–(t/T)) \tag{5}$$

where E_0 is a random number in the range [–1, 1], indicating the prey's initial energy; T represents the maximum number of iterations; c_1 is the default constant set to 1.5; and E_1 denotes the prey's decreasing energy.

In Equations (2) and (3), $|Y_M(t) - rl \cdot Prey(t)|$ denotes the distance between the golden jackal and prey and "rl" is the vector of random numbers calculated by the Levy flight function.

$$rl = 0.05 \cdot LF(y) \tag{6}$$

$$LF(y) = 0.01 \times (\mu \times \sigma) / \left(\left| v^{(1/\beta)} \right| \right) \qquad \sigma = \left\{ \frac{\Gamma(1+\beta) \times \sin(\pi\beta/2)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times \left(2^{\beta-1}\right)} \right\}^{1/\beta} \tag{7}$$

where u and v are random values in (0, 1) and β is the default constant set to 1.5.

$$Y(t+1) = \frac{Y_1(t) + Y_2(t)}{2} \tag{8}$$

where $Y(t+1)$ is the updated position of the prey based on the male and the female golden jackals.

When the prey is harassed by the golden jackals, the evading energy is decreased. The mathematical model of the golden jackals surrounding prey and devouring it is as follows ($|E| \leq 1$):

$$Y_1(t) = Y_M(t) - E \cdot |rl \cdot Y_M(t) - Prey(t)| \tag{9}$$

$$Y_2(t) = Y_{FM}(t) - E \cdot |rl \cdot Y_{FM}(t) - rl \cdot Prey(t)| \tag{10}$$

The pseudo-code of the above GJO is shown in Algorithm 1.

Algorithm 1: Golden Jackal Optimization

Inputs: The population size N and maximum number of iterations T
Outputs: The location of prey and its fitness value
Initialize the random prey population Y_i (i = 1, 2, ... , N)
While ($t < T$)
  Calculate the fitness values of prey
  Y_1 = best prey individual (Male Jackal Position)
  Y_2 = second best prey individual (Female Jackal Position)
  for (each prey individual)
    Update the evading energy "E" using Equations (4) and (6)
    Update "rl" using Equations (6) and (7)
    If ($|E| \leq 1$) (Exploration phase)
    Update the prey position using Equations (2), (3), and (8)
    If ($|E| > 1$) (Exploitation phase)
    Update the prey position using Equations (8), (9), and (10)
  end for
  $t = t + 1$
end while
return Y_1

3. Proposed LSGJO

When solving some optimization problems, GJO easily falls into iterative stagnation, slow convergence in later stages, and insufficient exploration and exploitation capacity, and the shortcomings are more apparent when solving complex problems. In this section, we propose two improvement strategies described in detail below.

3.1. Dynamic Lens-Imaging Learning Strategy

Lens-imaging learning strategy is a recently proposed opposition-based learning method [36]. This strategy is derived from the law of optics in the convex lens-imaging law. The principle of the strategy is to refract the entity on one side to the other through a convex lens to form an inverted image. Here, Figure 1 is used to outline its principle: on the left of the coordinate axis y, there is an individual G (the male golden jackal); its projection on the coordinate axis x is X, and its distance from the coordinate axis x is h. The coordinate axis y denotes a convex lens of focal length f, and the O point is the center of the convex lens. G passes through a convex lens to produce an opposite individual G', whose projection on the coordinate axis x is X', and its distance from the coordinate axis x is h'. The individual X and its opposite individual X' are obtained.

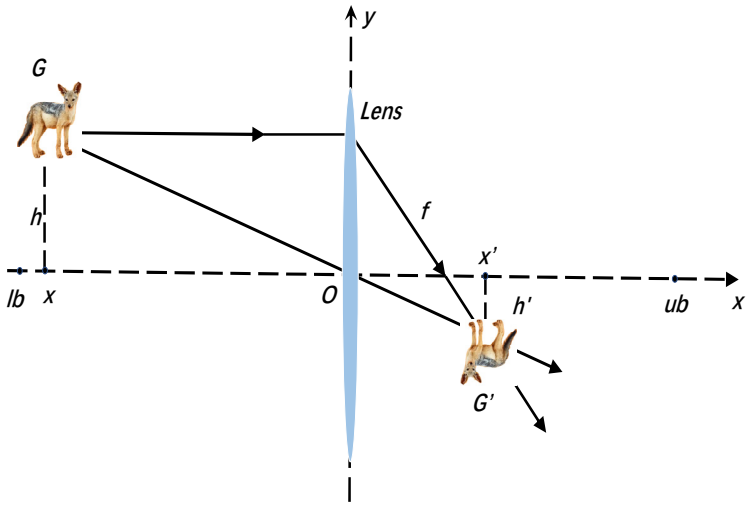

Figure 1. The principle of lens-imaging learning strategy.

According to Figure 1, X and X' can be derived from the convex lens-imaging principle:

$$\frac{\frac{ub+lb}{2} - X}{X' - \frac{ub+lb}{2}} = \frac{h}{h'} \tag{11}$$

where ub and lb are the upper and lower bounds. Let $h/h' = \alpha$, and α is called the scaling factor; then, Equation (11) is transformed to obtain the formula for the opposite point X':

$$X' = \frac{ub + lb}{2} + \frac{ub + lb}{2 \cdot \alpha} - \frac{X}{\alpha} \tag{12}$$

The scaling factor α can increase the local development ability of the LSGJO. In the original lens-imaging learning strategy, the scaling factors are generally considered constant, which reduces the convergence performance of the algorithm. Therefore, this paper proposes a new scaling factor based on nonlinear dynamic decreasing, which can obtain larger values in the early iteration of the algorithm so that the algorithm can search in the broader range of different dimensional regions and improve the diversity of the population. At the end of the algorithm iteration, a smaller value is obtained, so the fine search near the optimal individual can be carried out to improve the local optimization ability. The nonlinear dynamic scaling factor α is calculated by Equation (13):

$$\alpha = \zeta_{min} - (\zeta_{max} - \zeta_{min}) * (t/T)^2 \tag{13}$$

where ζ_{max} is the maximum scaling factor, ζ_{min} is the minimum scaling factor, and T is the maximum number of iterations; the value ζ_{max} is 100, and the value ζ_{min} is 10.

Equation (12) can be popularized to the n-dimensional search space:

$$X'_j = \frac{ub_j + lb_j}{2} + \frac{ub_j + lb_j}{2 \cdot \alpha} - \frac{X_j}{\alpha} \tag{14}$$

where X'_j and X_j are the components of X' and X in dimension j, respectively, and lb_j and ub_j are the upper and lower bounds of dimension j, respectively. The dynamic lens-imaging strategy considers the candidate and opposite solutions and selects the best solution according to the calculated fitness. In this paper, the dynamic lens-imaging learning strategy is applied to the current global optima of the swarm in the GJO and is beneficial to help the population avoid stagnation in local optima.

Learning strategies mainly include the opposition-based learning (OBL) strategy, the quasi-opposition-based learning strategy, and the dynamic lens-imaging-based learning

strategy. The original OBL is a special form of $\alpha = 1$ in Equation (12). Quasi-oppositional learning, proposed by Tizhoosh et al. [37], is utilized to improve the overall exploration of the initial and execution stages, and it is an excellent improvement on the original opposition-based learning method. It can increase the diversity of the population but ignores that with the increase in the number of iterations, the algorithm changes from global optimization to local optimization. The dynamic lens-imaging learning proposed in this paper takes this into account.

3.2. Novel Update Rules

The golden sine algorithm was proposed by Tanyildizi et al. [2]; it is inspired by the relationship between the sine function and the unit circle in mathematics. The golden section coefficient is introduced in the position update in the golden sine algorithm, and the special relationship between the sine function and the unit element is combined with the golden section. The golden sine algorithm finds the global optimal solution by reducing the search scope continuously. First, the global search finds the optimal solution space, then the local search is carried out, and finally the global optimal solution is sought. The golden sine algorithm has better local search ability, and its mathematical model is as follows:

$$X_i^{t+1} = X_i^t \cdot |sin(R_1)| + R_2 \cdot sin(R_1) \cdot \left| x_1 \cdot P_i^t - x_2 \cdot X_i^t \right| \tag{15}$$

where t denotes the current iteration number; R_1 is a random value inside $[0, 2\pi]$; R_2 is a random value inside $[0, \pi]$; R_1 and R_2 indicate the direction and distance of the next generation of movement, respectively; and x_1 and x_2 are the golden section coefficients, which are used to narrow the search space and guide the individual to converge to the optimal solution.

$$x_1 = a \cdot (1 - \tau) + b \cdot \tau \tag{16}$$

$$x_2 = a \cdot \tau + b \cdot (1 - \tau) \tag{17}$$

$$\tau = \left(\sqrt{5} - 1\right)/2 \tag{18}$$

where a and b are the initial values $-\pi$ and π, and τ represents the golden number.

When the golden sine algorithm and the golden jackal algorithm are combined, the position update rules of male and female jackals in the exploitation stage are as follows:

$$Y_1(t) = prey(t) \cdot |sin(R_1)| + R_2 \cdot sin(R_1) \cdot |x_1 \cdot Y_M(t) - x_2 \cdot prey(t)| \tag{19}$$

$$Y_2(t) = prey(t) \cdot |sin(R_1)| + R_2 \cdot sin(R_1) \cdot |x_1 \cdot Y_{FM}(t) - x_2 \cdot prey(t)| \tag{20}$$

The position updating rule adopts the dual golden spiral update rules, mimics the golden jackal surrounding the prey in a curve way, consumes the prey's physical strength, gradually narrows the encircling circle of the prey, and then captures the prey. This position updating rule is more in line with the natural golden jackal surrounding and capturing prey state, and the principle of the dual golden spiral update rules is shown in Figure 2.

In a word, combining the lens-imaging strategy with a nonlinear dynamic decreasing factor and the new update rules enables GJO to jump out of the local optimum, accelerate the convergence, and improve the convergence accuracy. In the exploitation phase of the golden jackal algorithm, adding a levy flight function can avoid falling into local optimization to a certain extent. Since the levy flight function is characterized by short-distance and occasional long-distance jumps, GJO still falls into local optima in some numerical optimizations. Especially in the high-dimensional function, its effect will be significantly reduced. In this regard, the dynamic lens-imaging learning strategy is used to find the opposite of the current global optimal solution, increase the population's diversity, and retain a better one by comparing the fitness function values. In the exploitation phase, the positions of male and female jackals are updated by the new update rules. The pseudo-code of LSGJO is shown in Algorithm 2 and Figure 3.

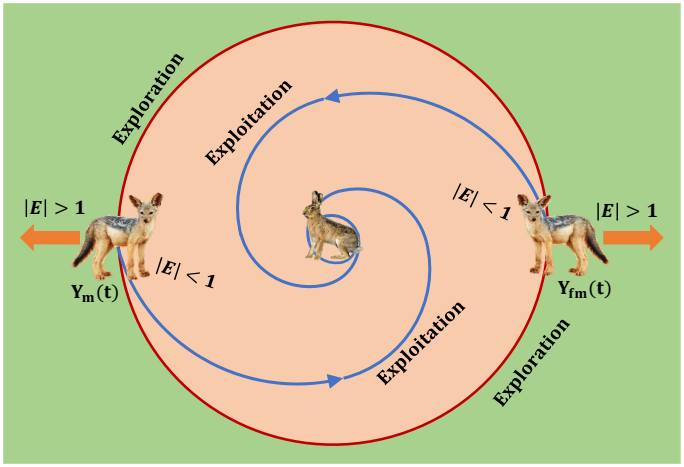

Figure 2. The principle of the dual golden spiral update rules.

Algorithm 2: The pseudo-code of LSGJO

Inputs: The population size N and maximum number of iterations T
Outputs: The location of prey and its fitness value
Initialize the random prey population Y_i (i = 1, 2, . . . , N)
While ($t < T$)
Calculate the fitness values of prey
Y_1 = best prey individual (Male Jackal Position)
Y_2 = second best prey individual (Female Jackal Position)
Obtain Y_1^* by Equation (14)
Calculate the fitness function values of Y_1 and Y_1^*, set the better one as Y_1
for (each prey individual)
Update the evading energy "E" using Equations (3) and (4)
Update "rl" using Equations (6) and (7)
If($|E| \geq 1$) (Exploration phase)
Update the prey position using Equations (8), (19), and (20)
If($|E| < 1$) (Exploitation phase)
Update the prey position using Equations (8), (9), and (10)
end for
$t = t + 1$
end while
return Y_1

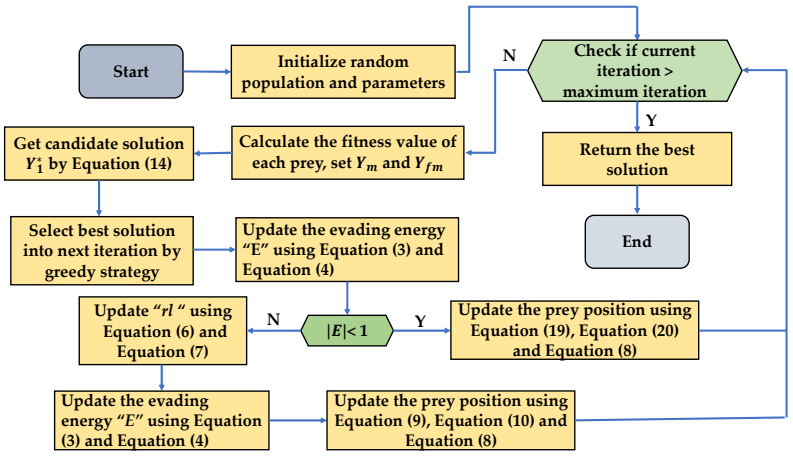

Figure 3. Flowchart of LSGJO.

3.3. The Computational Complexity of LSGJO

The time complexity indirectly reflects the convergence rate of the algorithm. Suppose the time required to initialize the parameters (population size N, dimension d, coefficient E, rl, etc.) is γ_1. According to Equation (7), the time needed for each dimension to update the position of the prey and the position of the golden jackal is γ_2, and the time for solving the fitness value of the objective function is $f(n)$; then, the time complexity of GJO is:

$$T_1(n) = O(\gamma_1 + N(d \times \gamma_2 + f(n))) = O(d + f(n)) \tag{21}$$

In the LSGJO algorithm, it is assumed that the initialization parameters (population size N, dimension d, τ, x_1, x_2 coefficient E, rl, etc.) are γ_3, and the time required to perform the lens-imaging learning strategy is γ_4. The time required to execute the greedy mechanism is γ_5. According to Equation (7), the time needed for each dimension to update the position of the prey and the position of the golden jackal is γ_6; then, the time complexity of the LSGJO is:

$$T_2(n) = O(\gamma_3 + \gamma_5 + N(d \times \gamma_6 + \gamma_4 + f(n))) = O(d + f(n)) \tag{22}$$

The LSGJO proposed in this paper has the same time complexity as GJO:

$$T_1(n) = T_2(n) \tag{23}$$

In summary, the LSGJO does not increase the time complexity.

