entropy

MDPI

Article

Improved Gravitational Search Algorithm Based on Adaptive Strategies

Zhonghua Yang [1,2], Yuanli Cai [2,*] and Ge Li [1]

1 College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
2 Faculty of Electronics & Information, Xi'an Jiaotong University, Xi'an 710049, China
* Correspondence: ylicai@mail.xjtu.edu.cn

Abstract: The gravitational search algorithm is a global optimization algorithm that has the advantages of a swarm intelligence algorithm. Compared with traditional algorithms, the performance in terms of global search and convergence is relatively good, but the solution is not always accurate, and the algorithm has difficulty jumping out of locally optimal solutions. In view of these shortcomings, an improved gravitational search algorithm based on an adaptive strategy is proposed. The algorithm uses the adaptive strategy to improve the updating methods for the distance between particles, gravitational constant, and position in the gravitational search model. This strengthens the information interaction between particles in the group and improves the exploration and exploitation capacity of the algorithm. In this paper, 13 classical single-peak and multi-peak test functions were selected for simulation performance tests, and the CEC2017 benchmark function was used for a comparison test. The test results show that the improved gravitational search algorithm can address the tendency of the original algorithm to fall into local extrema and significantly improve both the solution accuracy and the ability to find the globally optimal solution.

Keywords: gravitational search algorithm; swarm intelligence algorithm; adaptive strategy; particle information interaction

Citation: Yang, Z.; Cai, Y.; Li, G. Improved Gravitational Search Algorithm Based on Adaptive Strategies. *Entropy* **2022**, *24*, 1826. https://doi.org/10.3390/e24121826

Academic Editor: Shu-Chuan Chu

Received: 9 November 2022
Accepted: 13 December 2022
Published: 14 December 2022

1. Introduction

With the progress of science and technology as well as the development of production and management, optimization problems cover almost all aspects of human life and production, becoming an important theoretical basis and indispensable method of modern science. The main solutions to optimization problems include traditional optimization methods and modern optimization methods. Traditional optimization methods are based on single-point optimization, and the main approaches are enumeration methods, numerical methods, and analytical methods. Modern optimization methods use swarm intelligence algorithms, inspired by the stimulation of biological evolution, that simulate the structural characteristics, evolutionary laws, thinking structures, and behavior patterns of human, natural, and other biological populations. Typical swarm intelligence algorithms include the evolutionary algorithm, artificial immune algorithm, memory algorithm, particle swarm optimization, shuffled frog leaping algorithm, cat swarm optimization, bacterial foraging optimization, artificial fish school algorithm, ant colony algorithm, and artificial bee colony algorithm. Traditional optimization methods have strict requirements for the optimization problems in practical projects, and the calculation speed is slow and the convergence is poor when solving large-scale complex problems. Often, the solution to the problem cannot be found in an acceptable time. Modern optimization methods have loose requirements for solving problems, and have good adaptability, robustness, and global search ability.

The gravitational search algorithm (GSA) was proposed by Raahedi et al. in 2009. It is a swarm intelligence global optimization algorithm that is simple to implement and achieves relatively good performance in global search and convergence. It is widely used in path planning [1,2], image classification [3–5], neural networks [6,7], data prediction [8–10], scheduling and parameter estimation [11–17], and other fields. However, the solution accuracy is not high, and the algorithm finds it difficult to jump out of locally optimal solutions in the later stages. To solve the shortcomings of this algorithm, scholars have improved the GSA algorithm with respect to the following three aspects: improvements to the adaptive strategy, integration with other swarm intelligence algorithms, and the introduction of other improvement strategies.

To address the first aspect, the authors of [18] proposed an adaptive GSA called SGSA in which an exponential decay model was introduced to the gravitation constant so that the algorithm can adjust the relevant parameters as required by the algorithm as the iteration proceeds. This also improves the exploration ability of the algorithm. An adaptive strategy based on population density was proposed for the distance between particles to prevent the algorithm from degenerating into random motion and to accelerate the convergence speed. The authors of [19] designed a new dynamic inertial weight and velocity position trend factor to improve the GSA, so that the inertial mass of particles has a certain trend as the iterations progress. This gives the change in position of each particle randomness and stability and gives the algorithm a certain degree of adaptability.

To address the second aspect, the authors of [20] combined the GSA with an immune algorithm, which introduces antibody diversity and immune memory characteristics into GSA and improves its global search ability. To overcome the problems of slow iterations and tendency to fall into local minima during the optimization of the standard GSA, one study [21] introduced the speed update mechanism of particle swarm optimization into the position update of GSA, combining the exploitation ability of particle swarm optimization and the exploration ability of GSA, and effectively solving the abovementioned problems. Another study [22] combined the free search differential evolution algorithm with the GSA to make full use of the exploration ability of GSA and the exploitation ability of the free search differential evolution algorithm, and to avoid the premature convergence of GSA. The authors of [23] combined the GSA with the sperm swarm optimization algorithm, which combines the advantages of both algorithms. Through testing, the hybrid method was found to have a better ability to avoid local extrema, and its convergence speed is relatively fast.

Finally, several studies have addressed the third aspect. The authors of [24] introduced a mutation operator to GSA and performed the mutation operation on particles with poor fitness values in the population. The particles were reinitialized, effectively preventing the algorithm from falling into locally optimal values, and the improved algorithm was successfully applied to an economic load scheduling problem. Another study [25] proposed the rotation GSA, which optimizes the selection of the k best in GSA by introducing a rotation operator so that unincluded particles have the opportunity to affect the motion of other particles and to enhance the exploration ability of the algorithm. The study [26] proposed a GSA based on Levy flight and chaos theory. The Levy distribution was used to improve the diversity of the population search space, and chaos search was used to strengthen candidate solutions to achieve global optimization. The authors of [27] proposed an improved GSA based on mutation strategy and reverse evaluation mechanism. The reverse learning bidirectional evaluation mechanism proposed by Tizhoosh was used to initialize and update the population so that particles were better distributed. In addition, the best individual and particles with poor fitness were cross-mutated using a mutation strategy to avoid premature convergence.

In summary, the key to improving the performance of the GSA is to balance the diversity and convergence of particles and prevent the algorithm from falling into local extreme values too early. Among the above three aspects of improvement, the adaptive strategy has a better effect and obtains the best performance. However, scholars have only used two adaptive strategies at most to improve the algorithm performance. Hence, there is room for further improvements in the algorithm's performance. In this paper, a new adaptive GSA is proposed in which three adaptive strategies for population density, gravitation constant, and location update are used in combination to improve the optimization accuracy and convergence of the GSA. The organizational structure of this paper is as follows: Section 2 outlines the basic GSA, Section 3 introduces the three adaptive improvement strategies, Section 4 describes the idea and steps of the improved GSA and analyzes the space–time complexity and convergence performance. Finally, Section 5 evaluates the performance of the improved GSA through experiments, and Section 6 summarizes the conclusions.

2. Basic GSA

The universal GSA treats all particles as objects with mass. During the optimization process, all particles move unimpeded. Each particle is affected by the gravity of the other particles in the solution space and generates acceleration to move toward the particles with greater mass. Because the mass of the particles is related to their fitness, particles with large fitness will have greater mass. Therefore, particles with small masses will gradually approach the optimal solution in the optimization problem in the process of approaching particles with large masses. The GSA is different from other swarm intelligence algorithms. In the GSA, particles do not need to perceive the environment through environmental factors but realize information sharing through the interaction of the gravitational forces between individuals. Therefore, without the influence of environmental factors, particles can also perceive the global situation to conduct a global search in the environment, thus realizing the global optimization of the problem.

In a GSA, we assume that a D-dimensional search space contains N objects, and the position of the i-th object is

$$\mathrm{X}_i = (x_i^1, x_i^2, x_i^3, \ldots, x_i^k \ldots x_i^{\mathrm{D}}), i = 1, 2, \ldots, N \tag{1}$$

In Equation (1), x_i^k represents the position of the i-th object in the k-th dimension.

2.1. Inertial Mass Calculation

In the GSA, the inertial mass of each particle is directly related to the fitness value obtained from the particle location. At time t, the mass of particle X_i is expressed by $M_i(t)$. Because the inertial mass M is calculated according to its corresponding fitness value, the particles with larger M values are closer to the optimal solution in the solution space, and they exert a greater attraction on other objects.

Particle mass $M_i(t)$ is calculated according to

$$\begin{cases} m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \\ M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \end{cases} \tag{2}$$

Here, $fit_i(t)$ represents the fitness of particle X_i, $best(t)$ represents the best solution at time t, $worst(t)$ represents the worst solution at time t, and the calculation is as follows:

$$\begin{cases} best(t) = \max\limits_{i \in \{1,2,\ldots,N\}} fit(t) \\ worst(t) = \min\limits_{i \in \{1,2,\ldots,N\}} fit(t) \end{cases} \tag{3}$$

It can be seen from Equation (2) that $m_i(t)$ normalizes the fitness of particles to the range [0, 1] and then takes its proportion in the total mass as the mass $M_i(t)$ of the particles.

2.2. Gravitational Calculation

At time t, the calculation of the gravitational force of object j subjected to object i in the k-th dimension is as follows:

$$F_{ij}{}^{k}(t) = G(t)\frac{M_{ai}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}(x_j^k(t) - x_i^k(t)), \quad (4)$$

where ε represents a very small constant, $M_{aj}(t)$ represents the inertial mass of the action object j, and $M_{ai}(t)$ represents the inertial mass of the action object i. Furthermore, $G(t)$ represents the constant of universal gravitation transformed over time, where its size is related to the number of iterations, and its calculation is

$$G(t) = G_0 \times e^{-\alpha t/T}. \quad (5)$$

In Equation (5), G_0 represents the value of G at time t_0, where G_0 = 100, α = 20, and T is the maximum number of iterations. Finally, $R_{ij}(t)$ represents the Euclidean distance between objects X_i and X_j, and is calculated as

$$R_{ij}(t) = ||X_i(t) - X_j(t)||_2. \quad (6)$$

At time t, the force acting on X_i in the k-th dimension is equal to the sum of the forces exerted on it by all other particles around as follows:

$$F_i^k(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^k(t). \quad (7)$$

2.3. Location Update

When a particle is subjected to the gravitational action of other particles, it will generate acceleration. According to the gravity calculated in Equation (7), the acceleration obtained by object i in the k-th dimension is the ratio of its force to inertial mass. The calculation is as follows:

$$\alpha_i^k(t) = \frac{F_i^k(t)}{M_i(t)}. \quad (8)$$

In each iteration, the algorithm updates the speed and position of object i according to the calculated acceleration. The update method is

$$v_i^k(t+1) = rand_i \times v_i^k + \alpha_i^k(t), \quad (9)$$

$$x_i^k(t+1) = x_i^k(t) + v_i^k(t+1). \quad (10)$$

The basic GSA implementation steps are as follows:

1. Initialize the position and acceleration of all particles in the algorithm, and set the number of iterations and parameters.
2. Calculate the fitness value for each particle, and update the gravitation constant according to the formula.
3. The mass of each particle is calculated according to the calculated fitness value, and the acceleration of each particle is calculated using Equations (2)–(8).
4. Calculate the speed of each particle according to Equation (9) and then update the particle position according to Equation (10).
5. If the termination condition is not met, return to step 2; otherwise, output the optimal solution of the algorithm.