4. Simulation and Result Analysis

To verify the performance of the LSGJO, this study uses 23 benchmark functions commonly used in the literature [2,35], which are listed in Table 1. The functions F1~F7 are high-dimensional unimodal functions and have a single global optimal solution. These functions are used to test the convergence rate of search algorithms, The functions F8~F13 are high-dimensional multimodal functions and have a single global optimum and multiple locally optimal solutions; these functions are designed to test the search capacities of optimization algorithms. The functions F14~F23 are low-dimensional multimodal functions and have a small number of local minima. The range indicates the solution space, and F_{min} denotes the optimal value. In order to verify the robustness of LSGJO, the 13 functions F1~F13 were tested with 100 and 500 dimensions.

Table 1. The benchmark functions.

Function	Dim	Range	F_{min}	Type
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30, 100, 500	$[-100, 100]$	0	Unimodal
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30, 100, 500	$[-1.28, 1.28]$	0	Unimodal
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30, 100, 500	$[-100, 100]$	0	Unimodal
$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \leqslant i \leqslant n \}$	30, 100, 500	$[-100, 100]$	0	Unimodal
$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30, 100, 500	$[-30, 30]$	0	Unimodal
$f_6(x) = \sum_{i=1}^{n} [x_i + 0.5]^2$	30, 100, 500	$[-100, 100]$	0	Unimodal
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1]$	30, 100, 500	$[-1.28, 1.28]$	0	Unimodal
$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30, 100, 500	$[-500, 500]$	$-418.9829 \times$ n	Multimodal
$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	30, 100, 500	$[-5.12, 5.12]$	0	Multimodal

Table 1. *Cont.*

Function	Dim	Range	F_{min}	Type
$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30, 100, 500	$[-32, 32]$	0	Multimodal
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30, 100, 500	$[-600, 600]$	0	Multimodal
$f_{12}(x) = \frac{\pi}{n}\left\{ 10\sin(\pi \mathrm{yi}) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2 \right\} + \sum_{i=1}^{n} u(xi, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(xi, a, k, m) = \begin{cases} k(x_i-a)^m, x_i > a \\ 0, -a < x_i < a \\ k(-x_i-a)^m, x_i < -a \end{cases}$	30, 100, 500	$[-50, 50]$	0	Multimodal
$f_{13}(x) = 0.1\left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right] + (x_n-1)^2\left[1+\sin^2(2\pi x_n)\right] \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30, 100, 500	$[-50, 50]$	0	Multimodal
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	$[-65.536, 65.536]$	1	Multimodal
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	$[-5, 5]$	0.0003	Multimodal
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.0316	Multimodal
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5, 5]$	0.398	Multimodal
$f_{18}(x) = \left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right] \times \left[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	$[-2, 2]$	3	Multimodal
$f_{19}(x) = -\sum_{i=1}^{4} c_i\exp\left(-\sum_{j=1}^{3} a_{ij}(x_j-p_{ij})^2\right)$	3	$[0, 1]$	-3.86	Multimodal
$f_{20}(x) = -\sum_{i=1}^{4} c_i\exp\left(-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2\right)$	6	$[0, 1]$	-3.32	Multimodal
$f_{21}(x) = -\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.1532	Multimodal
$f_{22}(x) = -\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.4029	Multimodal
$f_{23}(x) = -\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.5364	Multimodal

All experiments were conducted on the same environment configuration, and all algorithms were implemented in Matlab 2016 b installed on Windows 10 (64 bit), CPU Intel(R) i5-9400 F at 2.9 GHz and 16 GB of RAM.

In this paper, some novel swarm intelligence algorithms, including grey wolf optimizer (GWO) [38], Harris's hawk optimization (HHO), chimp optimization algorithm (ChoA) [39], golden jackal optimization (GJO), equilibrium optimizer (EO), whale optimization algorithm (WOA), salp swarm algorithm (SSA), snake optimizer (SO) [40], particle swarm optimization (PSO) [41], modified particle swarm optimization (MPSO), and selective opposition-based grey wolf optimization (SOGWO), are compared with the improved algorithms.

The parameters of all the comparison algorithms are shown in Table 2. In order to ensure the fairness of the experimental results, the population size of each algorithm was set to 30, and the maximum number of iterations was set to 500. Each algorithm ran 30 times independently, and its average and standard deviation were recorded.

Table 2. Parameter settings of various algorithms.

Algorithm	Parameter Settings
GWO	a = 2(linearly decreased over iterations)
HHO	J = [0, 2]
ChoA	a = 2 (linearly decreased over iterations), m = chaos (3, 1, 1)
GJO	a = 1.5 (linearly decreased over iterations)
EO	a1 = 2, a2 = 1, GP = 0.5, t = 1 (nonlinearly decreased over iterations)
WOA	b = 1
SSA	c1 = 2 (nonlinearly decreased over iterations)
SO	a = 2 (linearly decreased over iterations)
PSO	W = 0.9, c1 = 2, c2 = 2
MPSO	W_{max} = 0.9, W_{min} = 0.4, c1 = 2, c2 = 2
SOGWO	a = 2 (linearly decreased over iterations)
LSGJO	A = 1.5 (linearly decreased over iterations)

4.1. Comparison and Analysis with Metaheuristic Algorithms

The experimental results of 11 algorithms on 23 benchmark functions are shown in Table 3. As can be seen from the mean and standard deviation, LSGJO performs better than GJO in almost every function. Compared with other algorithms, LSGJO is the first in all the test functions except the average ranking of F6, F12, F14, and F20. In all benchmark function tests, the average value and standard deviation of LSGJO test results are small, indicating that the performance of the LSGJO is the best. From the convergence curve in Figure 4, it can be seen that LSGJO converges to the optimal solution much faster than other algorithms.

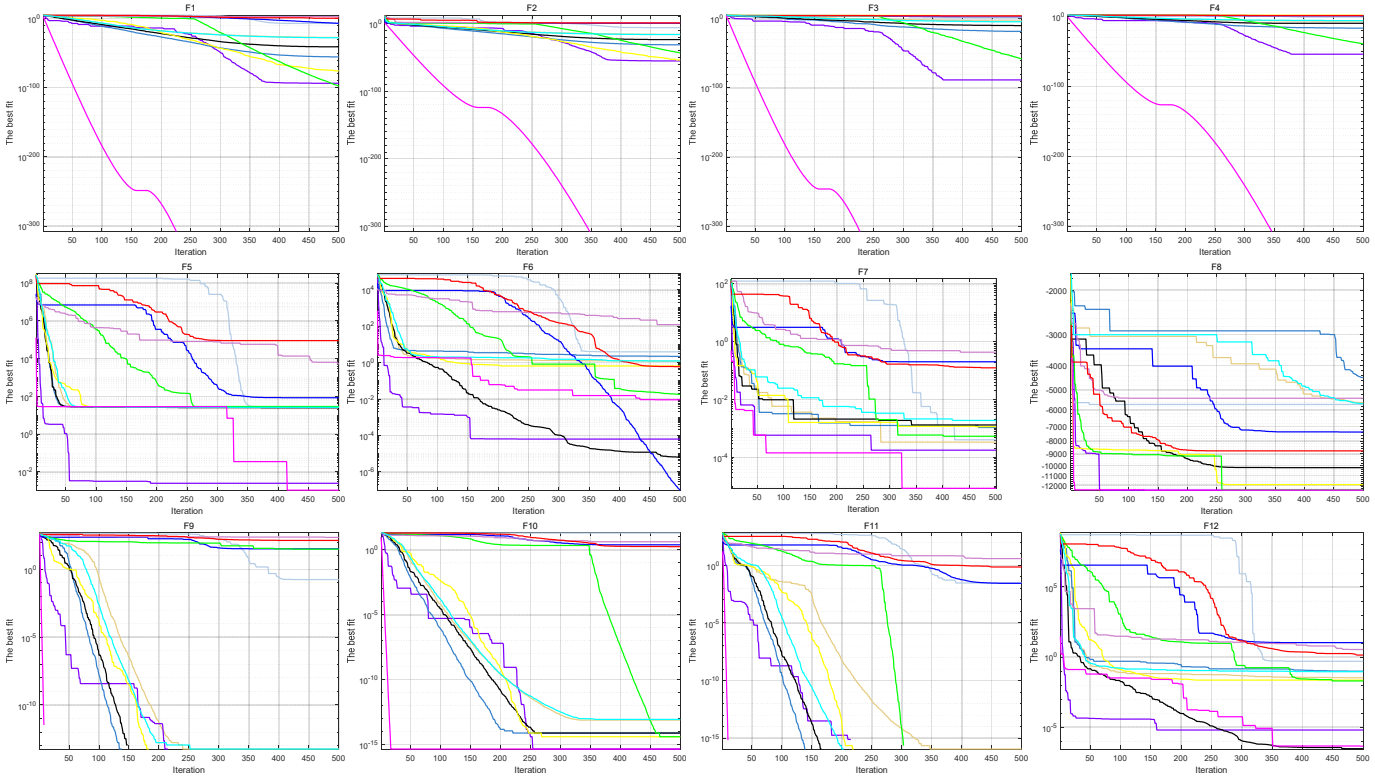

Figure 4. *Cont.*

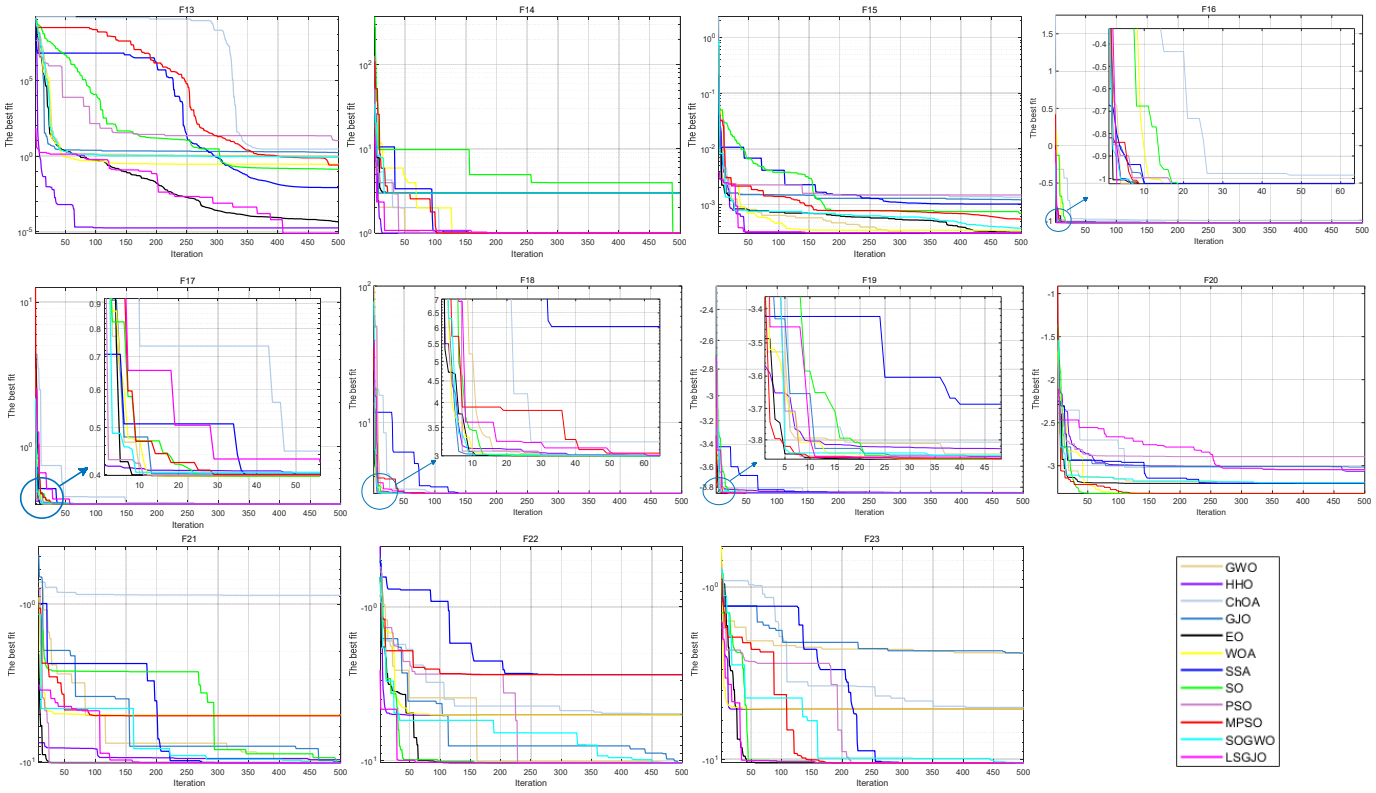

Figure 4. The convergence curves of the LSGJO and other comparison algorithms with Dim = 30.

4.2. Experimental Analysis of the Algorithm in Different Dimensions of Function

As the dimension of the function increases, the computational cost of the function increases exponentially. The other setting conditions are shown above when the dimensions are set to 100 and 500. As can be seen from Tables 4 and 5, LSGJO can obtain the optimal solutions in both 100 dimensions and 500 dimensions. To further observe the performance of LSGJO, 100-dimensional convergence curves and the 500-dimensional convergence curve are shown in Figures 5 and 6, respectively. The convergence speed of LSGJO in the image of functions F1–F13 is faster than that of other algorithms, and the convergence accuracy is higher in Figures 5 and 6. The results show that LSGJO has better robustness than other comparison algorithms.

Multidimensional testing not only reflects the robustness of the algorithm but also has a certain practical significance. The traveling salesman problem (TSP) is a typical NPC problem that aims to minimize the path traversing all cities. When there are many cities, the algorithm needs to have the ability to solve multidimensional problems. When swarm intelligence algorithms are used to optimize the weights and thresholds of multilayer neural networks and when the number of layers of the network is large, the number of variables will exceed 500, and the algorithm needs to have the ability to solve 500-dimensional problems. When solving large-scale job-shop scheduling problems, due to a large number of jobs and machines, the algorithm will need to be able to solve multidimensional problems. When a swarm intelligence algorithm is used for wireless sensor coverage optimization, if the coverage area is large, the algorithm needs to have the ability to solve multidimensional problems. In addition, swarm intelligence algorithms are also used in assembly sequence and process planning, and under certain conditions, the ability of algorithms to deal with multidimensional variables is also required.

Table 3. Results and comparison of different algorithms on 23 benchmark functions with Dim = 30. The best results of the experiments are shown in bold.