3. Adaptive Strategies

3.1. Adaptive Population Density Strategy

The distance between particles in the basic GSA is the Euclidean distance. Through a large number of experiments in [18], it was found that a constant fixed distance is better than the Euclidean distance, but the fixed distance value has obvious shortcomings: First, when the population is divided, the distance between particles is large and the interaction force is very small, and hence, the GSA degenerates into random motion. Second, when the population is dense, the distance between particles is very small, and the interaction force is very large. The particles in the algorithm will oscillate at a high frequency near the optimal solution and reduce the convergence speed. The population density is an indicator for evaluating the distance between particles, and it is the median of the average distance of all particles in the population. A smaller population density means the population is more concentrated; by contrast, a higher population density means the population is more dispersed. To solve the above two issues, balance the exploration and exploitation abilities of the algorithm, adjust the search ability of the algorithm, and propose an adaptive strategy based on population density, we dynamically adjust the distance between particles according to the GSA population density. That is, when the population density is relatively large, the population is relatively dispersed, reducing the particle distance between populations, promoting information exchange between particles, and preventing random movement between particles. When the population density value is small, the population is dense. We hence increase the distance between population particles appropriately to speed up the convergence of the algorithm. The calculation of population density δ is as follows:

$$\delta = \frac{1}{\mathrm{N}}\sum_{i=1}^{N} dis_i, \tag{11}$$

where N is the number of particles, D is the dimensionality of the particles, and dis_i is the average distance between the i-th particle and all other particles, calculated as follows:

$$\mathrm{dis}_i = \frac{1}{\mathrm{N}-1}\sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2}. \tag{12}$$

The gravitational force calculated in the basic universal GSA is modified as follows:

$$F_{ij}{}^{k}(t) = G(t)\frac{M_{a\mathrm{i}}(t) \times M_{aj}(t)}{R_{ij}{}^{\mathrm{Rp}(\delta)}(t) + \varepsilon}(x_j^k(t) - x_i^k(t)). \tag{13}$$

The calculation of $\mathrm{Rp}(\delta)$ is

$$\mathrm{Rp}(\delta) = \begin{cases} Rp_{\min} + (Rp_{\max} - Rp_{\min})e^{1-1/\delta} & \delta < 1 \\ Rp_{\min} + (\mathrm{Rp}_{\max} - Rp_{\min})e^{1-\delta} & \delta \geq 1 \end{cases}, \tag{14}$$

where $Rp_{\max}$ and $Rp_{\min}$ are the maximum and minimum values of the given fixed distance, respectively, and δ is the population density.

3.2. Adaptive Gravitational Constant Strategy

Gravitational constant G is an important variable that transforms over time. Its change directly affects the magnitude of resultant force and acceleration, as well as determines the current step size and convergence speed of particles in the algorithm. The reasonable selection of parameters G_0 and α plays an important role in the size of the iterative steps in the algorithm and determines whether the algorithm can jump out of local optima and determine the direct factor of optimal accuracy. If the original gravitational constant decreases quickly at the beginning of the algorithm, the algorithm can converge quickly, but it also tends to fall into the local optima and is difficult to jump out. To improve the exploration ability of the algorithm, prevent the algorithm from falling into locally optimal solutions, and improve the accuracy of the solution, an adaptive strategy for the universal gravitational constant is proposed. The adaptive gravitational constant G is expressed as follows:

$$G(t) = \frac{G_0}{1+\mathrm{e}^{\alpha(t-t_c)/T}} \quad 0 \leq t_c < T \tag{15}$$

Here, G_0 is the initial value of the universal gravitational constant, α is the parameter of the decay rate, T is the total number of iterations, and t_c is a constant value in the interval $[0, T)$.

3.3. Adaptive Location Update Strategy

A position in the basic GSA is updated according to the current speed of the algorithm and the position in the last iteration. In each iteration, if the current update speed of particles is small, the change in position will also be small, the convergence ability of the algorithm will be reduced, and the algorithm will tend to fall into local extrema. By contrast, if the current update speed of particles is too large, the change in position will also increase, and the algorithm will move far from the global optimum. To address these defects, the improved strategy in [3] is adopted in this study to make the position of the particles change with respect to the iterative evolution. In the early stages of the algorithm, each particle moves with a large step size so that the algorithm can quickly converge to the vicinity of the optimal solution; in the later stages of the algorithm, the particle update step is smaller, and the particle depth search is near the optimal value. The expression of the adaptive position update is

$$x_i^k(t+1) = \alpha \times x_i^k(t) + \beta \times v_i^k(t+1), \tag{16}$$

where α is calculated by

$$\alpha = e^{(-\dim*(t/\mathrm{T}_{\max})^{\omega})} \tag{17}$$

and β is calculated by

$$\beta = 1 - \frac{t}{T_{\max} + betarnd}, \tag{18}$$

where dim is the dimension; ω is an integer in the range [1, 50]; T is the current number of iterations of the algorithm; $T_{\max}$ is the maximum number of iterations set for the algorithm; and *betarnd* is the random number generated by the [0, 1] beta distribution. The range of α is (0, 1) and the range of β is (0, 1).

4. Improved GSA Based on Adaptive Strategies

4.1. Basic Concept

To improve the low solution accuracy and difficulty of jumping out of the locally optimal solutions of conventional GSA, the parameters for the distance between particles, gravitational constant, and position update in GSA are improved using adaptive strategies to strengthen the information interaction between particles and improve the exploration and exploitation capabilities of the algorithm.

The steps of the proposed algorithm are as follows.

Step 1: The proposed adaptive GSA is initialized to generate the initial particle swarm. Set the size of algorithm particle swarm N and the maximum number of iterations NC_{Max}, search space dimension X_{Dim}, maximum distance $Rp_{\max}$, minimum distance $Rp_{\min}$, gravitational constant, attenuation rate, constant value, and other parameter values.
Step 2: Check the particle boundary in the population, and calculate the fitness value of particles in the population.
Step 3: Use Equation (3) to calculate *best*(t) and *worst*(t).

Step 4: The inertial mass M_i(t) of the particles is obtained according to *best*(t), *worst*(t), and Equation (2).
Step 5: Update the gravitational constant G according to Equation (15).
Step 6: Calculate the distance between particles according to Equations (6), (11), and (14).
Step 7: Calculate the gravitational and resultant forces around particles according to Equation (13).
Step 8: Calculate the acceleration of the particles according to Equation (8).
Step 9: Update particle speeds and positions according to Equations (9) and (16)–(18).
Step 10: Return to the iteration cycle in step 2 until the number of cycles or accuracy requirements are met.
Step 11: Exit the loop and output the algorithm results.

4.2. Temporal and Spatial Complexity Analyses

4.2.1. Time Complexity Analysis

The time complexity of the algorithm is the time spent executing the algorithm, which is equal to the cumulative number of times the algorithm performs basic operations such as addition, subtraction, multiplication, division, and comparison. Assuming that the particle swarm size of the proposed adaptive GSA is N, the time complexity of the algorithm is analyzed according to the steps of the algorithm execution using the method in [28].

In step 1, the initialization of the particle swarm of the proposed adaptive GSA requires N operations, and the initialization operations of the other parameters are a constant; thus, the time complexity of step 1 is O (N).

In step 2, the particle boundary check requires N operations, the fitness calculation requires N operations, and hence, the time complexity of step 2 is O(N) + O(N).

In step 3, it takes one operation to calculate the best fitness value *best*(t) and one operation to calculate the worst fitness value *worst*(t), and hence, the time complexity of step 3 is O(1) + O(1).

Calculating the inertial mass $M_i(t)$ of particles in step 4 requires N operations; thus, the time complexity of step 4 is O(N).

Updating the gravitational constant G in step 5 requires one operation; thus, the time complexity of step 5 is O(1).

In step 6, calculating the average distance of all particles requires $N \times (N - 1)$ operations, calculating the population density requires one operation, and calculating the particle distance requires one operation, and hence, the time complexity of step 6 is $O(N \times (N - 1)) + O(1) + O(1)$.

In step 7, calculating the gravity of particles in the population requires N operations, and calculating the resultant force of particles in the population requires N operations; thus, the time complexity of step 7 is O(N) + O(N).

In step 8, calculating particle acceleration requires N operations, and calculating particle velocity requires N operations, and hence, the time complexity of step 8 is O(N) + O(N).

Updating particle positions in step 9 requires N operations; thus, the time complexity of step 9 is O(N).

In step 10, evaluating the termination condition requires one comparison operation, and terminating the algorithm requires one assignment operation; thus, the time complexity of step 10 is O(1) + O(1).

After the above steps, the proposed adaptive GSA performs $NC_{\max}$ iterations. The time complexity the proposed adaptive GSA after the maximum number of iterations is $O(NC_{\max} \times (N \times N))$.

4.2.2. Spatial Complexity Analysis

Space complexity is a measure of the storage space occupied by the algorithm during execution. Assume that the population size of the proposed adaptive GSA is N, the number of iterations of the algorithm is $NC_{\max}$, and the dimensionality of the optimization function is D. We perform spatial complexity analysis according to the steps of algorithm execution. In the proposed adaptive GSA, X [N] [D] is used to store the value of initialization independent variable, Y [1] [N] is used to store the fitness value of initialization function, Xm [1] [N] is used to store the inertial mass value of population particles, Xd [1] [N] is used to store the average distance between population particles, Xf [N] [D] is used to store the resultant force value around each particle in the population, Xa [N] [D] is used to store the acceleration value of each particle in the population, and Xv [N] [D] is used to store the velocity value of each particle in the population. Therefore, the space complexity of the whole GSA based on adaptive strategy improvement is 4 × O(N × D).

4.2.3. Analysis of Algorithm Convergence

The convergence of the proposed adaptive GSA is proven using the contraction mapping theorem. For the relevant concepts and theorems in space compression theory, we refer readers to the definitions in [28].