F(x)	Item	GWO	HHO	ChoA	GJO	EO	WOA	SSA	SO	PSO	MPSO	SOGWO	LSGJO
F1	Ave	1.34×10^{-27}	3.65×10^{-94}	2.98×10^{-7}	2.66×10^{-54}	3.13×10^{-41}	7.02×10^{-73}	1.93×10^{-7}	4.32×10^{-94}	2.96×10^{2}	1.00	3.87×10^{-27}	**0**
	Std	1.53×10^{-27}	2.00×10^{-93}	5.47×10^{-7}	8.99×10^{-54}	5.33×10^{-41}	2.85×10^{-72}	2.61×10^{-7}	2.07×10^{-93}	1.72×10^{2}	2.86	1.30×10^{-26}	**0**
	rank	7.0	2.0	10.0	5.0	6.0	4.0	9.0	3.0	12.0	11.0	8.0	1.0
F2	Ave	1.13×10^{-16}	1.75×10^{-50}	2.64×10^{-6}	2.97×10^{-32}	6.19×10^{-24}	9.23×10^{-51}	2.08	7.39×10^{-43}	3.34×10^{1}	2.54×10^{1}	9.35×10^{-17}	**0**
	Std	1.02×10^{-16}	4.21×10^{-50}	2.65×10^{-6}	6.64×10^{-32}	6.39×10^{-24}	2.67×10^{-50}	1.57	1.61×10^{-42}	1.43×10^{1}	1.65×10^{1}	6.13×10^{-17}	**0**
	rank	8.0	3.0	9.0	5.0	6.0	2.0	10.0	4.0	11.5	11.5	7.0	1.0
F3	Ave	6.17×10^{-5}	8.14×10^{-77}	2.24×10^{1}	3.81×10^{-17}	2.96×10^{-9}	4.43×10^{4}	1.66×10^{3}	1.41×10^{-57}	1.12×10^{4}	1.49×10^{4}	1.10×10^{-4}	**0**
	Std	2.66×10^{-4}	3.96×10^{-76}	5.57×10^{1}	1.26×10^{-16}	8.69×10^{-9}	1.87×10^{4}	8.31×10^{2}	5.97×10^{-57}	1.03×10^{4}	7.41×10^{3}	2.64×10^{-4}	**0**
	rank	6.5	2.0	8.0	4.0	5.0	12.0	3.0	4.0	10.5	10.5	6.5	1.0
F4	Ave	7.43×10^{-7}	4.58×10^{-47}	1.27×10^{-1}	1.36×10^{-14}	3.56×10^{-10}	4.64×10^{1}	1.06×10^{1}	3.25×10^{-40}	9.72	1.95×10^{1}	1.16×10^{-6}	**0**
	Std	6.07×10^{-7}	2.50×10^{-46}	1.43×10^{-1}	5.72×10^{-14}	8.51×10^{-10}	2.63×10^{1}	3.37	9.32×10^{-40}	2.68	6.00	1.18×10^{-6}	**0**
	rank	6.0	2.0	8.0	4.0	5.0	12.0	10.0	3.0	9.0	11.0	7.0	1.0
F5	Ave	2.71×10^{1}	1.86×10^{-2}	2.88×10^{1}	2.79×10^{1}	2.53×10^{1}	2.79×10^{1}	3.98×10^{2}	1.80×10^{1}	1.85×10^{4}	2.78×10^{4}	2.72×10^{1}	$\mathbf{1.83 \times 10^{-2}}$
	Std	8.68×10^{-1}	2.26×10^{-2}	1.98×10^{-1}	7.20×10^{-1}	1.64×10^{-1}	5.37×10^{-1}	1.27×10^{3}	1.24×10^{1}	1.43×10^{4}	4.15×10^{4}	7.65×10^{-1}	$\mathbf{3.27 \times 10^{-2}}$
	rank	6.5	1.5	6.5	6.75	3.5	6.25	10.0	6.0	11.0	12.0	6.5	1.5
F6	Ave	7.79×10^{-1}	1.16×10^{-4}	3.90	2.77	8.70×10^{-6}	3.97×10^{-1}	$\mathbf{2.77 \times 10^{-7}}$	7.37×10^{-1}	3.54×10^{2}	1.21	7.77×10^{-1}	5.84×10^{-4}
	Std	3.60×10^{-1}	1.51×10^{-4}	3.82×10^{-1}	4.87×10^{-1}	5.34×10^{-6}	2.47×10^{-1}	$\mathbf{8.79 \times 10^{-7}}$	5.84×10^{-1}	1.56×10^{2}	3.99	3.63×10^{-1}	1.10×10^{-3}
	rank	7.0	3.0	9.5	9.5	2.0	5.0	1.0	8.0	12.0	10.0	7.0	4.0
F7	Ave	2.23×10^{-3}	1.64×10^{-4}	1.78×10^{-3}	5.14×10^{-4}	1.39×10^{-3}	3.46×10^{-3}	1.65×10^{-1}	2.99×10^{-4}	1.53	3.72×10^{-1}	1.77×10^{-3}	$\mathbf{1.47 \times 10^{-4}}$
	Std	1.05×10^{-3}	2.09×10^{-4}	2.04×10^{-3}	4.42×10^{-4}	6.33×10^{-4}	6.16×10^{-3}	6.79×10^{-2}	2.89×10^{-4}	3.90	1.07	9.29×10^{-4}	$\mathbf{1.45 \times 10^{-4}}$
	rank	7.5	2.0	7.5	4.0	5.0	9.0	10.0	3.0	12.0	11.0	6.0	1.0
F8	Ave	-5.83×10^{3}	-1.26×10^{4}	-5.73×10^{3}	-3.85×10^{3}	-9.23×10^{3}	-1.00×10^{4}	-7.43×10^{3}	1.25×10^{4}	-7.37×10^{3}	-8.82×10^{3}	-6.02×10^{3}	$\mathbf{-1.26 \times 10^{4}}$
	Std	8.82×10^{2}	6.06×10^{1}	6.23×10^{1}	1.14×10^{3}	8.11×10^{2}	1.89×10^{3}	8.41×10^{2}	1.81×10^{2}	9.01×10^{2}	6.26×10^{2}	8.98×10^{2}	$\mathbf{1.30 \times 10^{-1}}$
	rank	8.5	1.75	6.5	11.0	5.0	7.0	6.5	8.0	8.5	5.0	8.0	1.25
F9	Ave	2.53	**0**	2.86	**0**	**0**	3.32×10^{-2}	5.23×10^{1}	2.20	2.20×10^{2}	1.32×10^{2}	3.06	**0**
	Std	4.66	**0**	2.68	**0**	**0**	1.82×10^{-1}	1.64×10^{1}	6.11	3.06×10^{1}	3.16×10^{1}	4.72	**0**
	rank	7.0	2.5	7.0	2.5	2.5	5.0	10.0	7.5	11.5	11.5	8.5	2.5
F10	Ave	1.03×10^{-13}	$\mathbf{8.88 \times 10^{-16}}$	2.00×10^{1}	7.40×10^{-15}	8.70×10^{-15}	4.09×10^{-15}	2.78	2.83×10^{-1}	6.05	3.52	1.03×10^{-13}	$\mathbf{8.88 \times 10^{-16}}$
	Std	1.58×10^{-14}	**0**	1.22×10^{-3}	1.35×10^{-15}	2.17×10^{-15}	3.14×10^{-15}	9.44×10^{-1}	7.38×10^{-1}	1.91	3.16	1.68×10^{-14}	**0**
	rank	6.25	1.5	10.0	3.5	4.5	4.0	9.5	8.5	11.0	11.0	7.0	1.5
F11	Ave	2.58×10^{-3}	**0**	1.09×10^{-2}	**0**	**0**	1.47×10^{-2}	1.80×10^{-2}	7.95×10^{-2}	3.79	2.49×10^{-1}	2.31×10^{-3}	**0**
	Std	5.54×10^{-3}	**0**	2.44×10^{-2}	**0**	**0**	4.66×10^{-2}	1.13×10^{-2}	2.05×10^{-1}	1.44	2.31×10^{-1}	5.37×10^{-3}	**0**
	rank	6.0	2.5	7.5	2.5	2.5	8.5	8.0	10.0	12.0.	11.0	5.0	2.5
F12	Ave	4.65×10^{-2}	$\mathbf{7.18 \times 10^{-6}}$	5.63×10^{-1}	2.59×10^{-1}	3.46×10^{-3}	2.87×10^{-2}	8.57	8.59×10^{-2}	5.89	3.67	5.01×10^{-2}	1.50×10^{-5}
	Std	2.74×10^{-2}	$\mathbf{1.06 \times 10^{-5}}$	2.40×10^{-1}	1.48×10^{-1}	1.89×10^{-2}	2.59×10^{-2}	4.28	1.33×10^{-1}	2.80	1.86	2.90×10^{-2}	2.13×10^{-5}
	rank	5.0	1.0	9.0	8.0	3.0	4.0	12.0	7.0	11.0	10.0	6.0	2.0
F13	Ave	6.08×10^{-1}	1.14×10^{-4}	2.78	1.64	1.64×10^{-2}	5.03×10^{-1}	1.79×10^{1}	2.66×10^{-1}	2.33×10^{1}	9.20	6.34×10^{-1}	$\mathbf{8.71 \times 10^{-5}}$
	Std	2.28×10^{-1}	1.45×10^{-4}	1.38×10^{-1}	2.19×10^{-1}	4.36×10^{-2}	2.37×10^{-1}	1.77×10^{1}	5.68×10^{-1}	2.59×10^{1}	6.25	2.65×10^{-1}	$\mathbf{1.23 \times 10^{-4}}$
	rank	6.0	2.0	6.5	6.5	3.0	6.0	11.0	6.5	12.0	10.0	7.5	1.0

Table 3. *Cont.*

F(x)	Item	GWO	HHO	ChoA	GJO	EO	WOA	SSA	SO	PSO	MPSO	SOGWO	LSGJO
	Ave	4.56	1.13	$\mathbf{9.98 \times 10^{-1}}$	5.82	$\mathbf{9.98 \times 10^{-1}}$	3.09	1.40	$\mathbf{9.99 \times 10^{-1}}$	$\mathbf{9.98 \times 10^{-1}}$	$\mathbf{9.98 \times 10^{-1}}$	3.36	1.36
F14	Std	4.20	3.44×10^{-1}	3.20×10^{-4}	4.45	1.75×10^{-16}	3.28	7.64×10^{-1}	4.13×10^{-3}	2.88×10^{-10}	$\mathbf{9.22 \times 10^{-17}}$	3.31	1.02
	rank	11.0	6.0	3.25	12.0	2.25	9.0	7.5	5.0	2.75	1.75	10.0	7.5
	Ave	7.76×10^{-3}	4.23×10^{-4}	1.32×10^{-3}	2.46×10^{-3}	6.36×10^{-3}	7.07×10^{-4}	1.54×10^{-3}	6.18×10^{-4}	1.23×10^{-2}	4.07×10^{-3}	7.06×10^{-3}	$\mathbf{3.86 \times 10^{-4}}$
F15	Std	9.75×10^{-3}	2.68×10^{-4}	5.84×10^{-5}	6.07×10^{-3}	9.32×10^{-3}	4.22×10^{-4}	3.57×10^{-3}	3.63×10^{-4}	9.54×10^{-3}	7.42×10^{-3}	9.57×10^{-3}	$\mathbf{5.82 \times 10^{-5}}$
	rank	11.5	2.5	3.5	7.0	9.0	4.5	6.0	3.5	11.0	8.0	10.5	1.0
	Ave	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	**−1.03**	−1.03	**−1.03**	**−1.03**	**−1.03**
F16	Std	2.01×10^{-8}	1.13×10^{-9}	2.23×10^{-5}	1.86×10^{-7}	6.32×10^{-16}	6.08×10^{-10}	4.34×10^{-14}	$\mathbf{5.45 \times 10^{-16}}$	8.05×10^{-5}	5.98×10^{-16}	1.81×10^{-8}	1.82×10^{-4}
	rank	7.25	6.25	8.25	7.75	4.75	5.75	5.25	3.75	8.75	4.25	6.75	9.25
	Ave	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.99 \times 10^{-1}}$	$\mathbf{3.99 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$
F17	Std	8.74×10^{-7}	7.79×10^{-4}	7.19×10^{-4}	7.26×10^{-6}	**0**	5.08×10^{-6}	1.90×10^{-14}	**0**	3.62×10^{-5}	**0**	3.09×10^{-6}	3.31×10^{-4}
	rank	5.25	11.75	11.25	6.75	3.75	6.25	4.75	3.75	7.25	3.75	5.75	7.75
	Ave	5.70	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	5.70	3.00	**3.00**	**3.00**	**3.00**
F18	Std	1.48×10^{1}	1.05×10^{-6}	2.20×10^{-4}	4.38×10^{-6}	1.28×10^{-15}	1.68×10^{-4}	3.32×10^{-13}	8.24	6.65×10^{-4}	$\mathbf{1.07 \times 10^{-15}}$	5.51×10^{-5}	3.38×10^{-3}
	rank	11.75	4.75	6.75	5.25	3.75	6.25	4.25	11.25	7.25	3.25	5.75	7.75
	Ave	**−3.86**	**−3.86**	−3.85	**−3.86**	**−3.86**	−3.85	**−3.86**	**−3.86**	**−3.86**	**−3.86**	**−3.86**	**−3.86**
F19	Std	1.99×10^{-3}	5.27×10^{-3}	1.84×10^{-3}	3.86×10^{-3}	2.58×10^{-15}	2.46×10^{-2}	7.77×10^{-12}	$\mathbf{2.43 \times 10^{-15}}$	3.72×10^{-3}	2.68×10^{-15}	2.74×10^{-3}	3.60×10^{-3}
	rank	5.75	8.25	8.25	7.75	3.75	11.75	4.75	3.25	7.25	4.75	6.25	6.75
	Ave	−3.26	−3.10	−2.66	−3.09	−3.27	−3.20	−3.22	**−3.31**	−2.96	−3.27	−3.23	−3.19
F20	Std	9.27×10^{-2}	9.27×10^{-2}	4.55×10^{-1}	2.06×10^{-1}	5.92×10^{-2}	2.25×10^{-1}	5.83×10^{-2}	$\mathbf{3.63 \times 10^{-2}}$	5.18×10^{-1}	6.03×10^{-2}	7.96×10^{-2}	7.58×10^{-2}
	rank	5.75	8.25	11.5	9.5	2.75	8.5	4.0	1.0	11.5	3.25	5.5	6.5
	Ave	−9.64	−5.22	−3.18	−8.52	−8.29	−7.77	−8.48	-1.01×10^{1}	−9.74	−6.82	−9.81	$\mathbf{-1.02 \times 10^{1}}$
F21	Std	1.55	9.18×10^{-1}	2.05	2.85	2.74	2.81	2.90	3.07×10^{-1}	1.77	3.49	1.28	$\mathbf{3.30 \times 10^{-3}}$
	rank	5.5	7.0	9.5	8.0	8.0	9.0	9.0	2.0	5.0	11.0	3.5	1.0
	Ave	−9.87	−5.58	−4.05	−9.68	−8.77	−7.52	−9.29	-1.03×10^{1}	−9.72	−8.07	−9.87	$\mathbf{-1.04 \times 10^{1}}$
F22	Std	1.62	1.51	1.78	1.83	2.79	3.20	2.59	3.07×10^{-1}	2.12	3.20	1.62	$\mathbf{2.36 \times 10^{-3}}$
	rank	4.0	7.0	9.0	6.5	9.0	10.75	8.0	2.0	6.5	10.25	4.0	1.0
	Ave	-1.03×10^{1}	−5.30	−4.46	-1.03×10^{1}	−9.43	−6.77	−8.42	-1.04×10^{1}	$\mathbf{-1.05 \times 10^{1}}$	−8.45	-1.01×10^{1}	$\mathbf{-1.05 \times 10^{1}}$
F23	Std	1.48	9.40×10^{-1}	1.40	9.79×10^{-1}	2.57	3.03	3.36	3.09×10^{-1}	$\mathbf{2.22 \times 10^{-5}}$	3.30	1.75	3.99×10^{-3}
	rank	5.75	7.5	9.0	4.75	8.0	10.0	10.5	3.0	1.25	9.5	7.0	1.75
Total Rank		160.75	96.0	185.25	147.5	108	166.5	174.0	117.0	200.5	195.25	155.0	71.5
Final Rank		8	2	10	5	3	6	9	4	12	11	7	1

Table 4. Results and comparison of different algorithms on 13 benchmark functions with Dim = 100. The best results of the experiments are shown in bold.