Theorem 1. *As the time tends to infinity, the proposed adaptive GSA is convergent.*

Proof: The state of the proposed adaptive GSA in the optimization process is represented by set X. The mutual transformation of states in set X is actually the embodiment of the whole optimization process of the proposed adaptive GSA. Therefore, the optimization process of the proposed adaptive GSA is a self-mapping process. If f is an optimization process mapping from X to X, then $X_{k+1} = f(X_k)$. Suppose $\exists \rho : X \times X \to \mathrm{R}$ is the distance between two points in metric space (X, ρ) and $\{x_n\}$ is any optimization sequence in (X, ρ). □

The proposed adaptive GSA is a continuous iterative process. Under the action of gravity, individuals in the algorithm attract each other, forcing small mass individuals to constantly move to larger mass individuals to determine the optimal solution X^*. Therefore, in metric space (X, ρ), for any ε, there exists an N, where n > N, such that $\rho(x_n, \mathrm{X}^*) < \varepsilon$ is true. According to its definition, the optimization sequence $\{x_n\}$ converges to X^*. Moreover, $\{x_n\}$ is a Cauchy sequence, and (X, ρ) is a complete metric space. Let ε be a random number in the range [0, 1]. Since the proposed adaptive GSA is a continuous optimization process, individuals are constantly approaching the optimal value, and hence, it is a convergent process. Then, there must be $\rho(f(x), f(y) \leq \varepsilon * \rho(x, y)$ in the metric space (X, ρ), and f is a compression mapping. According to Theorem 4.2 in [28], $\mathrm{x}^* = \lim_{k \to \infty} f^*(x_0)$ is true, where $x^* \in X$ is the only fixed point in the compressed mapping f. Hence, the proposed adaptive GSA is convergent, and Theorem 1 is proven.

5. Experimental Analysis

5.1. Performance of the Algorithm Improvement Strategies

This section evaluates and analyzes the combinations of the three adaptive improvement strategies in the algorithm: the adaptive population density strategy, adaptive gravitational constant strategy, and adaptive location update strategy. For the convenience of description, we refer to the GSA based on the adaptive population density strategy as the RGSA. Similarly, we call the GSA based on the adaptive gravitational constant strategy the GGSA, and the GSA based on the adaptive position strategy is called the LGSA. The GSA based on the adaptive population density and gravity constant strategies is the RGGSA, the GSA based on the adaptive population density and location update strategies is the RLGSA, the GSA based on the adaptive gravity constant and position update strategies is the GLGSA, and the GSA based on all three adaptive population density, gravity constant, and location update strategies is the RGLGSA. In the experiment, we tested the convergence of the basic GSA and the GSA with the seven different adaptive strategies on benchmark functions f1 to f15.

5.1.1. Test Functions

Table 1 lists the test functions. Among the 15 standard test functions, the minimum value 0.397887 is obtained at points (π, 12.275), (π, 2.275), and (9.42478, 2.475), and the other functions obtain the minimum value 0 at point (0, 0, ..., 0). The functions f1, f2, f3, f7, f8, f10, and f13 are unimodal test functions. These functions only have one globally optimal solution, and are mainly used to test the solution accuracy and development ability of the algorithm. Functions f4, f5, f6, f9, f11, f12, f14, and f15 are multimodal test functions. These functions have many local extrema. The GSA is prone to premature convergence or falling into local extrema. To determine the optimal values for these test functions, the algorithm must have the ability to jump out of local extrema, avoid premature convergence, and have strong global exploration ability.

Table 1. Test function set.

Function No.	Test Function	Value Range
1	$f_1(x) = \sum\limits_{i=1}^{n} x_i^2$	$x_i \in [-100, 100]$
2	$f_2(x) = \sum\limits_{i=1}^{n} \lvert x_i \rvert + \prod\limits_{i=1}^{n} \lvert x_i \rvert$	$x_i \in [-10, 10]$
	$f_3(x) = \max \lvert x_i \rvert$	$x_i \in [-100, 100]$
4	$f_4(x) = -20\exp[-0.2\sqrt{\frac{1}{n}\sum\limits_{i=1}^{n} x_i^2}] - \exp(\frac{1}{n}\sum\limits_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	$x_i \in [-32, 32]$
5	$f_5(x) = \sum\limits_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	$x_i \in [-10, 10]$
6	$f_6(x) = (x_2 - \frac{5.1}{4\pi} x_1^2 - \frac{5}{\pi} x_1 - 6)^2 + 10 \times (1 - \frac{1}{8\pi}) \cos x_1 + 10$	$x_i \in [-10, 10]$
7	$f_7(x) = \sum\limits_{i=1}^{n} (\sum\limits_{j=1}^{i} x_j)^2$	$x_i \in [-100, 100]$
8	$f_8(x) = \sum\limits_{i=1}^{n} i x_i^4 + random[0, 1]$	$x_i \in [-1.28, 1.28]$
9	$f_9(x) = \frac{1}{4000}\sum\limits_{i=1}^{n} x_i^2 - \prod\limits_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$x_i \in [-600, 600]$
10	$f_{10}(x) = \sum\limits_{i=1}^{n} {x_i}^2 + (\sum\limits_{i=1}^{n} \frac{i}{2} x_i)^2 + (\sum\limits_{i=1}^{n} \frac{i}{2} x_i)^4$	$x_i \in [-32, 32]$
11	$f_{11}(x) = x^2 + y^2 + 25(\sin^2 x + \sin^2 y)$	$x_i \in [-5.14, 5.14]$
12	$f_{12}(x) = \sum\limits_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$x_i \in [-5, 5]$
13	$f_{13}(x) = \sum\limits_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	$x_i \in [-100, 100]$
14	$f_{14}(x) = \cos(2\pi\sqrt{\sum\limits_{i=1}^{n} x_i^2}) + 0.1(\sqrt{\sum\limits_{i=1}^{n} x_i^2}) + 1$	$x_i \in [-100, 100]$
15	$f_{15}(x) = 0.5 + \frac{(\sin\sqrt{x_2^2 + x_1^2})^2 - 0.5}{(1 + 0.001(x_2^2 + x_1^2))^2}$	$x_i \in [-10, 10]$

5.1.2. Data Analysis

We set the initial parameters of the algorithms as follows: for the basic GSA, G_0 was 100 and α was 20; for RGSA: G_0 was 100, $\alpha^{\cdot}$ was 20, t_c was T/4, Rp_{max} was 1.5, and Rp_{min} was 0.5; for GGSA: G_0 was 50 and $\alpha^{\cdot}$ was 30; for SGSA: G_0 was 100, α was 20, and ω was 10. The parameter settings of the other GSAs were consistent with those of the previous three GSAs. The population size for all algorithms was 50, the number of algorithm iterations was 1000, the test dimensions were 30 and 50, functions f6 and f11 were two-dimensional tests, and the number of independent algorithm runs was 30.

The following observations can be inferred from the simulation test results of Tables 2–16.

Table 2. Test results for $f_1(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	3.1784×10^{-17}	1.1678×10^{-17}	2.0655×10^{-17}	4.9762×10^{-18}
	RGSA	5.7262×10^{-33}	4.1402×10^{-34}	1.0540×10^{-33}	1.0033×10^{-33}
	GGSA	7.8267×10^{-20}	1.6952×10^{-20}	4.2861×10^{-20}	1.5505×10^{-20}
	LGSA	1.1171×10^{-27}	7.0551×10^{-31}	1.0968×10^{-28}	2.3663×10^{-28}
	RGGSA	2.4777×10^{-38}	2.4197×10^{-39}	8.1763×10^{-39}	4.6307×10^{-39}
	RLGSA	4.3833×10^{-48}	4.9380×10^{-51}	3.9451×10^{-49}	9.0845×10^{-49}
	GLGSA	2.2316×10^{-29}	2.3715×10^{-34}	9.0465×10^{-31}	4.0589×10^{-30}
	RGLGSA	$\mathbf{1.1227 \times 10^{-53}}$	$\mathbf{1.3115 \times 10^{-56}}$	$\mathbf{1.8360 \times 10^{-54}}$	$\mathbf{3.3730 \times 10^{-54}}$
50	GSA	1.0370×10^{-16}	4.2637×10^{-17}	7.1477×10^{-17}	1.8306×10^{-17}
	RGSA	7.9678×10^{-32}	4.0968×10^{-33}	1.7587×10^{-32}	1.6047×10^{-32}
	GGSA	3.1420×10^{-19}	7.4754×10^{-20}	2.0370×10^{-19}	6.3958×10^{-20}
	LGSA	7.8808×10^{-28}	4.8414×10^{-31}	8.3115×10^{-29}	1.5517×10^{-28}
	RGGSA	1.2945×10^{-35}	1.7729×10^{-37}	1.5203×10^{-36}	2.7454×10^{-36}
	RLGSA	3.2260×10^{-48}	1.9879×10^{-50}	6.0257×10^{-49}	7.9558×10^{-49}
	GLGSA	2.6738×10^{-30}	2.0636×10^{-33}	2.3400×10^{-31}	5.0105×10^{-31}
	RGLGSA	$\mathbf{1.4061 \times 10^{-53}}$	$\mathbf{3.6091 \times 10^{-56}}$	$\mathbf{2.1920 \times 10^{-54}}$	$\mathbf{3.3828 \times 10^{-54}}$

Table 3. Test results for $f_2(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	2.9944×10^{-8}	1.8259×10^{-8}	2.3354×10^{-8}	2.8498×10^{-9}
	RGSA	3.7319×10^{-16}	1.1320×10^{-16}	2.2498×10^{-16}	6.5816×10^{-17}
	GGSA	1.5355×10^{-9}	6.4974×10^{-10}	1.0641×10^{-9}	2.3299×10^{-10}
	LGSA	5.3312×10^{-14}	4.3968×10^{-16}	1.2055×10^{-14}	1.4035×10^{-14}
	RGGSA	1.8821×10^{-17}	3.8605×10^{-19}	1.7182×10^{-18}	3.3370×10^{-18}
	RLGSA	5.1586×10^{-24}	1.0922×10^{-25}	8.5523×10^{-25}	9.6857×10^{-25}
	GLGSA	1.6789×10^{-15}	1.0556×10^{-16}	4.5455×10^{-16}	3.1838×10^{-16}
	RGLGSA	$\mathbf{4.8916 \times 10^{-27}}$	$\mathbf{2.0799 \times 10^{-28}}$	$\mathbf{1.8049 \times 10^{-27}}$	$\mathbf{1.2726 \times 10^{-27}}$
50	GSA	0.0236	3.7905×10^{-08}	7.8561×10^{-4}	0.0043
	RGSA	0.0861	6.2460×10^{-16}	0.0047	0.0168
	GGSA	0.2069	2.1929×10^{-9}	0.0181	0.0546
	LGSA	5.1145×10^{-14}	1.2501×10^{-15}	1.3527×10^{-14}	1.1082×10^{-14}
	RGGSA	0.2833	2.7805×10^{-13}	0.0320	0.0748
	RLGSA	3.1275×10^{-24}	1.8799×10^{-25}	1.2276×10^{-24}	8.5222×10^{-25}
	GLGSA	2.8762×10^{-15}	1.3163×10^{-16}	6.1703×10^{-16}	5.8851×10^{-16}
	RGLGSA	$\mathbf{8.7338 \times 10^{-27}}$	$\mathbf{1.9445 \times 10^{-28}}$	$\mathbf{2.2760 \times 10^{-27}}$	$\mathbf{1.8606 \times 10^{-27}}$