F(x)	Item	GWO	HHO	ChoA	GJO	EO	WOA	SSA	SO	PSO	MPSO	SOGWO	LSGJO
F1	Ave	2.64×10^{-12}	2.77×10^{-94}	2.15×10^{-1}	9.33×10^{-28}	4.13×10^{-29}	3.87×10^{-70}	1.45×10^{3}	5.20×10^{-82}	3.96×10^{3}	3.29×10^{4}	2.43×10^{-12}	**0**
	Std	2.73×10^{-12}	1.45×10^{-93}	2.12×10^{-1}	1.85×10^{-27}	5.48×10^{-29}	1.85×10^{-69}	4.54×10^{2}	1.09×10^{-81}	1.37×10^{3}	1.11×10^{4}	1.78×10^{-12}	**0**
	rank	8.0	2.0	9.0	6.0	5.0	4.0	10.0	3.0	11.0	12.0	7.0	1.0
F2	Ave	4.25×10^{-8}	2.34×10^{-49}	3.34×10^{-2}	1.06×10^{-17}	2.14×10^{-17}	6.02×10^{-51}	4.81×10^{1}	1.32×10^{-35}	1.33×10^{2}	2.86×10^{2}	4.12×10^{-8}	**0**
	Std	1.37×10^{-8}	8.16×10^{-49}	1.87×10^{-2}	8.02×10^{-18}	1.44×10^{-17}	2.02×10^{-50}	8.35	1.59×10^{-35}	4.04×10^{1}	3.80×10^{1}	1.40×10^{-8}	**0**
	rank	7.5	3.0	9.0	5.0	6.0	2.0	10.0	4.0.	11.5	11.5.	7.5	1.0
F3	Ave	8.96×10^{2}	8.65×10^{-52}	6.10×10^{4}	1.51	8.96×10^{1}	1.15×10^{6}	5.63×10^{4}	3.16×10^{-38}	1.24×10^{5}	2.35×10^{5}	1.41×10^{3}	**0**
	Std	1.45×10^{3}	4.74×10^{-51}	2.56×10^{4}	5.10	4.05×10^{2}	3.22×10^{5}	2.59×10^{4}	1.70×10^{-37}	6.73×10^{4}	3.96×10^{4}	1.20×10^{3}	**0**
	rank	6.5	2.0	8.5	4.0	5.0	12.0	8.5	3.0	10.5	10.5	6.5	1.0
F4	Ave	1.09	5.52×10^{-49}	7.56×10^{1}	6.67	6.75×10^{-2}	7.91×10^{1}	2.69×10^{1}	1.04×10^{-36}	2.33×10^{1}	6.70×10^{1}	8.12×10^{-1}	**0**
	Std	1.95	1.74×10^{-48}	1.49×10^{1}	9.02	3.55×10^{-1}	2.26×10^{1}	3.76	1.31×10^{-36}	4.53	5.36	6.49×10^{-1}	**0**
	rank	6.0	2.0	11.0	8.5	4.0	12.0	8.0	3.0	8.0	9.5	5.0	1.0
F5	Ave	9.76×10^{1}	4.00×10^{-2}	1.54×10^{2}	9.82×10^{1}	9.65×10^{1}	9.82×10^{1}	1.53×10^{5}	7.38×10^{1}	5.64×10^{5}	2.70×10^{7}	9.80×10^{1}	$\mathbf{3.30 \times 10^{-2}}$
	Std	7.59×10^{-1}	8.62×10^{-2}	1.25×10^{2}	5.60×10^{-1}	9.08×10^{-1}	2.20×10^{-1}	6.64×10^{4}	4.05×10^{1}	4.88×10^{5}	3.12×10^{7}	6.15×10^{-1}	$\mathbf{4.06 \times 10^{-2}}$
	rank	5.5	2.0	9.0	5.75	5.5	5.25	10.0	5.5	11.0	12.0	5.5	1.0
F6	Ave	9.77	$\mathbf{4.26 \times 10^{-4}}$	2.22×10^{1}	1.62×10^{1}	4.03	4.34	1.52×10^{3}	1.32×10^{1}	4.75×10^{3}	2.79×10^{4}	1.07×10^{1}	2.83×10^{-3}
	Std	1.01	$\mathbf{6.28 \times 10^{-4}}$	1.74	9.56×10^{-1}	8.06×10^{-1}	1.42	4.77×10^{2}	1.05×10^{1}	2.30×10^{3}	1.03×10^{4}	1.01	4.29×10^{-3}
	rank	5.25	1.0	8.5	6.0	3.0	5.5	10.0	8.0	11.0	12.0	5.75	2.0
F7	Ave	6.43×10^{-3}	2.01×10^{-4}	1.36×10^{-2}	1.37×10^{-3}	2.30×10^{-3}	4.23×10^{-3}	2.88	2.25×10^{-4}	2.75×10^{1}	8.05×10^{1}	7.61×10^{-3}	$\mathbf{1.29 \times 10^{-4}}$
	Std	2.31×10^{-3}	3.48×10^{-4}	9.10×10^{-3}	1.18×10^{-3}	8.01×10^{-4}	5.41×10^{-3}	5.82×10^{-1}	1.42×10^{-4}	4.66×10^{1}	4.50×10^{1}	2.68×10^{-3}	$\mathbf{1.27 \times 10^{-4}}$
	rank	6.5	2.5	9.0	4.5	4.5	7.0	10.0	2.5	11.5	11.5	7.5	1.0
F8	Ave	-1.61×10^{4}	$\mathbf{-4.19 \times 10^{4}}$	-1.81×10^{4}	-8.23×10^{3}	-2.59×10^{4}	-3.63×10^{4}	-2.16×10^{4}	-4.18×10^{4}	-1.53×10^{4}	-2.22×10^{4}	-1.70×10^{4}	$\mathbf{-4.19 \times 10^{4}}$
	Std	2.37×10^{3}	3.94×10^{1}	1.34×10^{2}	3.31×10^{3}	1.29×10^{3}	5.42×10^{3}	1.86×10^{3}	1.51×10^{2}	2.27×10^{3}	1.53×10^{3}	1.49×10^{3}	$\mathbf{3.66 \times 10^{-1}}$
	rank	10.0	1.75	5.5	11.5	5.0	8.0	7.5	3.0	10.0	6.5	7.5	1.25
F9	Ave	9.74	**0**	1.16×10^{1}	**0**	**0**	7.58×10^{-15}	2.41×10^{2}	8.79	9.60×10^{2}	7.60×10^{2}	1.04×10^{1}	**0**
	Std	7.14	**0**	1.09×10^{1}	**0**	**0**	2.88×10^{-14}	4.13×10^{1}	2.28×10^{1}	2.56	6.98×10^{1}	7.84	**0**
	rank	7.0	2.5	9.0	2.5	2.5	5.0	10.5	8.0	9.0	11.5	8.0	2.5
F10	Ave	1.14×10^{-7}	$\mathbf{8.88 \times 10^{-16}}$	2.00×10^{1}	4.78×10^{-14}	3.59×10^{-14}	4.56×10^{-15}	1.04×10^{1}	4.44×10^{-15}	9.65	1.83×10^{1}	1.37×10^{-7}	$\mathbf{8.88 \times 10^{-16}}$
	Std	4.83×10^{-8}	**0**	1.16×10^{-2}	7.66×10^{-15}	4.82×10^{-15}	2.55×10^{-15}	1.03	0	2.56	1.08	5.72×10^{-8}	**0**
	rank	7.0	1.75	10.5	6.0	5.0	4.0	10.5	2.5	10.5	11	8.0	1.75
F11	Ave	2.96×10^{-3}	**0**	1.98×10^{-1}	**0**	2.55×10^{-4}	**0**	1.54×10^{1}	0	3.66×10^{1}	2.75×10^{2}	4.52×10^{-3}	**0**
	Std	8.15×10^{-3}	**0**	1.94×10^{-1}	**0**	1.40×10^{-3}	**0**	3.84	0	1.97×10^{1}	6.90×10^{1}	9.46×10^{-3}	**0**
	rank	7.0	3.0	9.0	3.0	6.0	3.0	10.0	3.0	11.0	12.0	8.0	3.0
F12	Ave	2.88×10^{-1}	$\mathbf{4.40 \times 10^{-6}}$	1.18	5.90×10^{-1}	4.30×10^{-2}	4.76×10^{-2}	3.60×10^{1}	9.04×10^{-2}	4.38×10^{2}	2.35×10^{7}	3.03×10^{-1}	1.91×10^{-5}
	Std	5.93×10^{-2}	$\mathbf{3.93 \times 10^{-6}}$	2.75×10^{-1}	8.80×10^{-2}	1.18×10^{-2}	1.95×10^{-2}	1.08×10^{1}	2.62×10^{-1}	2.15×10^{3}	6.40×10^{7}	6.40×10^{-2}	3.53×10^{-5}
	rank	5.5	1.0	9.0	7.5	3.0	4.0	10.0	6.5	11.0	12.0	6.5	2.0
F13	Ave	6.75	1.77×10^{-4}	9.77	8.31	6.08	3.02	5.49×10^{3}	1.04	6.72×10^{4}	7.71×10^{7}	6.77	$\mathbf{1.49 \times 10^{-4}}$
	Std	3.43×10^{-1}	2.36×10^{-4}	1.05	2.79×10^{-1}	1.02	9.83×10^{-1}	9.48×10^{3}	1.96	1.35×10^{5}	1.48×10^{8}	5.04×10^{-1}	$\mathbf{2.99 \times 10^{-4}}$
	rank	5.0	1.5	8.5	5.5	6.0	5.0	10.0	6.0	11.0	12.0	6.0	1.5
Total Rank		86.75	26	115.5	75.75	60.5	76.5	125	54	137	132.5	88.75	20.0
Final Rank		7	2	9	5	4	6	10	3	12	11	8	1

Table 5. Results and comparison of different algorithms on 13 benchmark functions with Dim = 500. The best results of the experiments are shown in bold.

F(x)	Item	GWO	HHO	ChoA	GJO	EO	WOA	SSA	SO	PSO	MPSO	SOGWO	LSGJO
F1	Ave	1.66×10^{-3}	3.29×10^{-94}	5.42×10^{2}	6.77×10^{-13}	2.24×10^{-22}	1.59×10^{-65}	9.59×10^{4}	2.66×10^{-71}	3.87×10^{4}	7.59×10^{5}	2.33×10^{-3}	**0**
	Std	4.60×10^{-4}	1.60×10^{-93}	2.78×10^{2}	4.56×10^{-13}	3.19×10^{-22}	8.71×10^{-65}	6.68×10^{3}	4.38×10^{-71}	1.61×10^{4}	3.44×10^{4}	7.75×10^{-4}	**0**
	rank	7.0	2.0	9.0	6.0	5.0	4.0	10.5	3.0	10.5	12.0	8.0	1.0
F2	Ave	1.12×10^{-2}	3.66×10^{-48}	7.76	6.47×10^{-9}	7.65×10^{-14}	2.15×10^{-48}	5.40×10^{2}	1.02×10^{-31}	7.69×10^{2}	2.70×10^{126}	1.14×10^{-2}	**0**
	Std	1.83×10^{-3}	1.70×10^{-47}	2.06	2.29×10^{-9}	3.52×10^{-14}	9.16×10^{-48}	1.65×10^{1}	2.27×10^{-31}	1.37×10^{2}	1.48×10^{127}	1.54×10^{-3}	**0**
	rank	7.5	3.0	9.0	6.0	5.0	2.0	10.0	4.0	11.0	12.0	7.5	1.0
F3	Ave	3.21×10^{5}	2.56×10^{-36}	4.18×10^{6}	3.27×10^{4}	3.94×10^{4}	2.75×10^{7}	1.43×10^{6}	1.84×10^{-15}	2.78×10^{6}	4.54×10^{6}	3.32×10^{5}	**0**
	Std	7.49×10^{4}	1.40×10^{-35}	1.80×10^{6}	2.62×10^{4}	5.81×10^{4}	7.48×10^{6}	6.54×10^{5}	1.01×10^{-14}	1.50×10^{6}	8.31×10^{5}	7.66×10^{4}	**0**
	rank	6.0	2.0	10.5	4.0	5.0	12.0	8.0	3.0	9.5	10.0	7.0	1.0
F4	Ave	6.56×10^{1}	1.66×10^{-47}	9.69×10^{1}	8.21×10^{1}	7.16×10^{1}	8.36×10^{1}	4.02×10^{1}	7.32×10^{-34}	3.56×10^{1}	9.92×10^{1}	6.57×10^{1}	**0**
	Std	7.16	8.38×10^{-47}	1.65	3.99	1.61×10^{1}	1.77×10^{1}	2.65	1.04×10^{-33}	4.80	2.26×10^{-1}	5.17	**0**
	rank	8.0	2.0	8.0	8.0	9.5	11.0	5.5	3.0	6.0	8.0	8.0	1.0
F5	Ave	4.98×10^{2}	2.82×10^{-1}	2.47×10^{5}	4.98×10^{2}	4.98×10^{2}	4.96×10^{2}	3.77×10^{7}	4.08×10^{2}	4.07×10^{7}	2.53×10^{9}	4.98×10^{2}	$\mathbf{1.81 \times 10^{-1}}$
	Std	2.50×10^{-1}	3.48×10^{-1}	3.49×10^{5}	1.64×10^{-1}	1.29×10^{-1}	3.38×10^{-1}	4.22×10^{6}	1.80×10^{2}	4.38×10^{7}	1.91×10^{8}	4.20×10^{-1}	$\mathbf{3.39 \times 10^{-1}}$
	rank	4.75	4.0	9.0	4.25	3.75	4.0	10.0	5.5	11.0	12.0	6.75	3.0
F6	Ave	9.15×10^{1}	$\mathbf{1.96 \times 10^{-3}}$	5.82×10^{2}	1.10×10^{2}	8.71×10^{1}	3.45×10^{1}	9.29×10^{4}	6.24×10^{1}	3.85×10^{4}	7.64×10^{5}	9.17×10^{1}	9.81×10^{-3}
	Std	2.13	$\mathbf{3.46 \times 10^{-3}}$	1.62×10^{2}	1.22	1.73	6.35	5.96×10^{3}	5.35×10^{1}	1.52×10^{4}	3.00×10^{4}	2.28	1.30×10^{-2}
	rank	5.5	1.0	9.0	5.5	4.5	5.0	10.5	6.0	10.5	12.0	6.5	2.0
F7	Ave	4.73×10^{-2}	2.10×10^{-4}	2.42	6.23×10^{-3}	3.97×10^{-3}	5.48×10^{-3}	2.69×10^{2}	1.69×10^{-4}	2.80×10^{3}	1.86×10^{4}	4.51×10^{-2}	$\mathbf{1.59 \times 10^{-4}}$
	Std	1.13×10^{-2}	2.76×10^{-4}	1.65	3.74×10^{-3}	1.43×10^{-3}	6.26×10^{-3}	3.71×10^{1}	1.55×10^{-4}	2.11×10^{3}	1.66×10^{3}	1.40×10^{-2}	$\mathbf{1.29 \times 10^{-4}}$
	rank	7.5	3.0	9.0	5.5	4.0	5.5	10.0	2.0	11.5	11.5	7.5	1.5
F8	Ave	-5.68×10^{4}	$\mathbf{-2.09 \times 10^{5}}$	-8.47×10^{4}	-2.35×10^{4}	-7.55×10^{4}	-1.81×10^{5}	-5.97×10^{4}	-2.08×10^{5}	-3.67×10^{4}	-6.06×10^{4}	-5.70×10^{4}	$\mathbf{-2.09 \times 10^{5}}$
	Std	3.68×10^{3}	2.72×10^{3}	6.33×10^{2}	1.34×10^{4}	4.18×10^{3}	2.86×10^{4}	3.85×10^{3}	1.40×10^{3}	5.35×10^{3}	3.70×10^{3}	8.69×10^{3}	**5.20**
	rank	7.5	2.75	3.5	11.5	7.0	8.0	7.5	3.0	10.0	6.5	9.5	1.25
F9	Ave	7.82×10^{1}	**0**	2.32×10^{2}	5.94×10^{-12}	9.09×10^{-14}	3.03×10^{-14}	3.20×10^{3}	5.52	4.78×10^{3}	5.97×10^{3}	7.40×10^{1}	**0**
	Std	2.23×10^{1}	**0**	5.80×10^{1}	1.49×10^{-11}	2.78×10^{-13}	1.66×10^{-13}	9.92×10^{1}	1.95×10^{1}	5.25×10^{2}	1.69×10^{2}	1.88×10^{1}	**0**
	rank	8.0	1.5	9.0	5.0	4.0	3.0	10.0	6.5	11.5	11.5	6.5	1.5
F10	Ave	1.88×10^{-3}	$\mathbf{8.88 \times 10^{-16}}$	2.01×10^{1}	3.31×10^{-8}	5.85×10^{-13}	4.20×10^{-15}	1.42×10^{1}	4.68×10^{-15}	1.36×10^{1}	2.02×10^{1}	2.14×10^{-3}	$\mathbf{8.88 \times 10^{-16}}$
	Std	3.24×10^{-4}	**0**	1.25×10^{-2}	1.06×10^{-8}	3.23×10^{-13}	2.79×10^{-15}	3.04×10^{-1}	9.01×10^{-16}	4.02	5.71×10^{-2}	3.37×10^{-4}	**0**
	rank	7.0	1.5	10.0	6.0	5.0	3.5	10.5	3.5	10.5	11.0	8.0	1.5
F11	Ave	6.51×10^{-3}	**0**	4.67	2.47×10^{-13}	9.62×10^{-17}	1.05×10^{-2}	8.45×10^{2}	0	3.41×10^{2}	6.91×10^{3}	5.30×10^{-2}	**0**
	Std	2.42×10^{-2}	**0**	1.56	4.74×10^{-13}	3.84×10^{-17}	5.73×10^{-2}	6.80×10^{1}	0	1.27×10^{2}	3.02×10^{2}	6.58×10^{-2}	**0**
	rank	6.0	2.0	9.0.	5.0	4.0	7.0	10.5	2.0.	10.5	12.0	8.0	2.0
F12	Ave	7.52×10^{-1}	$\mathbf{2.00 \times 10^{-6}}$	6.84×10^{4}	9.32×10^{-1}	5.84×10^{-1}	1.05×10^{-1}	1.41×10^{6}	1.47×10^{-1}	2.42×10^{5}	4.98×10^{9}	7.60×10^{-1}	2.02×10^{-5}
	Std	4.46×10^{-2}	$\mathbf{3.43 \times 10^{-6}}$	2.20×10^{5}	2.31×10^{-2}	2.42×10^{-2}	4.44×10^{-2}	7.36×10^{5}	3.56×10^{-1}	4.28×10^{5}	4.32×10^{8}	4.53×10^{-2}	3.59×10^{-5}
	rank	6.0	1.0	9.0	5.5	4.5	4.0	11.0	6.0	10.0	12.0	7.0	2.0
F13	Ave	5.03×10^{1}	9.07×10^{-4}	2.78×10^{5}	4.80×10^{1}	4.92×10^{1}	1.79×10^{1}	3.80×10^{7}	6.19	5.78×10^{6}	1.04×10^{10}	5.10×10^{1}	$\mathbf{4.19 \times 10^{-4}}$
	Std	1.58	1.57×10^{-3}	7.97×10^{5}	4.95×10^{-1}	2.54×10^{-1}	7.17	1.03×10^{7}	1.25×10^{1}	7.57×10^{6}	8.90×10^{8}	1.60	$\mathbf{7.61 \times 10^{-4}}$
	rank	6.0	2.0	9.0	4.5	4.5	5.5	11.0	5.5	10.0	12.0	7.0	1.0
Total Rank		86.75	27.75	104	76.75	65.75	74	125	51	132.5	142.5	97.25	19.75
Final Rank		6	2	8	9	4	5	10	3	12	11	7	1

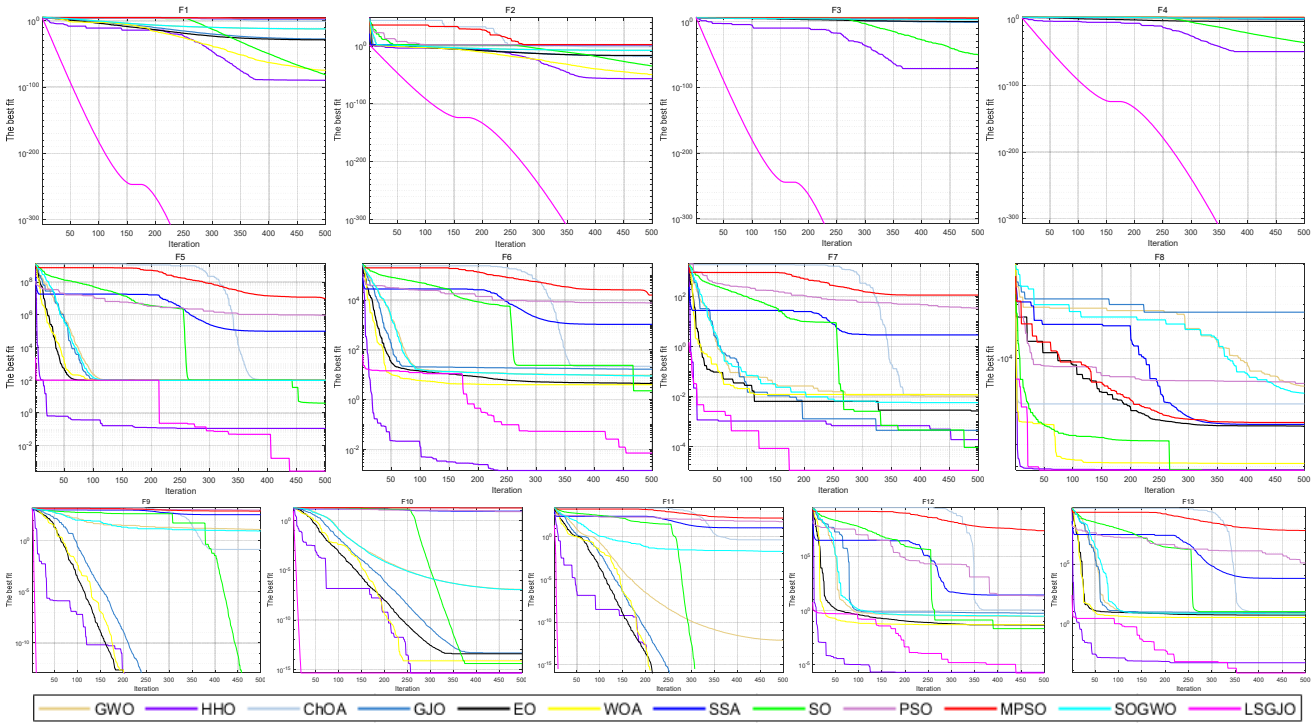

Figure 5. The convergence curves of the LSGJO and other comparison algorithms with Dim = 100.

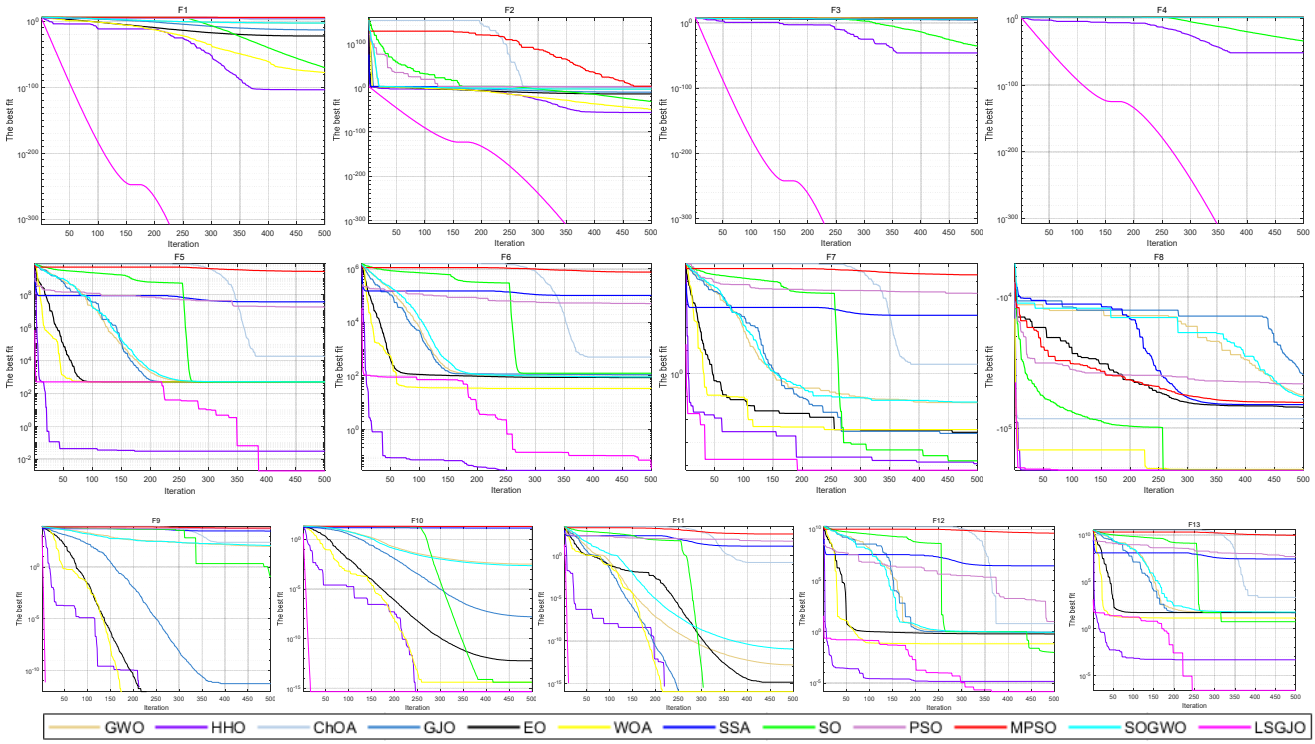

Figure 6. The convergence curves of the LSGJO and other comparison algorithms with Dim = 500.

4.3. Convergence Behavior Analysis

This experiment is used to observe the convergence behavior of LSGJO with 30 dimensions and 500 iterations. The convergence process of the LSGJO is shown in Figure 7. The diagram in the first column is a three-dimensional plot of the benchmark function. The diagram in the second column is the convergence curve of the LSGJO, which is the

optimal value of the current iteration. It can be seen that LSGJO converges quickly on the unimodal function, and the ladder shape appears on the multimodal function, which shows that the improved algorithm has better exploration ability and exploitation ability. The diagram in the third column is the trajectory of the first golden jackal in the first dimension. The significant fluctuation at the beginning is due to the global optimization in the early iteration stage. The trajectory fluctuates significantly in the later stage of the iteration because of the dynamic lens-imaging learning strategy added, which can avoid falling into iterative stagnation. The fourth column diagram is the average fitness of the overall solution, which is used to evaluate the overall performance of the population. The curve will be relatively high in the initial iteration, and the average fitness will likely be stable as the number of iterations increases. The fifth column diagram shows the historical position of the search agent in the iterative process. In the search history of functions F1–F4 and functions F9–F11, the point positions are more clustered, indicating that the fitness of the search agent is small, and the next iteration will be a local search in this area. In the search history of functions F5, F6, and F16, many points are scattered, indicating that if the optimal value is not found quickly, other search agents will continue to search for the optimal solution.