Table 4. Test results for $f_3(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	4.6203×10^{-9}	2.0095×10^{-9}	3.1109×10^{-9}	7.1216×10^{-10}
	RGSA	9.4873×10^{-16}	5.6256×10^{-17}	2.5335×10^{-16}	2.4418×10^{-16}
	GGSA	3.0792×10^{-10}	9.8951×10^{-11}	2.1632×10^{-10}	4.9367×10^{-11}
	LGSA	3.2172×10^{-14}	6.3082×10^{-16}	8.1911×10^{-15}	8.6301×10^{-15}
	RGGSA	0.0167	3.0764×10^{-18}	0.0011	0.0041
	RLGSA	4.0438×10^{-24}	2.3297×10^{-26}	7.0978×10^{-25}	8.7193×10^{-25}
	GLGSA	2.9153×10^{-15}	4.4903×10^{-17}	5.0494×10^{-16}	6.3364×10^{-16}
	RGLGSA	$\mathbf{9.4406 \times 10^{-27}}$	$\mathbf{2.0399 \times 10^{-28}}$	$\mathbf{1.9853 \times 10^{-27}}$	$\mathbf{1.9898 \times 10^{-27}}$

Table 4. *Cont.*

D	Algorithm	Worst	Best	Mean	SD
50	GSA	7.3060	1.4952	3.8958	1.2656
	RGSA	7.7420	1.4171	3.8093	1.6933
	GGSA	0.0065	1.8832×10^{-9}	2.4286×10^{-4}	0.0012
	LGSA	2.9909×10^{-14}	2.6038×10^{-16}	9.7994×10^{-15}	8.8800×10^{-15}
	RGGSA	1.2411	0.3102	0.6938	0.2702
	RLGSA	6.4042×10^{-24}	1.3099×10^{-25}	1.2454×10^{-24}	1.5236×10^{-24}
	GLGSA	5.2383×10^{-15}	2.7606×10^{-17}	7.0144×10^{-16}	1.0081×10^{-15}
	RGLGSA	$\mathbf{1.0648 \times 10^{-26}}$	$\mathbf{6.6210 \times 10^{-29}}$	$\mathbf{3.0313 \times 10^{-27}}$	$\mathbf{3.1224 \times 10^{-27}}$

Table 5. Test results for $f_4(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	4.5997×10^{-9}	2.5140×10^{-9}	3.5884×10^{-9}	4.2747×10^{-10}
	RGSA	1.5099×10^{-14}	7.9936×10^{-15}	1.0599×10^{-14}	2.7886×10^{-15}
	GGSA	2.5246×10^{-10}	1.2959×10^{-10}	1.6997×10^{-10}	2.7635×10^{-11}
	LGSA	9.3259×10^{-14}	$\mathbf{8.8818 \times 10^{-16}}$	1.1428×10^{-14}	1.7165×10^{-14}
	RGGSA	1.5099×10^{-14}	7.9936×10^{-15}	9.6515×10^{-15}	2.5945×10^{-15}
	RLGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**
	GLGSA	4.4409×10^{-15}	$\mathbf{8.8818 \times 10^{-16}}$	1.1250×10^{-15}	9.0135×10^{-16}
	RGLGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**
50	GSA	6.2569×10^{-9}	3.7704×10^{-9}	4.9280×10^{-9}	6.2123×10^{-10}
	RGSA	2.5757×10^{-14}	1.5099×10^{-14}	2.0191×10^{-14}	3.1890×10^{-15}
	GGSA	6.4020×10^{-10}	1.9065×10^{-10}	2.6571×10^{-10}	8.2614×10^{-11}
	LGSA	2.2204×10^{-14}	$\mathbf{8.8818 \times 10^{-16}}$	5.7436×10^{-15}	4.4239×10^{-15}
	RGGSA	0.0193	1.1546×10^{-14}	0.0014	0.0043
	RLGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**
	GLGSA	4.4409×10^{-15}	$\mathbf{8.8818 \times 10^{-16}}$	1.2434×10^{-15}	1.0840×10^{-15}
	RGLGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**

Table 6. Test results for $f_5(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	2.8447×10^{-9}	1.3648×10^{-9}	2.2171×10^{-9}	3.7633×10^{-10}
	RGSA	0.0045	3.8202×10^{-18}	8.5381×10^{-4}	0.0013
	GGSA	1.5427×10^{-10}	7.2127×10^{-11}	1.1032×10^{-10}	1.8751×10^{-11}
	LGSA	1.0398×10^{-14}	7.2806×10^{-17}	1.3223×10^{-15}	2.1370×10^{-15}
	RGGSA	0.0069	4.7484×10^{-21}	0.0014	0.0021
	RLGSA	6.9037×10^{-25}	2.8753×10^{-26}	1.2272×10^{-25}	1.3593×10^{-25}
	GLGSA	2.3019×10^{-16}	7.6262×10^{-18}	6.1309×10^{-17}	5.3079×10^{-17}
	RGLGSA	$\mathbf{2.1912 \times 10^{-27}}$	$\mathbf{3.3527 \times 10^{-29}}$	$\mathbf{2.2977 \times 10^{-28}}$	$\mathbf{4.0053 \times 10^{-28}}$
50	GSA	0.0075	4.2822×10^{-9}	7.8908×10^{-4}	0.0017
	RGSA	0.0239	2.7429×10^{-14}	0.0055	0.0053
	GGSA	0.0039	1.9820×10^{-10}	7.3396×10^{-4}	0.0011
	LGSA	8.0741×10^{-15}	6.6396×10^{-17}	1.2979×10^{-15}	1.5773×10^{-15}
	RGGSA	0.0361	1.4921×10^{-5}	0.0105	0.0086
	RLGSA	6.5537×10^{-25}	1.5424×10^{-26}	1.4210×10^{-25}	1.2736×10^{-25}
	GLGSA	2.4325×10^{-16}	1.1630×10^{-17}	5.7302×10^{-17}	4.5711×10^{-17}
	RGLGSA	$\mathbf{9.6359 \times 10^{-28}}$	$\mathbf{6.0262 \times 10^{-29}}$	$\mathbf{3.0080 \times 10^{-28}}$	$\mathbf{2.1554 \times 10^{-28}}$

Table 7. Test results for $f_6(x)$.

D	Algorithm	Worst	Best	Mean	SD
2	GSA	**0.3979**	**0.3979**	**0.3979**	**0**
	RGSA	**0.3979**	**0.3979**	**0.3979**	**0**
	GGSA	**0.3979**	**0.3979**	**0.3979**	**0**
	LGSA	**0.3979**	**0.3979**	**0.3979**	**0**
	RGGSA	**0.3979**	**0.3979**	**0.3979**	**0**
	RLGSA	**0.3979**	**0.3979**	**0.3979**	**0**
	GLGSA	**0.3979**	**0.3979**	**0.3979**	**0**
	RGLGSA	**0.3979**	**0.3979**	**0.3979**	**0**

Table 8. Test results for $f_7(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	441.6230	100.6353	232.0828	73.4886
	RGSA	3.0106×10^{3}	905.0757	1.7284×10^{3}	568.6537
	GGSA	173.7834	36.2260	101.6022	39.9589
	LGSA	2.2037×10^{-24}	2.6581×10^{-30}	7.9806×10^{-26}	4.0151×10^{-25}
	RGGSA	4.3546×10^{3}	839.7732	2.3577×10^{3}	895.2346
	RLGSA	4.2249×10^{-46}	2.3761×10^{-50}	7.2154×10^{-47}	1.1422×10^{-46}
	GLGSA	2.5695×10^{-28}	5.9115×10^{-33}	3.4041×10^{-29}	7.3331×10^{-29}
	RGLGSA	$\mathbf{1.0902 \times 10^{-51}}$	$\mathbf{8.8591 \times 10^{-55}}$	$\mathbf{2.3283 \times 10^{-52}}$	$\mathbf{3.4958 \times 10^{-52}}$
50	GSA	1.5986×10^{3}	678.5357	988.3067	261.3114
	RGSA	8.8062×10^{3}	3.8307×10^{3}	5.6479×10^{3}	1.2912×10^{3}
	GGSA	865.9041	351.5170	642.5225	125.1997
	LGSA	1.0505×10^{-24}	1.2817×10^{-29}	7.9992×10^{-26}	2.1493×10^{-25}
	RGGSA	1.0125×10^{4}	4.1745×10^{3}	6.9878×10^{3}	1.2641×103
	RLGSA	7.9867×10^{-46}	1.2999×10^{-49}	1.1830×10^{-46}	1.8003×10^{-46}
	GLGSA	8.3184×10^{-28}	3.3753×10^{-32}	9.8107×10^{-29}	1.9489×10^{-28}
	RGLGSA	$\mathbf{3.9046 \times 10^{-51}}$	$\mathbf{9.6762 \times 10^{-56}}$	$\mathbf{5.4271 \times 10^{-52}}$	$\mathbf{8.8234 \times 10^{-52}}$