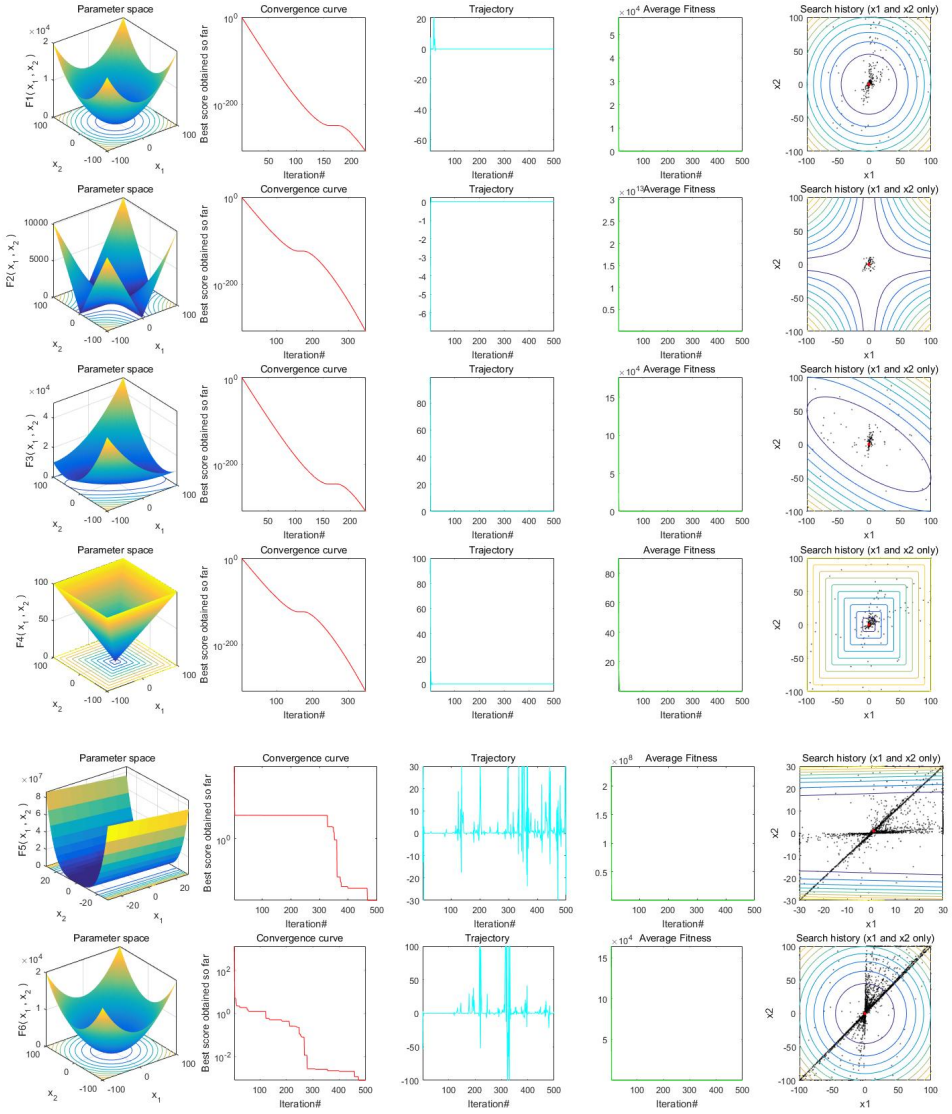

Figure 7. *Cont.*

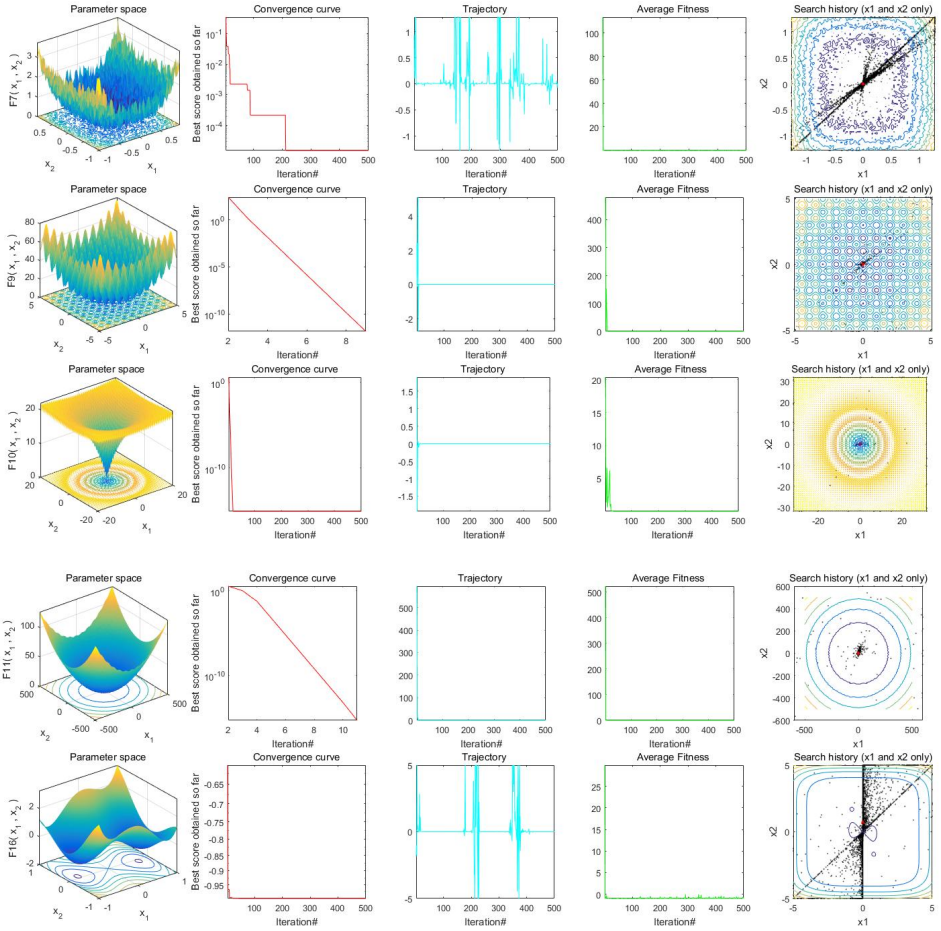

Figure 7. The convergence curve, the trajectories, the average fitness history, and the search history of certain functions.

4.4. Statistical Analysis

In the statistical processing of experimental data, each experiment's average value and standard deviation have been calculated and can be used to judge the algorithms' quality. In order to further verify the significant differences between the proposed algorithm and other algorithms, the Wilcoxon rank-sum test is performed at a significance level of 0.05 [42]. If the *p*-value is > 0.05, we should consider the performance of these two algorithms to be similar, and the values are underlined. The performance of all the algorithms is ranked by the Friedman rank test [43]. The results of the Wilcoxon rank sum test are shown in Table 6. NaN indicates that significance cannot be determined. The total number of significant differences is shown in the last column. In F9 and F11, some algorithms do not have significant differences in 30 and 100 dimensions. However, they have significant differences in 500 dimensions, indicating that other algorithms have poor performance in high dimensions, while LSGJO has good performance. The sorting results by Friedman show that LSGJO ranks first in both low-dimensional and high-dimensional functions.

Table 6. Statistical analysis results.

F(x)	Dim	GWO	HHO	ChoA	GJO	EO	WOA	SSA	SO	PSO	MPSO	SOGWO	Total
F1	30	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	100	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	500	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
F2	30	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	100	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	500	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
F3	30	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	100	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	500	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
F4	30	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	100	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
	500	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	10
F5	30	3.02×10^{-11}	3.78×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	8.15×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
	100	3.02×10^{-11}	$\underline{7.84 \times 10^{-1}}$	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.69×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	9
	500	3.02×10^{-11}	$\underline{1.33 \times 10^{-1}}$	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.69×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	9
F6	30	1.61×10^{-10}	2.25×10^{-4}	3.02×10^{-11}	3.02×10^{-11}	2.23×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.50×10^{-11}	1.78×10^{-10}	10
	100	3.02×10^{-11}	3.85×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.08×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
	500	3.02×10^{-11}	4.71×10^{-4}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.78×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
F7	30	3.02×10^{-11}	$\underline{3.04 \times 10^{-1}}$	6.12×10^{-10}	8.66×10^{-5}	3.69×10^{-11}	2.38×10^{-7}	3.02×10^{-11}	3.78×10^{-2}	4.11×10^{-7}	3.02×10^{-11}	4.98×10^{-11}	9
	100	3.02×10^{-11}	$\underline{2.90 \times 10^{-1}}$	4.08×10^{-11}	4.62×10^{-10}	3.02×10^{-11}	1.43×10^{-8}	3.02×10^{-11}	1.38×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	9
	500	3.02×10^{-11}	$\underline{7.96 \times 10^{-1}}$	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.17×10^{-9}	3.02×10^{-11}	$\underline{9.82 \times 10^{-1}}$	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	8
F8	30	3.02×10^{-11}	2.16×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.34×10^{-11}	3.02×10^{-11}	1.09×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
	100	3.02×10^{-11}	3.34×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
	500	3.02×10^{-11}	3.03×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.41×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
F9	30	4.43×10^{-12}	NaN	1.21×10^{-12}	NaN	NaN	NaN	1.21×10^{-12}	8.87×10^{-7}	1.21×10^{-12}	1.21×10^{-12}	4.47×10^{-12}	6
	100	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN	NaN	$\underline{1.61 \times 10^{-1}}$	1.21×10^{-12}	3.45×10^{-7}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	6
	500	1.21×10^{-12}	NaN	1.21×10^{-12}	9.51×10^{-13}	$\underline{8.14 \times 10^{-2}}$	$\underline{3.34 \times 10^{-1}}$	1.21×10^{-12}	2.16×10^{-2}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	7
F10	30	1.13×10^{-12}	NaN	1.21×10^{-12}	2.43×10^{-13}	4.16×10^{-14}	1.16×10^{-8}	1.21×10^{-12}	1.20×10^{-13}	1.21×10^{-12}	1.21×10^{-12}	1.10×10^{-12}	9
	100	1.21×10^{-12}	NaN	1.19×10^{-12}	9.98×10^{-13}	5.94×10^{-13}	3.86×10^{-9}	1.21×10^{-12}	1.69×10^{-14}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	9
	500	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.05×10^{-7}	1.21×10^{-12}	4.16×10^{-14}	1.18×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	9
F11	30	6.62×10^{-4}	NaN	1.21×10^{-12}	NaN	NaN	$\underline{3.34 \times 10^{-1}}$	1.21×10^{-12}	1.10×10^{-2}	1.21×10^{-12}	1.21×10^{-12}	2.16×10^{-2}	6
	100	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN	$\underline{3.34 \times 10^{-1}}$	NaN	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	5
	500	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}	1.97×10^{-11}	3.34×10^{-1}	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	8

Table 6. *Cont.*

F(x)	Dim	GWO	HHO	ChoA	GJO	EO	WOA	SSA	SO	PSO	MPSO	SOGWO	Total
	30	3.02×10^{-11}	3.50×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	8.35×10^{-8}	3.02×10^{-11}	3.02×10^{-11}	3.34×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
F12	100	3.02×10^{-11}	5.01×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.34×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	9
	500	3.02×10^{-11}	1.34×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	5.57×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
	30	3.02×10^{-11}	7.51×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	1.76×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
F13	100	3.02×10^{-11}	1	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
	500	3.02×10^{-11}	4.21×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	10
F14	2	6.20×10^{-1}	2.38×10^{-3}	1.84×10^{-2}	4.84×10^{-2}	1.52×10^{-11}	5.11×10^{-1}	4.65×10^{-5}	2.53×10^{-7}	4.42×10^{-6}	5.14×10^{-12}	1.49×10^{-1}	8
F15	4	9.93×10^{-2}	1.70×10^{-2}	3.02×10^{-11}	3.56×10^{-4}	1.68×10^{-4}	2.75×10^{-3}	2.87×10^{-10}	1.37×10^{-1}	6.70×10^{-11}	1.38×10^{-6}	1.22×10^{-1}	7
F16	2	3.02×10^{-11}	3.02×10^{-11}	8.35×10^{-8}	3.02×10^{-11}	1.25×10^{-11}	3.02×10^{-11}	3.01×10^{-11}	1.25×10^{-11}	4.03×10^{-3}	1.34×10^{-11}	3.02×10^{-11}	10
F17	2	3.02×10^{-11}	8.15×10^{-11}	9.52×10^{-4}	1.56×10^{-8}	1.21×10^{-12}	3.08×10^{-8}	2.75×10^{-11}	1.21×10^{-12}	1.69×10^{-9}	1.21×10^{-12}	2.15×10^{-10}	10
F18	2	7.77×10^{-9}	3.02×10^{-11}	1.11×10^{-6}	3.34×10^{-11}	2.49×10^{-11}	3.82×10^{-9}	3.02×10^{-11}	1.04×10^{-7}	3.83×10^{-6}	1.77×10^{-11}	1.20×10^{-8}	10
F19	3	2.60×10^{-5}	1.27×10^{-2}	6.36×10^{-5}	1.58×10^{-1}	1.34×10^{-11}	4.64×10^{-1}	3.02×10^{-11}	1.25×10^{-11}	9.03×10^{-4}	2.36×10^{-12}	5.86×10^{-6}	9
F20	6	3.99×10^{-4}	1.43×10^{-5}	3.82×10^{-10}	3.03×10^{-2}	4.51×10^{-6}	5.08×10^{-3}	1.33×10^{-2}	6.86×10^{-10}	3.87×10^{-1}	4.78×10^{-6}	6.67×10^{-3}	9
F21	4	2.84×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	1.55×10^{-9}	5.64×10^{-4}	5.19×10^{-7}	6.63×10^{-1}	8.74×10^{-2}	4.86×10^{-9}	2.68×10^{-2}	5.75×10^{-2}	7
F22	4	2.84×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	4.44×10^{-7}	1.83×10^{-3}	5.97×10^{-9}	1.95×10^{-3}	4.13×10^{-3}	6.23×10^{-5}	6.60×10^{-1}	4.29×10^{-1}	7
F23	4	8.88×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	1.25×10^{-7}	8.14×10^{-6}	5.97×10^{-9}	9.51×10^{-6}	7.61×10^{-3}	2.00×10^{-9}	1.99×10^{-2}	1.33×10^{-1}	8

5. Real-World Engineering Design Problems

In order to verify the optimization performance of LSGJO in real-world engineering design problems, this paper introduces three constraint problems, namely the speed reducer design problem [44], the gear train design problem [45], and the multiple-disk clutch design problem [46]. The number of iterations for all algorithms is set to 500, and the population size is set to 30.

5.1. Speed Reducer Design Problem

The speed reducer is widely used in mechanical products. According to its use occasion, its specific functions are mainly manifested as reducing the speed, increasing the torque, reducing the inertia of the movement mechanism, and so on. The main goal of this design problem is to minimize the weight of the speed reducer. The variables include face width y_1, the module of teeth y_2, the number of teeth in the pinion y_2, the length of the first shaft y_3, the length of the second shaft y_4, the diameter of the first shaft y_5, and the diameter of the second shaft y_6. The speed reducer is shown in Figure 8. The mathematical model of the speed reducer design is stated in Appendix A, Equation (A1).

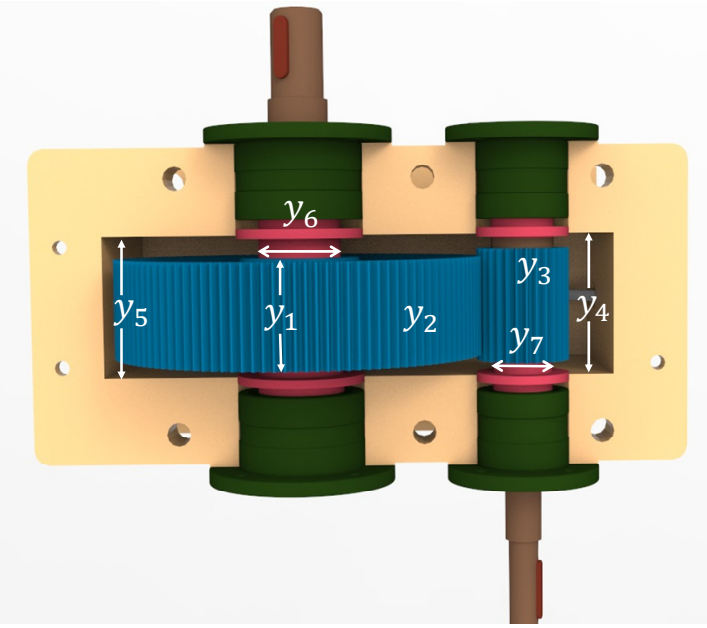

Figure 8. Speed reducer design problem.

Table 7 shows the optimal solution to the speed reducer design problem obtained by 10 popular intelligent algorithms, namely the proposed algorithm in this paper, GWO, HHO, ChoA, GJO, EO, WOA, SO, MPSO, and SOGWO. The experimental results show that LSGJO is better than other algorithms. The minimum weight of the speed reducer is Minimize$f\left(\vec{y}\right)$ = 2994.4711, with the optimal solution $\vec{y}$ = {3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867}.

Table 7. Comparison results of speed reducer design problem. The best results of the experiments are shown in bold.

Algorithm	y_1	y_2	y_3	y_4	y_5	y_6	y_7	Optimum Value
GWO	3.5023	0.7000	17.0000	7.4808	7.7251	3.3631	5.2872	3000.8341
HHO	3.5026	0.7000	17.0000	8.0413	8.0301	3.4989	5.2868	3049.1657
ChoA	3.6000	0.7000	17.0000	7.3000	8.3000	3.4427	5.3656	3121.8909
GJO	3.5584	0.7002	17.0000	7.4252	8.0148	3.3849	5.2873	3035.4171
EO	3.5000	0.7000	17.0000	7.3000	8.3000	3.3502	5.2869	3007.4366

Table 7. *Cont.*

Algorithm	y_1	y_2	y_3	y_4	y_5	y_6	y_7	Optimum Value
WOA	3.5000	0.7000	17.0000	7.9128	7.9308	3.5822	5.3606	3116.4355
SO	3.5000	0.7000	17.0000	7.8849	7.7153	3.3519	5.2867	3000.2703
MPSO	3.5000	0.7000	17.0000	7.3000	8.3000	3.3502	5.2869	3046.7137
SOGWO	3.5067	0.7000	17.0000	7.3000	7.9316	3.3534	5.2930	3006.7350
LSGJO	3.5000	0.7000	17.0000	7.3000	7.7153	3.3502	5.2867	**2994.4711**

5.2. Gear Train Design Problem

The gear train plays an essential role in watches, clutches, differentials, machine tools, fans, mixers, and many other products. It is one of the most common mechanisms in the mechanical field. The main objective of the gear train design problem is to minimize the gear ratio. The variable is the number of teeth of four gears. The gear train is shown in Figure 9. The mathematical model of the gear train design is stated in Appendix A, Equation (A2).

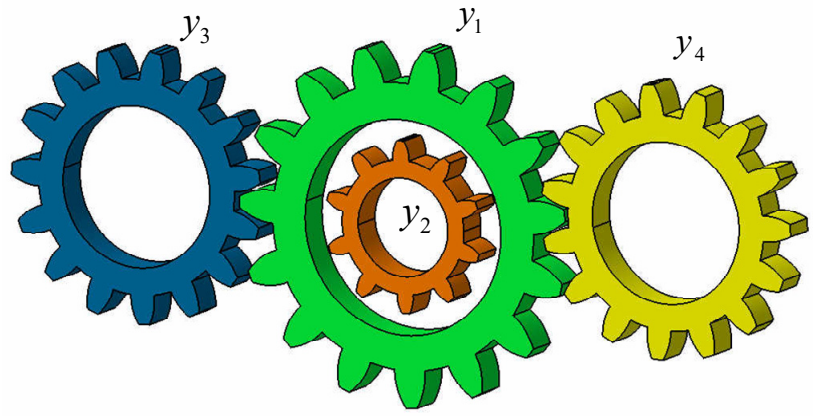

Figure 9. Gear train design problem.

Table 8 shows the optimal solution to the gear train design problem by 14 popular intelligent algorithms, namely LSGJO, GWO, GJO, PSO, bat algorithm (BA) [47], ant colony optimization (ACO) [48], simulated annealing (SA) [49], flower pollination algorithm (FPA) [50], dragonfly algorithm (DA) [51], moth–flame optimization algorithm (MFO) [52], polar bear optimization algorithm (PBO) [53], firefly algorithm (FA) [54], SOGWO, and EO. Experimental results show that LSGJO is significantly superior to other algorithms. The minimum transmission ratio of the gear train is Minimize$f\left(\vec{y}\right) = 2.63 \times 10^{-19}$, and the optimal solution $\vec{y} = \{3.17 \times 10^1, 1.20 \times 10^1, 1.20 \times 10^1, 3.15 \times 10^1\}$.

Table 8. Comparison results of gear train design problem. The best results of the experiments are shown in bold.

Algorithm	y_1	y_2	y_3	y_4	Optimum Value
GWO	4.03×10^1	2.46×10^1	1.20×10^1	5.08×10^1	1.18×10^{13}
GJO	5.00×10^1	1.71×10^1	1.26×10^1	2.98×10^1	1.52×10^{13}
PSO	5.13×10^1	2.10×10^1	1.48×10^1	4.78×10^1	3.08×10^{-4}
BA	5.75×10^1	1.95×10^1	1.86×10^1	4.37×10^1	1.53×10^{-11}
ACO	5.15×10^1	2.14×10^1	1.58×10^1	4.73×10^1	2.87×10^{-5}
SA	5.13×10^1	2.13×10^1	1.50×10^1	4.74×10^1	1.71×10^{-4}
FPA	5.12×10^1	2.25×10^1	1.80×10^1	5.59×10^1	4.83×10^{-11}
DA	5.24×10^1	1.70×10^1	2.30×10^1	5.17×10^1	3.02×10^{-11}
MFO	4.42×10^1	1.88×10^1	2.11×10^1	5.70×10^1	1.44×10^{-14}

Table 8. *Cont.*

Algorithm	y_1	y_2	y_3	y_4	Optimum Value
PBO	5.01×10^1	2.33×10^1	1.48×10^1	4.79×10^1	1.37×10^{-15}
FA	5.01×10^1	2.44×10^1	1.40×10^1	4.64×10^1	6.52×10^{-13}
SOGWO	4.81×10^1	2.99×10^1	1.38×10^1	5.94×10^1	2.35×10^{-11}
EO	4.49×10^1	1.28×10^1	2.93×10^1	5.79×10^1	5.76×10^{-14}
LSGJO	3.17×10^1	1.20×10^1	1.20×10^1	3.15×10^1	$\mathbf{2.63 \times 10^{-19}}$

5.3. Multiple-Disk Clutch Design Problem

The multiple-disk clutch is widely used in mechanical transmission systems in machine tools, steel rolling, metallurgical mining, handling, ship fishery equipment, etc. The main objective of the multiple-disk clutch design problem is to minimize the weight of the clutch. The variables include the internal surface area radius y_1, the external surface radius y_2, the disc thickness y_3, the driving force y_4, and the number of friction surfaces y_5. The multiple-disk clutch is shown in Figure 10. The mathematical model of multiple-disk clutch design is stated in Appendix A, Equation (A3).

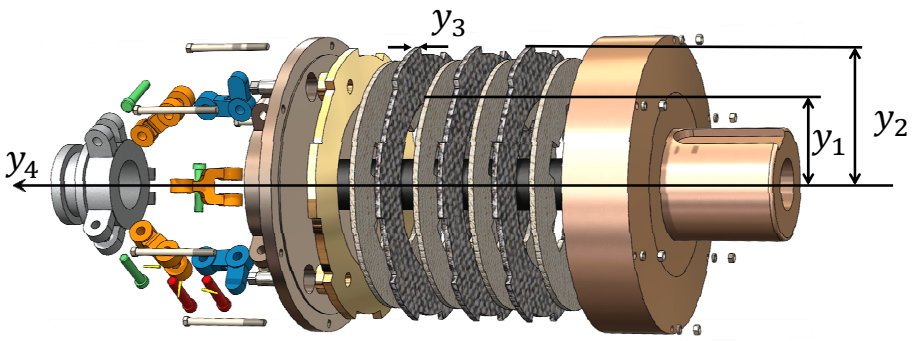

Figure 10. Multiple-disk clutch design problem.

Table 9 is the optimal solution for multiple-disk clutch design obtained by 11 advanced intelligent algorithms. These intelligent algorithms are LSGJO, GWO, GJO, ChoA, ant lion optimizer (ALO) [55], multi-verse optimizer (MVO) [56], ACO, sine cosine algorithm (SCA) [57], EO, SOGWO, and MPSO. The experimental results show that LSGJO is better than the other 10 algorithms; the minimum weight of the multiple-disk clutch is Minimize$f\left(\vec{y}\right)$ = 0.2352425, and the optimal solution $\vec{y}$ = {69.9999928, 90.0000000, 1.0000000, 945.1761801, 2.0000000}.

Table 9. Comparison results of multiple-disk clutch design problem. The best results of the experiments are shown in bold.

Algorithm	y_1	y_2	y_3	y_4	y_5	Optimum Value
GWO	69.9898148	90.0000000	1.0000000	565.6572929	2.0000000	0.2353473
GJO	69.9906674	90.0000000	1.0000000	524.8143417	2.0000000	0.2353385
ChoA	69.9657899	90.0000000	1.0000000	61.9191980	2.0000000	0.2355945
ALO	69.9999996	90.0000000	1.0000000	246.9492771	2.0000000	**0.2352425**
MVO	69.9880862	21.4000000	15.8000000	912.4722915	2.0000000	0.2353651
SCA	69.2616541	90.0000000	1.0000000	57.9068873	2.0000000	0.2428013
EO	70.0000000	90.0000000	1.0000000	45.1874349	2.0000000	**0.2352425**
SOGWO	69.9989554	90.0000000	1.0000000	525.2165780	2.0000000	0.2352532
MPSO	70.0000000	90.0000000	1.0000000	996.1753765	2.0000000	**0.2352425**
LSGJO	69.9999928	90.0000000	1.0000000	945.1761801	2.0000000	**0.2352425**

6. Discussion

Every metaheuristic should be critically evaluated [58]. Metaheuristic algorithms are created to solve practical problems. The algorithm proposed in this paper only proves the effectiveness of numerical optimization problems and can not prove that the algorithm is universal in other problems. The algorithm proposed in this paper only obtains the approximate solution to the optimization problem, but not the exact solution, which is worthy of further improvement and also our future work. The improved algorithm in this paper is closer to the hunting state of the real golden jackal than the original algorithm, but there is still a gap between it and the real hunting state. It is worth studying how to establish a mathematical model consistent with the actual hunting state. Determining which components of the algorithm have an impact on the optimization problem is also an important issue that will help us further improve the algorithm.

7. Conclusions

In order to improve the efficiency of GJO in global numerical optimization and practical design problems, a hybrid GJO and golden sine algorithm with dynamic lens-imaging learning is proposed in this paper. LSGJO makes two effective improvements over the GJO. Firstly, the candidate solution of the optimal solution is generated by the dynamic lens-imaging learning strategy, which increases the possibility of finding the optimal value quickly. Secondly, novel dual golden spiral update rules are introduced in the exploitation stage to accelerate convergence and avoid falling into local optima. The algorithm's global search ability and local search ability are enhanced and achieve balance with each other by combining the two proposed improvements. Twenty-three benchmark functions were tested to evaluate the performance of the LSGJO, including three dimensions (30, 100, 500). Experimental results and statistical data show that the algorithm proposed in this paper has a fast convergence speed, high convergence precision, strong robustness, and stable searching performance. Compared to 11 state-of-the-art optimization algorithms, LSGJO has excellent competitiveness. In addition, LSGJO was successfully applied to three real-world engineering problems in the mechanical field (speed reducer design, gear train design, and multiple-disk clutch design), and its optimization effect was better than that of other algorithms.

In the future, the potential of LSGJO will be explored and focused on applications, and research in other directions, such as (1) path planning for unmanned aerial vehicles (UAVs), (2) the use of the oppositional learning method in the initialization stage, and (3) a multiobjective optimization algorithm based on LSGJO, will be studied and applied to feature selection and process parameter optimization.

Author Contributions: Conceptualization, P.Y., T.Z. and L.Y.; methodology, P.Y. and L.Y.; software, P.Y.; writing—original draft, P.Y.; writing—review and editing, P.Y., T.Z. and L.Y.; data curation, W.Z.; visualization W.Z.; supervision, T.Z. and Y.L.; funding acquisition, T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: National Natural Science Foundation (Grant No. 72061006, 71761007); Academic New Seedling Foundation Project of Guizhou Normal University (Grant No. Qianshixinmiao [2021] A30); Growth Project for Young Scientific and Technological Talents in General Colleges and Universities of Guizhou Province (Grant No. Qianjiaohe KY [2022] 167); Guizhou Provincial Science and Technology Projects (Grant No. Qiankehejichu-ZK [2022] General 320).

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.