Table 9. Test results for $f_8(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	0.0386	0.0090	0.0193	0.0072
	RGSA	0.0587	0.0093	0.0301	0.0118
	GGSA	0.0547	0.0167	0.0310	0.0105
	LGSA	1.9174×10^{-4}	8.5337×10^{-8}	4.5438×10^{-5}	4.6017×10^{-5}
	RGGSA	0.0772	0.0161	0.0455	0.0156
	RLGSA	$\mathbf{1.1234 \times 10^{-4}}$	7.6825×10^{-7}	$\mathbf{3.7319 \times 10^{-5}}$	$\mathbf{3.0182 \times 10^{-5}}$
	GLGSA	2.0404×10^{-4}	$\mathbf{1.1790 \times 10^{-7}}$	5.0362×10^{-5}	5.3880×10^{-5}
	RGLGSA	1.8700×10^{-4}	1.1549×10^{-6}	4.5049×10^{-5}	4.7092×10^{-5}
50	GSA	0.1932	0.0296	0.0627	0.0317
	RGSA	0.2004	0.0517	0.1121	0.0359
	GGSA	0.1453	0.0307	0.0783	0.0290
	LGSA	2.9610×10^{-4}	3.1457×10^{-6}	5.6102×10^{-5}	5.8344×10^{-5}
	RGGSA	0.2718	0.0840	0.1449	0.0453
	RLGSA	1.8462×10^{-4}	$\mathbf{6.1063 \times 10^{-7}}$	3.6584×10^{-5}	3.8892×10^{-5}
	GLGSA	$\mathbf{1.1842 \times 10^{-4}}$	2.1433×10^{-6}	$\mathbf{3.3349 \times 10^{-5}}$	$\mathbf{2.7926 \times 10^{-5}}$
	RGLGSA	1.9608×10^{-4}	7.4748×10^{-7}	3.9030×10^{-5}	4.4529×10^{-5}

Table 10. Test results for $f_9(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	7.4411	1.3746	4.0420	1.5575
	RGSA	0.0074	**0**	2.4653×10^{-4}	0.0014
	GGSA	0.0537	**0**	0.0079	0.0162
	LGSA	**0**	**0**	**0**	**0**
	RGGSA	0.0123	**0**	9.8565×10^{-4}	0.0031
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	23.4237	11.0850	17.2132	3.4714
	RGSA	0.0124	**0**	0.0015	0.0036
	GGSA	1.1759	**0**	0.1319	0.2313
	LGSA	**0**	**0**	**0**	**0**
	RGGSA	0.0099	**0**	3.5078×10^{-4}	0.0018
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

Table 11. Test results for $f_{10}(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	22.3692	9.7247	15.4151	2.8357
	RGSA	30.9017	10.7804	20.9584	5.1313
	GGSA	20.0970	6.6160	12.5334	3.7834
	LGSA	7.1106×10^{-24}	7.4616×10^{-30}	1.2373×10^{-24}	1.7594×10^{-24}
	RGGSA	45.7283	23.1280	35.3209	6.6800
	RLGSA	1.0930×10^{-45}	2.7914×10^{-47}	2.5198×10^{-46}	2.5438×10^{-46}
	GLGSA	1.3735×10^{-26}	3.1561×10^{-31}	2.2552×10^{-27}	3.0441×10^{-27}
	RGLGSA	$\mathbf{2.5063 \times 10^{-51}}$	$\mathbf{7.9170 \times 10^{-53}}$	$\mathbf{6.7030 \times 10^{-52}}$	$\mathbf{6.1941 \times 10^{-52}}$
50	GSA	49.1786	18.8750	31.4671	6.6325
	RGSA	47.8996	27.5320	37.6831	5.7390
	GGSA	42.9315	14.3102	29.6947	8.4815
	LGSA	2.1156×10^{-23}	9.4851×10^{-29}	4.9534×10^{-24}	5.0170×10^{-24}
	RGGSA	83.9796	55.0682	67.5294	8.7154
	RLGSA	2.2174×10^{-45}	5.9626×10^{-47}	5.5519×10^{-46}	4.4312×10^{-46}
	GLGSA	3.4122×10^{-26}	1.9063×10^{-30}	8.4759×10^{-27}	9.3119×10^{-27}
	RGLGSA	$\mathbf{6.6221 \times 10^{-51}}$	$\mathbf{1.3981 \times 10^{-52}}$	$\mathbf{1.5619 \times 10^{-51}}$	$\mathbf{1.3807 \times 10^{-51}}$

Table 12. Test results for $f_{11}(x)$.

D	Algorithm	Worst	Best	Mean	SD
2	GSA	6.9837×10^{-19}	2.4931×10^{-21}	1.4808×10^{-19}	1.5281×10^{-19}
	RGSA	1.5693×10^{-36}	3.9431×10^{-39}	3.6757×10^{-37}	4.0331×10^{-37}
	GGSA	1.1021×10^{-21}	3.8066×10^{-24}	2.6538×10^{-22}	2.4632×10^{-22}
	LGSA	2.5328×10^{-25}	1.0902×10^{-35}	2.4752×10^{-26}	6.4866×10^{-26}
	RGGSA	1.6362×10^{-41}	4.2936×10^{-44}	2.4011×10^{-42}	3.8616×10^{-42}
	RLGSA	3.1987×10^{-47}	2.1284×10^{-53}	1.9892×10^{-48}	5.9666×10^{-48}
	GLGSA	2.5843×10^{-27}	3.2845×10^{-35}	2.1927×10^{-28}	5.3440×10^{-28}
	RGLGSA	$\mathbf{4.3268 \times 10^{-53}}$	$\mathbf{6.2023 \times 10^{-60}}$	$\mathbf{5.7435 \times 10^{-54}}$	$\mathbf{1.1301 \times 10^{-53}}$

Table 13. Test results for $f_{12}(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	21.8891	7.9597	14.8912	3.5586
	RGSA	34.8235	13.9294	22.2539	5.7095
	GGSA	26.8639	9.9496	18.5726	4.2961
	LGSA	**0**	**0**	**0**	**0**
	RGGSA	34.8235	14.9244	25.6699	5.8915
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	50.7429	22.8841	33.0326	6.0459
	RGSA	50.7429	23.8790	37.7089	7.0031
	GGSA	47.7580	21.8891	37.0125	6.4396
	LGSA	**0**	**0**	**0**	**0**
	RGGSA	81.5865	34.8235	51.2403	11.3435
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

Table 14. Test results for $f_{13}(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	**0**	**0**	**0**	**0**
	RGSA	**0**	**0**	**0**	**0**
	GGSA	**0**	**0**	**0**	**0**
	LGSA	**0**	**0**	**0**	**0**
	RGGSA	**0**	**0**	**0**	**0**
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	4	**0**	0.6333	0.9994
	RGSA	5	**0**	0.6667	1.2130
	GGSA	**0**	**0**	**0**	**0**
	LGSA	**0**	**0**	**0**	**0**
	RGGSA	4	**0**	0.7667	1.0400
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

Table 15. Test results for $f_{14}(x)$.

D	Algorithm	Worst	Best	Mean	SD
30	GSA	2.4001	0.8014	1.2938	0.4433
	RGSA	1.2989	0.7001	0.9479	0.1567
	GGSA	0.9031	0.5999	0.6820	0.0680
	LGSA	5.6621×10^{-15}	**0**	1.1361×10^{-15}	1.5394×10^{-15}
	RGGSA	1.4072	0.8052	1.1264	0.1450
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	1.1102×10^{-16}	**0**	1.1102×10^{-17}	3.3876×10^{-17}
	RGLGSA	**0**	**0**	**0**	**0**

Table 15. *Cont.*

D	Algorithm	Worst	Best	Mean	SD
50	GSA	4.8987	2.4057	3.3326	0.5198
	RGSA	2.3001	1.4005	1.7868	0.2488
	GGSA	2.6005	0.9999	1.6749	0.3434
	LGSA	8.3267×10^{-15}	1.1102×10^{-16}	1.3582×10^{-15}	1.7851×10^{-15}
	RGGSA	2.5008	1.5183	1.8890	0.2373
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	2.2204×10^{-16}	**0**	3.3307×10^{-17}	5.9395×10^{-17}
	RGLGSA	**0**	**0**	**0**	**0**

Table 16. Test results for $f_{15}(x)$.

D	Algorithm	Worst	Best	Mean	SD
2	GSA	0.0098	1.3534×10^{-5}	0.0063	0.0039
	RGSA	0.0097	1.0306×10^{-4}	0.0054	0.0038
	GGSA	0.0097	2.7770×10^{-4}	0.0041	0.0032
	LGSA	**0**	**0**	**0**	**0**
	RGGSA	0.0097	5.8631×10^{-4}	0.0057	0.0033
	RLGSA	**0**	**0**	**0**	**0**
	GLGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

First, the results of the RGSA, GGSA, and LGSA on the test functions are better than those of the GSA. The results show that the three adaptive strategies of population density, gravitational constant, and location update can effectively improve the performance of the GSA. The detailed analysis is as follows. The result of the GGSA is inferior to those of the RGSA and LGSA, but superior to that of the GSA. This shows that although the adaptive gravitational constant strategy is inferior to the crowd density and location update strategy in improving the performance of the GSA, it also improves the performance of the GSA to a certain extent because it helps to improve the iteration step size and convergence speed. The result of the RGSA is much better than that of the GSA, which indicates that the adaptive population density strategy dynamically adjusts the distance between particles according to the population density in the evolution process and better balances the exploration and mining capabilities of the algorithm, thus improving the search capability of the algorithm. The LGSA is not only better than the GSA, but also better than the GGSA and RGSA, which shows that the location update strategy plays the largest role in improving the performance of the GSA and reflects that the balance of location and speed between individuals is the key to ensuring the good solution quality of a swarm intelligence algorithm.

Second, the results of the RGGSA, RLGSA, and GLGSA on the test function are better than those of the RGSA, GGSA, and LGSA. The results show that the combination of two of the three adaptive strategies proposed in this paper can effectively improve the performance of the GSAs using a single strategy. The result of the GLGSA is better than that of the RGGSA, but it is also inferior to that of the RLGSA, which indicates that the better single improvement strategy still has a higher performance advantage in the combined improvement strategy. The test result of the RGLGSA is higher than those of the RGGSA, RLGSA and GLGSA. The results shows that the RGSGSA, which combines the three strategies, leverages the advantages of the RGSA, GGSA, and LGSA, making the advantages more effective in the search process and achieving the best solution performance.

5.2. Comparison and Analysis of Algorithm Test Results

To fully evaluate the overall performance of the adaptive GSA proposed in this paper (RGLGSA), we selected the following classic and efficient GSA algorithms: the weight-based GSA (GSAGJ) [29], SGSA [18], and multipoint adaptive constraint-based gravitation improved algorithm (MACGSA) [19]. A comparative analysis of simulation tests was performed using 16 benchmark test functions of the CEC2017 benchmark. In the experiment, we tested the convergence of the basic GSA and the four comparison algorithms in 10, 30, and 50 dimensions.

5.2.1. Test Function

In this study, 16 benchmark test functions of CEC2017 were selected to test the effectiveness of the proposed algorithm. Table 17 lists each test function. Among the 16 test functions, f19 and f23 take the minimum value 0 at point (1, 1, ..., 1), f29 and f30 take the minimum value 0 at point (−1, −1, ..., −1), and the other functions take the minimum value 0 at point (0, 0, ..., 0). Functions f16, f17, f18, f24, and f25 are multidimensional unimodal reference functions, whereas f19–f23 and f26–f31 are multidimensional multimodal reference functions.

Table 17. CEC2017 benchmark function set.

Function No.	Test Function	Value Range
16	$f_{16}(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$	$x_i \in [-100, 100]$
17	$f_{17}(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	$x_i \in [-100, 100]$
18	$f_{18}(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5x_i)^2 + (\sum_{i=1}^{n} 0.5x_i)^4$	$x_i \in [-100, 100]$
19	$f_{19}(x) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	$x_i \in [-10, 10]$
20	$f_{20}(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$x_i \in [-5.12, 5.12]$
21	$g(x,y) = 0.5 + \frac{(\sin^2(\sqrt{x^2+y^2}) - 0.5)}{(1 + 0.001(x^2+y^2))^2}$ $f_{21}(x) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_{n-1}, x_n) + g(x_n, x_1)$	$x_i \in [-100, 100]$
22	$y_i = \begin{cases} x_i \rightarrow \lvert x_i \rvert \langle \frac{1}{2} \\ round(2x_i)/2 \rightarrow \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$ $f_{22}(x) = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10)$	$x_i \in [-100, 100]$
23	$y_i = 1 + \frac{x_i - 1}{4}, \forall i = 1, \ldots, n$ $f_{23}(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_i + 1)] + (y_n - 1)^2 [1 + \sin^2(2\pi y_n)]$	$x_i \in [-100, 100]$
24	$f_{24}(x) = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2$	$x_i \in [-100, 100]$
25	$f_{25}(x) = \sum_{i=2}^{n} {x_i}^2 + 10^6 x_1^2$	$x_i \in [-100, 100]$
26	$f_{26}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	$x_i \in [-32, 32]$
27	$f_{27}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$x_i \in [-600, 600]$
28	$f_{28}(x) = \frac{10}{n^2}\prod_{i=1}^{n}(1 + i\sum_{j=1}^{32}\frac{\lvert 2^j x_i - round(2^j x_i)\rvert}{2^j})^{\frac{10}{n^{1.2}}} - \frac{10}{n^2}$	$x_i \in [-100, 100]$
29	$f_{29}(x) = \left\lvert \sum_{i=1}^{n} x_i^2 - n \right\rvert^{1/4} + (0.5\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_i)/n + 0.5$	$x_i \in [-100, 100]$
30	$f_{30}(x) = \left\lvert (\sum_{i=1}^{n} x_i^2)^2 - (\sum_{i=1}^{n} x_i)^2 \right\rvert^{1/2} + (0.5\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_i)/n + 0.5$	$x_i \in [-100, 100]$
31	$y_i = \sqrt{x_i^2 + x_{i+1}^2}$ $f_{31}(x) = \left[\frac{1}{n-1}\sum_{i=1}^{n-1}(\sqrt{y_i}(\sin(50y_i^{0.2}) + 1))^2\right]^2$	$x_i \in [-100, 100]$

5.2.2. Data Analysis

To prevent errors caused by accidental factors and to ensure objectivity and fairness of the evaluation, in the experiment, the five algorithms were independently run 30 times and were iterated 1000 times in the same environment. The other parameter settings of the algorithms were consistent with those listed in Section 5.1.

The CEC2017 benchmark test function simulation results in Tables 18–33 reveal the following.

Table 18. Test results for $f_{16}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	**927.9558**	**0.1349**	**131.5188**	**208.5184**
	GSAGJ	2.0404×10^{3}	0.0028	416.9901	526.4530
	SGSA	4.5419×10^{3}	0.0048	878.3686	1.1364×10^{3}
	MACGSA	2.3696×10^{-29}	1.4460×10^{-34}	2.3722×10^{-30}	5.3549×10^{-30}
	RGLGSA	$\mathbf{6.4516 \times 10^{-45}}$	$\mathbf{3.8488 \times 10^{-50}}$	$\mathbf{9.3471 \times 10^{-46}}$	$\mathbf{1.8916 \times 10^{-45}}$
30	GSA	**318.6005**	0.0013	74.1322	80.4764
	GSAGJ	**906.5667**	1.2895	209.9172	243.1416
	SGSA	$\mathbf{6.8746 \times 10^{3}}$	1.0985	1.5708×10^{3}	1.5861×10^{3}
	MACGSA	$\mathbf{8.2065 \times 10^{-28}}$	6.1559×10^{-33}	5.0553×10^{-29}	1.5421×10^{-28}
	RGLGSA	$\mathbf{2.8601 \times 10^{-42}}$	$\mathbf{4.7112 \times 10^{-48}}$	$\mathbf{1.6732 \times 10^{-43}}$	$\mathbf{5.4578 \times 10^{-43}}$
50	GSA	355.4676	0.7546	81.2026	86.5312
	GSAGJ	1.7790×10^{3}	0.0090	305.5190	394.2691
	SGSA	5.8571×10^{3}	0.0504	988.6499	1.4382×10^{3}
	MACGSA	2.8593×10^{-27}	9.2657×10^{-31}	2.6063×10^{-28}	5.9198×10^{-28}
	RGLGSA	$\mathbf{1.0848 \times 10^{-42}}$	$\mathbf{2.8539 \times 10^{-47}}$	$\mathbf{1.1947 \times 10^{-43}}$	$\mathbf{2.4321 \times 10^{-43}}$

Table 19. Test results for $f_{17}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	8.6678×10^{-11}	1.1821×10^{-14}	1.3260×10^{-11}	2.1751×10^{-11}
	GSAGJ	9.4942×10^{-13}	1.0772×10^{-15}	1.7974×10^{-13}	2.7312×10^{-13}
	SGSA	5.5934×10^{-8}	9.2999×10^{-14}	2.1619×10^{-9}	1.0164×10^{-8}
	MACGSA	2.3449×10^{-79}	1.8483×10^{-93}	1.7312×10^{-80}	5.4499×10^{-80}
	RGLGSA	$\mathbf{8.8740 \times 10^{-108}}$	$\mathbf{4.0256 \times 10^{-127}}$	$\mathbf{4.6188 \times 10^{-109}}$	$\mathbf{1.8277 \times 10^{-108}}$
30	GSA	2.7721×10^{9}	22.8367	1.1527×10^{8}	5.0804×10^{8}
	GSAGJ	6.9577×10^{5}	59.9353	9.5043×10^{4}	1.9611×10^{5}
	SGSA	2.7279×10^{8}	115.3677	1.4384×10^{7}	5.1481×10^{7}
	MACGSA	8.5890×10^{-102}	1.7331×10^{-118}	3.1893×10^{-103}	1.5710×10^{-102}
	RGLGSA	$\mathbf{5.9700 \times 10^{-122}}$	$\mathbf{7.6982 \times 10_{-135}}$	$\mathbf{2.0933 \times 10^{-123}}$	$\mathbf{1.0888 \times 10^{-122}}$
50	GSA	3.4420×10^{29}	1.0669×10^{15}	2.2045×10^{28}	8.1793×10^{28}
	GSAGJ	2.5940×10^{23}	8.6661×10^{10}	1.2069×10^{22}	4.8832×10^{22}
	SGSA	4.1466×10^{24}	8.2921×10^{13}	1.5953×10^{23}	7.6141×10^{23}
	MACGSA	1.7681×10^{-107}	4.2408×10^{-121}	6.4581×10^{-109}	3.2318×10^{-108}
	RGLGSA	$\mathbf{1.0401 \times 10^{-123}}$	$\mathbf{1.5384 \times 10^{-144}}$	$\mathbf{3.5856 \times 10^{-125}}$	$\mathbf{1.8973 \times 10^{-124}}$

Table 20. Test results for $f_{18}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	3.4437×10^{-18}	3.1320×10^{-19}	2.0191×10^{-18}	7.2080×10^{-19}
	GSAGJ	7.4174×10^{-17}	7.2964×10^{-18}	3.7707×10^{-17}	1.4441×10^{-17}
	SGSA	5.5986×10^{-34}	3.5185×10^{-35}	2.7418×10^{-34}	1.1874×10^{-34}
	MACGSA	1.5097×10^{-33}	6.4933×10^{-38}	1.3977×10^{-34}	2.9910×10^{-34}
	RGLGSA	$\mathbf{4.2237 \times 10^{-48}}$	$\mathbf{1.4052 \times 10^{-51}}$	$\mathbf{5.3557 \times 10^{-49}}$	$\mathbf{9.8273 \times 10^{-49}}$

Table 20. *Cont.*

D	Algorithm	Worst	Best	Mean	SD
30	GSA	22.1013	1.1135×10^{-17}	0.7773	4.0327
	GSAGJ	93.9609	2.2794×10^{-16}	5.1395	18.3694
	SGSA	4.0050×10^{3}	736.0369	2.3252×10^{3}	872.4120
	MACGSA	9.4929×10^{-33}	3.0631×10^{-35}	1.8175×10^{-33}	2.6357×10^{-33}
	RGLGSA	$\mathbf{2.3447 \times 10^{-47}}$	$\mathbf{3.6562 \times 10^{-50}}$	$\mathbf{3.2662 \times 10^{-48}}$	$\mathbf{5.8753 \times 10^{-48}}$
50	GSA	812.8504	56.8831	373.8760	184.1014
	GSAGJ	1.3323×10^{3}	134.8774	501.4951	327.9025
	SGSA	7.1515×10^{3}	2.8452×10^{3}	4.9205×10^{3}	917.0428
	MACGSA	1.5722×10^{-32}	5.1156×10^{-36}	1.4101×10^{-33}	3.0031×10^{-33}
	RGLGSA	$\mathbf{2.1532 \times 10^{-46}}$	$\mathbf{8.1105 \times 10^{-50}}$	$\mathbf{2.3062 \times 10^{-47}}$	$\mathbf{5.3073 \times 10^{-47}}$

Table 21. Test results for $f_{19}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	**5.6309**	5.0609	5.4357	**0.1358**
	GSAGJ	5.7058	**5.0384**	**5.3939**	0.1572
	SGSA	6.6388	6.0138	6.3808	0.1723
	MACGSA	7.0299	6.2602	6.5791	0.1821
	RGLGSA	7.4642	6.4863	6.9113	0.2123
30	GSA	26.6071	**25.5609**	26.0710	0.2053
	GSAGJ	**26.4659**	25.6065	**26.0522**	**0.1975**
	SGSA	28.2171	26.4256	26.9333	0.4014
	MACGSA	27.8766	25.8729	27.1567	0.3542
	RGLGSA	27.9882	27.0381	27.3609	0.2197
50	GSA	49.2016	45.7221	46.5439	0.6840
	GSAGJ	49.2007	**45.4268**	**46.4512**	0.7858
	SGSA	99.6238	45.6100	49.4379	10.0356
	MACGSA	48.9909	46.1487	47.8639	0.8066
	RGLGSA	**48.9011**	46.5779	47.7200	**0.5182**