Appendix A

See Equations (A1)–(A3)

$$\vec{y} = \{y_1, y_2, y_3, y_4, y_5, y_6, y_7\}$$

Minimize :

$$f\left(\vec{y}\right) = 0.7854 y_1 y_2^2 \left(3.3333 y_3^2 + 14.9334 y_3 - 43.0934\right) - 1.508 y_1 \left(y_6^2 + y_7^2\right) + 7.4777\left(y_6^3 + y_7^3\right) + 0.7854\left(y_4 y_6^2 + y_5 y_7^2\right)$$

Subject to :

$$h_1\left(\vec{y}\right) = \frac{27}{y_1 y_2^2 y_3} - 1 \le 0$$

$$h_2\left(\vec{y}\right) = \frac{397.5}{y_1 y_2^2 y_3^2} - 1 \le 0$$

$$h_3\left(\vec{y}\right) = \frac{1.93 y_4^3}{y_2 y_6^4 y_3} - 1 \le 0$$

$$h_4\left(\vec{y}\right) = \frac{1.93 y_5^3}{y_2 y_7^4 y_3} - 1 \le 0$$

$$h_5\left(\vec{y}\right) = \frac{\left[\left(745\left(y_4 / y_2 y_3\right)\right)^2 + 16.9 * 10^6\right]^{1/2}}{110 y_6^3} - 1 \le 0 \quad \text{(A1)}$$

$$h_6\left(\vec{y}\right) = \frac{\left[\left(745\left(y_5 / y_2 y_3\right)\right)^2 + 157.5 * 10^6\right]^{1/2}}{85 y_7^3} - 1 \le 0$$

$$h_7\left(\vec{y}\right) = \frac{y_2 y_3}{40} - 1 \le 0$$

$$h_8\left(\vec{y}\right) = \frac{5 y_2}{y_1} - 1 \le 0$$

$$h_9\left(\vec{y}\right) = \frac{y_1}{12 y_2} - 1 \le 0$$

$$h_{10}\left(\vec{y}\right) = \frac{1.5 y_6 + 1.9}{y_4} - 1 \le 0$$

$$h_{11}\left(\vec{y}\right) = \frac{1.1 y_7 + 1.9}{y_5} - 1 \le 0$$

Variable range :

$$2.6 \le y_1 \le 3.6, 0.7 \le y_2 \le 0.8, 17 \le y_3 \le 28,$$
$$7.3 \le y_4 \le 8.3, 7.3 \le y_5 \le 8.3, 2.9 \le y_6 \le 3.9, 5.0 \le y_7 \le 5.5$$

$$\vec{y} = \{y_1, y_2, y_3, y_4\}$$

Minimize :

$$f\left(\vec{y}\right) = \left(\frac{1}{6.931} - \frac{y_3 y_2}{y_1 y_4}\right)^2 \quad \text{(A2)}$$

Variable range :

$$12 \le y_1 \le 60, 12 \le y_2 \le 60, 12 \le y_3 \le 60, 12 \le y_4 \le 60$$

$$
\begin{aligned}
&\vec{y} = \{y_1, y_2, y_3, y_4, y_5\} \\
&\text{Minimze :} \\
&f(\vec{x}) = \pi(y_2^2 - y_1^2) y_3 (y_5 + 1)\rho \\
&\text{Subject to :} \\
&g_1(\vec{x}) = -p_{\max} + p_{rz} \leq 0 \\
&g_2(\vec{x}) = p_{rz} V_{sr} - V_{sr,\max} p_{\max} \leq 0 \\
&g_3(\vec{x}) = \Delta R + y_1 - y_2 \leq 0 \\
&g_4(\vec{x}) = -L_{\max} + (y_5 + 1)(y_3 + \delta) \leq 0 \\
&g_5(\vec{x}) = sM_s - M_h \leq 0 \\
&g_6(\vec{x}) = T \geq 0 \\
&g_7(\vec{x}) = -V_{sr,\max} + V_{sr} \leq 0 \\
&g_7(\vec{x}) = T - T_{\max} \leq 0 \\
&\text{where :} \\
&60 \leq y_1 \leq 80 \\
&90 \leq y_2 \leq 110 \\
&1 \leq y_3 \leq 3 \\
&0 \leq y_4 \leq 1000 \\
&2 \leq y_5 \leq 9 \\
&M_h = \frac{2}{3}\mu y_4 y_5 \frac{y_2^3 - y_1^3}{y_2^2 - y_1^2}\text{N.mm} \\
&\omega = \frac{\pi n}{30}\text{rad/s} \\
&A = \pi(y_2^2 - y_1^2)\text{mm}^2 \\
&p_{rz} = \frac{y_4}{A}\text{N/mm}^2 \\
&V_{sr} = \frac{\pi R_{sr} n}{30}\text{mm/s} \\
&R_{sr} = \frac{2}{3}\frac{y_2^3 - y_1^3}{y_2^2 y_1^2}\text{mm} \\
&T = \frac{I_z \omega}{M_h + M_f} \\
&\Delta R = 20\text{mm}, L_{\max} = 30\text{mm}, \mu = 0.6 \\
&V_{sr,\max} = 10\text{m/s}, \delta = 0.5\text{mm}, s = 1.5 \\
&T_{\max} = 15\text{s}, n = 250\text{rpm}, I_z = 55\text{Kg} \cdot \text{m}^2 \\
&M_s = 40\text{Nm}, M_f = 3\text{Nm}, \rho = 0.0000078, p_{\max} = 1
\end{aligned}
\tag{A3}
$$

References

1. Wu, G. Across neighborhood search for numerical optimization. *Inf. Sci.* **2016**, *329*, 597–618. [CrossRef]
2. Tanyildizi, E.; Demir, G. Golden Sine Algorithm: A Novel Math-Inspired Algorithm. *Adv. Electr. Comput. Eng.* **2017**, *17*, 71–78. [CrossRef]
3. Sun, X.X.; Pan, J.S.; Chu, S.C.; Hu, P.; Tian, A.Q. A novel pigeon-inspired optimization with QUasi-Affine TRansformationevolutionary algorithm for DV-Hop in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1–15. [CrossRef]
4. Song, P.C.; Chu, S.C.; Pan, J.S.; Yang, H. Simplified Phasmatodea population evolution algorithm for optimization. *Complex Intell. Syst.* **2022**, *8*, 2749–2767. [CrossRef]
5. Sun, L.; Koopialipoor, M.; Armaghani, D.J.; Tarinejad, R.; Tahir, M. Applying a meta-heuristic algorithm to predict and op-timize compressive strength of concrete samples. *Eng. Comput.* **2019**, *37*, 1133–1145. [CrossRef]
6. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
7. 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]
8. 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. [CrossRef]
9. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl.-Based Syst.* **2019**, *191*, 105190. [CrossRef]
10. Nematollahi, A.F.; Rahiminejad, A.; Vahidi, B. A novel physical based meta-heuristic optimization method known as lightning attachment procedure optimization. *Appl. Soft Comput.* **2017**, *59*, 596–621. [CrossRef]

11. Ghasemi, M.; Davoudkhani, I.F.; Akbari, E.; Rahimnejad, A.; Ghavidel, S.; Li, L. A novel and effective optimization algorithm for global optimization and its engineering applications: Turbulent Flow of Water-based Optimization (TFWO). *Eng. Appl. Artif. Intell.* **2020**, *92*, 103666. [CrossRef]
12. Malakar, S.; Ghosh, M.; Bhowmik, S.; Sarkar, R.; Nasipuri, M. A GA based hierarchical feature selection approach for handwritten word recognition. *Neural Comput. Appl.* **2019**, *32*, 2533–2552. [CrossRef]
13. Wang, L.; Pan, J.; Jiao, L. The immune algorithm. *Acta Electonica Sin.* **2000**, *28*, 96.
14. Valayapalayam Kittusamy, S.R.; Elhoseny, M.; Kathiresan, S. An enhanced whale optimization algorithm for vehicular communication networks. *Int. J. Commun. Syst.* **2019**, *35*, e3953. [CrossRef]
15. Rather, S.A.; Bala, P.S. Constriction coefficient based particle swarm optimization and gravitational search algorithm for multilevel image thresholding. *Expert Syst.* **2021**, *38*, e12717. [CrossRef]
16. Yan, Z.; Zhang, J.; Yang, Z.; Tang, J. Kapur's Entropy for Underwater Multilevel Thresholding Image Segmentation Based on Whale Optimization Algorithm. *IEEE Access* **2020**, *9*, 41294–41319. [CrossRef]
17. Navarro-Acosta, J.A.; García-Calvillo, I.D.; Reséndiz-Flores, E.O. Fault detection based on squirrel search algorithm and support vector data description for industrial processes. *Soft Comput.* **2022**, 1–12. [CrossRef]
18. Cao, X.; Yang, Z.Y.; Hong, W.C.; Xu, R.Z.; Wang, Y.T. Optimizing Berth-quay Crane Allocation considering Economic Factors Using Cha-otic Quantum SSA. *Appl. Artif. Intell.* **2022**, *36*, 2073719. [CrossRef]
19. Samieiyan, B.; MohammadiNasab, P.; Mollaei, M.A.; Hajizadeh, F.; Kangavari, M. Novel optimized crow search algorithm for feature selection. *Expert Syst. Appl.* **2022**, *204*, 117486. [CrossRef]
20. Phung, M.D.; Ha, Q.P. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. *Appl. Soft Comput.* **2021**, *107*, 107376. [CrossRef]
21. Trojovský, P.; Dehghani, M. Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. *Sensors* **2022**, *22*, 855. [CrossRef] [PubMed]
22. Yuan, Y.; Mu, X.; Shao, X.; Ren, J.; Zhao, Y.; Wang, Z. Optimization of an auto drum fashioned brake using the elite opposition-based learning and chaotic k-best gravitational search strategy based grey wolf optimizer algorithm. *Appl. Soft Comput.* **2022**, *123*, 108947. [CrossRef]
23. Kamboj, V.K.; Nandi, A.; Bhadoria, A.; Sehgal, S. An intensify Harris Hawks optimizer for numerical and engineering optimization problems. *Appl. Soft Comput.* **2020**, *89*, 106018. [CrossRef]
24. Rahati, A.; Rigi, E.M.; Idoumghar, L.; Brévilliers, M. Ensembles strategies for backtracking search algorithm with application to engi-neering design optimization problems. *Appl. Soft Comput.* **2022**, *121*, 108717. [CrossRef]
25. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
26. Alkayem, N.F.; Shen, L.; Asteris, P.G.; Sokol, M.; Xin, Z.; Cao, M. A new self-adaptive quasi-oppositional stochastic fractal search for the inverse problem of structural damage assessment. *Alex. Eng. J.* **2022**, *61*, 1922–1936. [CrossRef]
27. Tian, D.; Shi, Z. MPSO: Modified particle swarm optimization and its applications. *Swarm Evol. Comput.* **2018**, *41*, 49–68. [CrossRef]
28. Dhargupta, S.; Ghosh, M.; Mirjalili, S.; Sarkar, R. Selective Opposition based Grey Wolf Optimization. *Expert Syst. Appl.* **2020**, *151*, 113389. [CrossRef]
29. Alkayem, N.F.; Cao, M.; Shen, L.; Fu, R.; Šumarac, D. The combined social engineering particle swarm optimization for real-world engi-neering problems: A case study of model-based structural health monitoring. *Appl. Soft Comput.* **2022**, *123*, 108919. [CrossRef]
30. Wei, J.; Huang, H.; Yao, L.; Hu, Y.; Fan, Q.; Huang, D. New imbalanced fault diagnosis framework based on Cluster-MWMOTE and MFO-optimized LS-SVM using limited and complex bearing data. *Eng. Appl. Artif. Intell.* **2020**, *96*, 103966. [CrossRef]
31. Fan, Q.; Huang, H.; Li, Y.; Han, Z.; Hu, Y.; Huang, D. Beetle antenna strategy based grey wolf optimization. *Expert Syst. Appl.* **2020**, *165*, 113882. [CrossRef]
32. Fan, Q.; Huang, H.; Yang, K.; Zhang, S.; Yao, L.; Xiong, Q. A modified equilibrium optimizer using opposition-based learning and novel update rules. *Expert Syst. Appl.* **2021**, *170*, 114575. [CrossRef]
33. Fan, Q.; Huang, H.; Chen, Q.; Yao, L.; Yang, K.; Huang, D. A modified self-adaptive marine predators algorithm: Framework and engineering ap-plications. *Eng. Comput.* **2021**, *38*, 3269–3294. [CrossRef]
34. Wei, J.; Huang, H.; Yao, L.; Hu, Y.; Fan, Q.; Huang, D. NI-MWMOTE: An improving noise-immunity majority weighted minority oversampling technique for imbalanced classification problems. *Expert Syst. Appl.* **2020**, *158*, 113504. [CrossRef]
35. Chopra, N.; Ansari, M.M. Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. *Expert Syst. Appl.* **2022**, *198*, 116924. [CrossRef]
36. Long, W.; Jiao, J.; Xu, M.; Tang, M.; Wu, T.; Cai, S. Lens-imaging learning Harris hawks optimizer for global optimization and its application to feature selection. *Expert Syst. Appl.* **2022**, *202*, 117255. [CrossRef]
37. Tizhoosh H, R. 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), IEEE, Vienna, Austria, 28–30 November 2005; Volume 1, pp. 695–701.
38. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
39. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [CrossRef]

40. Hashim, F.A.; Hussien, A.G. Snake Optimizer: A novel meta-heuristic optimization algorithm. *Knowl. Based Syst.* **2022**, *242*, 108320. [CrossRef]
41. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–35. [CrossRef]
42. Maesono, Y. Competitors of the Wilcoxon signed rank test. *Ann. Inst. Stat. Math.* **1987**, *39*, 363–375. [CrossRef]
43. Theodorsson-Norheim, E. Friedman and Quade tests: BASIC computer program to perform nonparametric two-way analysis of variance and multiple comparisons on ranks of several related samples. *Comput. Biol. Med.* **1987**, *17*, 85–99. [CrossRef]
44. Gandomi, A.; Alavi, A.H. An introduction of krill herd algorithm for engineering optimization. *J. Civ. Eng. Manag.* **2015**, *22*, 302–310. [CrossRef]
45. Sandgren, E. Nonlinear Integer and Discrete Programming in Mechanical Design Optimization. *J. Mech. Des.* **1990**, *112*, 223–229. [CrossRef]
46. Osyczka, A.; Osyczka, A. *Evolutionary Algorithms for Single and Multicriteria Design Optimization*; Physica-Verlag: Heidelberg, Germany, 2002.
47. Yang, X.S. *Nature-Inspired Metaheuristic Algorithms*; Luniver Press: Frome, UK, 2010.
48. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
49. Bertsimas, D.; Tsitsiklis, J. Simulated annealing. *Stat. Sci.* **1993**, *8*, 10–15. [CrossRef]
50. Yang, X.S. Flower pollination algorithm for global optimization. In *International Conference on Unconventional Computing and Natural Computation*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 240–249.
51. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [CrossRef]
52. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [CrossRef]
53. Połap, D.; Woz´niak, M. Polar Bear Optimization Algorithm: Meta-Heuristic with Fast Population Movement and Dynamic Birth and Death Mechanism. *Symmetry* **2017**, *9*, 203. [CrossRef]
54. Yang, X.S. Firefly algorithms for multimodal optimization. In *International Symposium on Stochastic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 169–178.
55. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
56. 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]
57. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
58. Sörensen, K. Metaheuristics-the metaphor exposed. *Int. Trans. Oper. Res.* **2013**, *22*, 3–18. [CrossRef]