Table 22. Test results for $f_{20}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	5.9698	**0.9950**	2.9517	**1.2389**
	GSAGJ	7.9597	**0**	3.6813	1.8318
	SGSA	9.9496	1.9899	5.5718	2.0507
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
30	GSA	23.8790	7.9597	14.9907	3.6105
	GSAGJ	32.8336	11.9395	18.4067	4.0119
	SGSA	36.8135	10.9445	25.5041	6.6435
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	42.7832	20.8941	31.9050	6.0093
	GSAGJ	57.7075	18.9042	35.0889	8.5387
	SGSA	61.6874	31.8387	47.1610	7.9322
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

Table 23. Test results for $f_{21}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	2.7304	0.9393	1.5832	0.4171
	GSAGJ	3.3976	1.7065	2.6364	0.4612
	SGSA	3.1766	1.4350	2.8282	0.3456
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
30	GSA	5.1985	2.9800	4.1999	0.6364
	GSAGJ	9.7717	5.3134	7.6816	0.9245
	SGSA	11.9302	9.2954	10.8964	0.6814
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	8.3152	4.4391	6.4113	0.9250
	GSAGJ	13.0620	8.2424	10.8121	1.2371
	SGSA	20.6838	15.8774	18.6364	1.2354
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

Table 24. Test results for $f_{22}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	7.3448	2	4.4654	1.5347
	GSAGJ	8	2	5.1476	1.6727
	SGSA	11.0040	3	6.7614	2.1870
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
30	GSA	55	12	22.4959	8.1175
	GSAGJ	39	17	26.8000	5.0950
	SGSA	57	20	35.0667	8.0982
	MACGSA	141.0086	**0**	7.7670	30.2505
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	175	57	108.8667	29.6133
	GSAGJ	94	42	59.8333	12.4957
	SGSA	107	47	79.9667	14.1262
	MACGSA	850.2239	**0**	145.2359	299.2123
	RGLGSA	**0**	**0**	**0**	**0**

Table 25. Test results for $f_{23}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	1.0829×10^{-18}	1.9660×10^{-19}	5.6692×10^{-19}	2.3275×10^{-19}
	GSAGJ	2.7991×10^{-17}	5.0265×10^{-18}	1.1209×10^{-17}	5.3271×10^{-18}
	SGSA	$\mathbf{1.4998 \times 10^{-32}}$	$\mathbf{1.4998 \times 10^{-32}}$	$\mathbf{1.4998 \times 10^{-32}}$	$\mathbf{1.1135 \times 10^{-47}}$
	MACGSA	5.0085×10^{-7}	1.0793×10^{-7}	3.0263×10^{-7}	1.0103×10^{-7}
	RGLGSA	8.4957×10^{-6}	1.3167×10^{-6}	4.2384×10^{-6}	1.6172×10^{-6}
30	GSA	45.0420	5.2599×10^{-18}	4.2947	10.5661
	GSAGJ	2.8179	8.8351×10^{-17}	0.1818	0.5973
	SGSA	**0.4543**	$\mathbf{1.4998 \times 10^{-32}}$	0.0665	0.1563
	MACGSA	3.2595	2.9752	3.2215	**0.0676**
	RGLGSA	0.4546	2.4474×10^{-4}	**0.0185**	0.0840
50	GSA	177.5581	3.9951	68.4516	41.2278
	GSAGJ	18.0904	3.6758×10^{-16}	1.3904	3.4206
	SGSA	7.7252	$\mathbf{4.1341 \times 10^{-31}}$	1.4550	1.7429
	MACGSA	5.0764	4.7496	5.0298	**0.0885**
	RGLGSA	**3.2825**	0.0024	**0.6285**	0.8729

Table 26. Test results for $f_{24}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	1.5824×10^5	65.0845	2.1881×10^4	2.9856×10^4
	GSAGJ	1.5669×10^5	2.2076×10^3	3.5142×10^4	3.8005×10^4
	SGSA	2.1238×10^5	4.9374×10^3	7.4512×10^4	5.7663×10^4
	MACGSA	1.4877×10^{-32}	6.5895×10^{-37}	1.6068×10^{-33}	3.6474×10^{-33}
	RGLGSA	$\mathbf{1.2592 \times 10^{-46}}$	$\mathbf{3.9071 \times 10^{-51}}$	$\mathbf{9.0954 \times 10^{-48}}$	$\mathbf{2.6328 \times 10^{-47}}$
30	GSA	2.2231×10^5	1.3209×10^4	7.7308×10^4	5.3591×10^4
	GSAGJ	$2.4017 \times 10_5$	9.2003×10^3	8.5470×10^4	6.0342×10^4
	SGSA	3.1540×10^5	4.3732×10^4	1.4493×10^5	6.9453×10^4
	MACGSA	3.6955×10^{-32}	1.8249×10^{-35}	3.2962×10^{-33}	7.3649×10^{-33}
	RGLGSA	$\mathbf{1.1867 \times 10^{-46}}$	6.5780×10^{-50}	$\mathbf{1.3542 \times 10^{-47}}$	$\mathbf{2.7548 \times 10^{-47}}$
50	GSA	2.2231×10^5	1.3209×10^4	7.7308×10^4	5.3591×10^4
	GSAGJ	2.4017×10^5	9.2003×10^3	8.5470×10^4	6.0342×10^4
	SGSA	3.1540×105	4.3732×10^4	1.4493×10^5	6.9453×10^4
	MACGSA	3.6955×10^{-32}	1.8249×10^{-35}	3.2962×10^{-33}	7.3649×10^{-33}
	RGLGSA	$\mathbf{1.1867 \times 10^{-46}}$	$\mathbf{6.5780 \times 10^{-50}}$	$\mathbf{1.3542 \times 10^{-47}}$	$\mathbf{2.7548 \times 10^{-47}}$

Table 27. Test results for $f_{25}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	1.6166×10^3	114.4355	717.1659	407.7005
	GSAGJ	3.2975×10^3	424.6107	1.6499×10^3	749.6641
	SGSA	3.0122×10^3	291.2547	1.5685×10^3	670.7036
	MACGSA	5.3094×10^{-33}	2.1696×10^{-37}	2.4561×10^{-34}	9.6672×10^{-34}
	RGLGSA	$\mathbf{7.6013 \times 10^{-48}}$	$\mathbf{1.4909 \times 10^{-51}}$	$\mathbf{6.8777 \times 10^{-49}}$	$\mathbf{1.7501 \times 10^{-48}}$
30	GSA	2.2537×10^3	299.4626	872.2012	393.5168
	GSAGJ	2.3856×10^3	3.3096×10^{-16}	486.5437	610.2161
	SGSA	3.2041×10^{-32}	5.0516×10^{-33}	1.3777×10^{-32}	6.3954×10^{-33}
	MACGSA	3.6098×10^{-32}	5.0320×10^{-36}	2.0449×10^{-33}	6.5866×10^{-33}
	RGLGSA	$\mathbf{2.1523 \times 10^{-47}}$	$\mathbf{4.3108 \times 10^{-50}}$	$\mathbf{3.0550 \times 10^{-48}}$	$\mathbf{5.7203 \times 10^{-48}}$
50	GSA	2.8219×10^3	566.1510	1.7062×10^3	645.7675
	GSAGJ	2.6789×10^3	1.3496×10^{-15}	722.0980	788.4850
	SGSA	2.0515×10^{-30}	8.0778×10^{-32}	4.2586×10^{-31}	4.3106×10^{-31}
	MACGSA	1.3764×10^{-32}	4.2468×10^{-36}	1.3404×10^{-33}	3.3730×10^{-33}
	RGLGSA	$\mathbf{6.3641 \times 10^{-47}}$	$\mathbf{1.1221 \times 10^{-49}}$	$\mathbf{6.7762 \times 10^{-48}}$	$\mathbf{1.5149 \times 10^{-47}}$

Table 28. Test results for $f_{26}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	2.7037×10^{-9}	1.3481×10^{-9}	1.8639×10^{-9}	3.3343×10^{-10}
	GSAGJ	9.7995×10^{-9}	4.6786×10^{-9}	7.5941×10^{-9}	1.3530×10^{-9}
	SGSA	7.9936×10^{-15}	4.4409×10^{-15}	4.5593×10^{-15}	6.4863×10^{-16}
	MACGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**
	RGLGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**
30	GSA	4.5600×10^{-9}	2.8379×10^{-9}	3.5839×10^{-9}	4.5843×10^{-10}
	GSAGJ	1.7140×10^{-8}	1.0784×10^{-8}	1.4658×10^{-8}	1.6459×10^{-9}
	SGSA	2.2204×10^{-14}	7.9936×10^{-15}	1.4744×10^{-14}	3.5343×10^{-15}
	MACGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**
	RGLGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**

Table 28. *Cont.*

D	Algorithm	Worst	Best	Mean	SD
50	GSA	8.2910×10^{-9}	3.6928×10^{-9}	4.9393×10^{-9}	8.8854×10^{-10}
	GSAGJ	2.4806×10^{-8}	1.5528×10^{-8}	1.9258×10^{-8}	2.2218×10^{-9}
	SGSA	5.7732×10^{-14}	1.5099×10^{-14}	2.9073×10^{-14}	7.5758×10^{-15}
	MACGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**
	RGLGSA	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	$\mathbf{8.8818 \times 10^{-16}}$	**0**

Table 29. Test results for $f_{27}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	0.0762	**0**	0.0200	0.0205
	GSAGJ	0.0270	**0**	0.0030	0.0061
	SGSA	0.0172	**0**	0.0031	0.0056
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
30	GSA	9.5227	1.7763	4.2129	1.7361
	GSAGJ	0.0515	**0**	0.0038	0.0108
	SGSA	0.0148	**0**	0.0015	0.0040
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	25.1669	11.4594	17.4486	4.1201
	GSAGJ	0.1761	**0**	0.0356	0.0477
	SGSA	0.0148	**0**	4.9241e-04	0.0027
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

Table 30. Test results for $f_{28}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	1.1961×10^{-10}	6.7969×10^{-11}	1.0001×10^{-10}	1.1193×10^{-11}
	GSAGJ	1.2086×10^{-10}	6.9683×10^{-11}	9.8470×10^{-11}	1.2157×10^{-11}
	SGSA	**0**	**0**	**0**	**0**
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
30	GSA	5.8032×10^{-11}	4.6224×10^{-11}	5.2202×10^{-11}	3.1700×10^{-12}
	GSAGJ	5.6719×10^{-11}	4.7120×10^{-11}	5.2218×10^{-11}	2.6389×10^{-12}
	SGSA	5.8824×10^{-13}	**0**	1.2443×10^{-13}	1.9250×10^{-13}
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**
50	GSA	3.5579×10^{-11}	3.1050×10^{-11}	3.3152×10^{-11}	9.9662×10^{-13}
	GSAGJ	3.4840×10^{-11}	2.7680×10^{-11}	3.3151×10^{-11}	1.4994×10^{-12}
	SGSA	1.2293×10^{-12}	8.3267×10^{-15}	4.3324×10^{-13}	3.1526×10^{-13}
	MACGSA	**0**	**0**	**0**	**0**
	RGLGSA	**0**	**0**	**0**	**0**

Table 31. Test results for $f_{29}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	0.5050	0.1770	0.3001	0.0798
	GSAGJ	**0.3542**	**0.1485**	**0.2515**	**0.0563**
	SGSA	0.5103	0.2174	0.3332	0.0810
	MACGSA	1.5100	0.4278	0.7252	0.2975
	RGLGSA	0.6482	0.3033	0.4403	0.0917

Table 31. *Cont.*

D	Algorithm	Worst	Best	Mean	SD
30	GSA	**0.1652**	**0.0440**	**0.1174**	**0.0290**
	GSAGJ	0.1938	0.0815	0.1362	0.0292
	SGSA	0.2739	0.1108	0.1815	0.0425
	MACGSA	0.4296	0.1637	0.2719	0.0714
	RGLGSA	0.3036	0.1204	0.2153	0.0409
50	GSA	0.5050	0.1770	0.3001	0.0798
	GSAGJ	**0.3542**	**0.1485**	**0.2515**	**0.0563**
	SGSA	0.5103	0.2174	0.3332	0.0810
	MACGSA	1.5100	0.4278	0.7252	0.2975
	RGLGSA	0.6482	0.3033	0.4403	0.0917

Table 32. Test results for $f_{30}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	0.5008	0.3570	**0.4925**	0.0267
	GSAGJ	0.5010	0.4387	0.4944	0.0175
	SGSA	0.5009	**0.3110**	0.4895	0.0412
	MACGSA	**0.5000**	0.3965	0.4860	0.0285
	RGLGSA	**0.5000**	0.4574	0.4981	**0.0081**
30	GSA	0.5013	0.4344	0.4884	**0.0159**
	GSAGJ	0.5010	0.3233	0.4581	0.0495
	SGSA	0.5016	**0.2767**	**0.4204**	0.0553
	MACGSA	**0.5000**	0.3568	0.4661	0.0438
	RGLGSA	**0.5000**	0.4159	0.4851	0.0263
50	GSA	0.6156	**0.2404**	0.3872	0.0743
	GSAGJ	**0.4495**	0.2743	**0.3814**	0.0405
	SGSA	0.5687	0.2730	0.3726	0.0542
	MACGSA	0.5000	0.3422	0.4711	0.0465
	RGLGSA	0.5000	0.4451	0.4900	**0.0154**

Table 33. Test results for $f_{31}(x)$.

D	Algorithm	Worst	Best	Mean	SD
10	GSA	1.0420×10^{-4}	2.7999×10^{-34}	4.6894×10^{-6}	1.9938×10^{-5}
	GSAGJ	2.8974×10^{-7}	2.1420×10^{-32}	1.9555×10^{-8}	7.1460×10^{-8}
	SGSA	1.9305×10^{-5}	**0**	7.0894×10^{-7}	3.5154×10^{-6}
	MACGSA	1.1429×10^{-21}	5.5902×10^{-24}	4.5938×10^{-22}	3.0559×10^{-22}
	RGLGSA	$\mathbf{7.1484 \times 10^{-30}}$	2.1262×10^{-32}	$\mathbf{1.0346 \times 10^{-30}}$	$\mathbf{1.3575 \times 10^{-30}}$
30	GSA	0.0786	1.5937×10^{-5}	0.0118	0.0185
	GSAGJ	5.8248×10^{-4}	1.3017×10^{-10}	7.2445×10^{-5}	1.4989×10^{-4}
	SGSA	0.0261	1.4485×10^{-6}	0.0040	0.0063
	MACGSA	$7.9223 \times 10+$	1.4338×10^{-23}	2.2198×10^{-22}	1.9444×10^{-22}
	RGLGSA	$\mathbf{1.1549 \times 10^{-30}}$	$\mathbf{9.2467 \times 10^{-33}}$	$\mathbf{3.4346 \times 10^{-31}}$	$\mathbf{2.8391 \times 10^{-31}}$
50	GSA	0.0785	6.1868×10^{-4}	0.0159	0.0207
	GSAGJ	0.0189	1.0070×10^{-5}	0.0031	0.0050
	SGSA	0.0548	5.2674×10^{-4}	0.0101	0.0146
	MACGSA	3.8503×10^{-19}	9.9380×10^{-21}	1.2760×10^{-19}	8.0890×10^{-20}
	RGLGSA	$\mathbf{1.0666 \times 10^{-25}}$	$\mathbf{6.5193 \times 10^{-28}}$	$\mathbf{1.6474 \times 10^{-26}}$	$\mathbf{2.0978 \times 10^{-26}}$

First, for the unimodal functions f16, f18, f24, and f25, both the RGLGSA and MACGSA have high accuracy. Under the same number of dimensions, the optimization accuracy of the RGLGSA proposed in this paper is significantly higher than those of the GSA, GSAGJ, SGSA, and MACGSA. With different dimensions, as the dimensions increase, the solution accuracy of the five algorithms gradually decreases, but the solution accuracy of the RGLGSA is still higher than those of the GSA, GSAGJ, SGSA, and MACGSA. For function f17, both the RGLGSA and MACGSA have very high accuracy. Under the same number of dimensions, the optimization accuracy of the RGLGSA is superior to those of the other four algorithms. As the dimensions increase, the solution accuracy of the GSA, GSAGJ, and SGSA decrease gradually, with an obvious trend. The solution accuracies of the RGLGSA and MACGSA increase gradually, but the solution accuracy of the RGLGSA is still higher than that of the other four algorithms.

For the multimodal functions f19, f29, and f30, the four algorithms all become trapped in the local extrema, with little difference in solution accuracy. For functions f20, f21, f26, f27, and f28, the RGLGSA and MACGSA find the globally optimal solution 0, but other algorithms cannot. For function f22, when the dimensions are 10, the RGLGSA and MACGSA can find the global optimal solution 0; when the dimensions are high, only the RGLGSA can find the globally optimal solution 0. For function f23, when the dimensions are 10, the SGSA has a higher precision, and when the dimensions are 30 and 50, the RGLGSA has the highest precision. For function f31, under the same number of dimensions, the optimization accuracy of the RGLGSA is significantly higher than those of the other algorithms. As the dimensions increase, the solution accuracies of the GSA, GSAGJ, and SGSA decrease gradually. When the dimensions reach 50, the solution accuracies of the MACGSA and RGLGSA decrease significantly.

The comparison and test results of the above algorithms reveal that, compared with other classical and efficient improved GSAs, the RGLGSA has a relatively stable overall search ability on unimodal functions and multimodal functions. Moreover, it has a high convergence accuracy.

5.2.3. Curve Analysis

Figures 1–16 show the convergence curves of the RGLGSA, basic GSA, and the comparison GSAs. The convergence curves of the unimodal functions reveal that there is little difference between the convergence speed of the five algorithms in the first 500 generations. After 500 generations, the GSA, GSAGJ, and SGSA converge to the local extreme values and stop searching. The MACGSA and RGLGSA do not fall into the local extreme values, but continue to evolve. However, the convergence speed of the RGLGSA is faster than that of the MACGSA, and the obtained value is better.

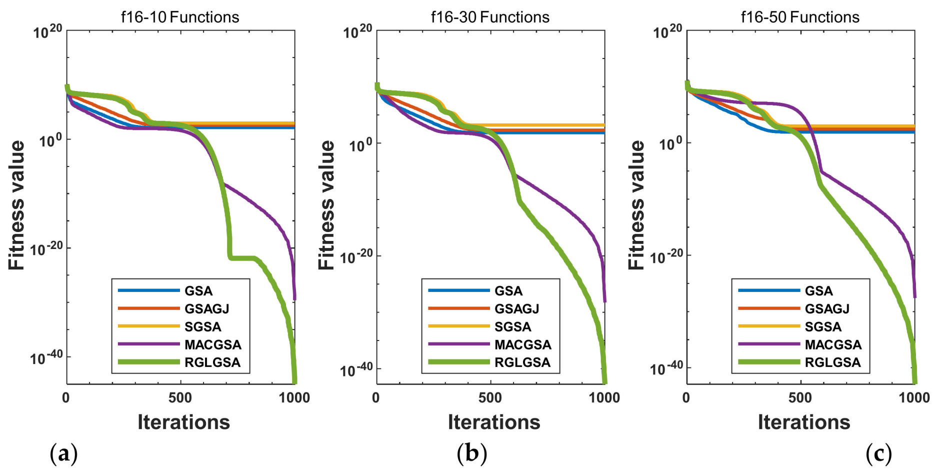
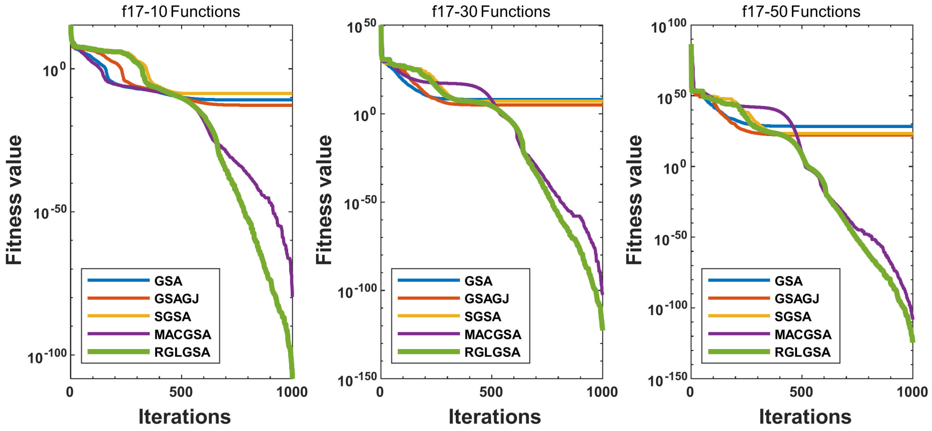
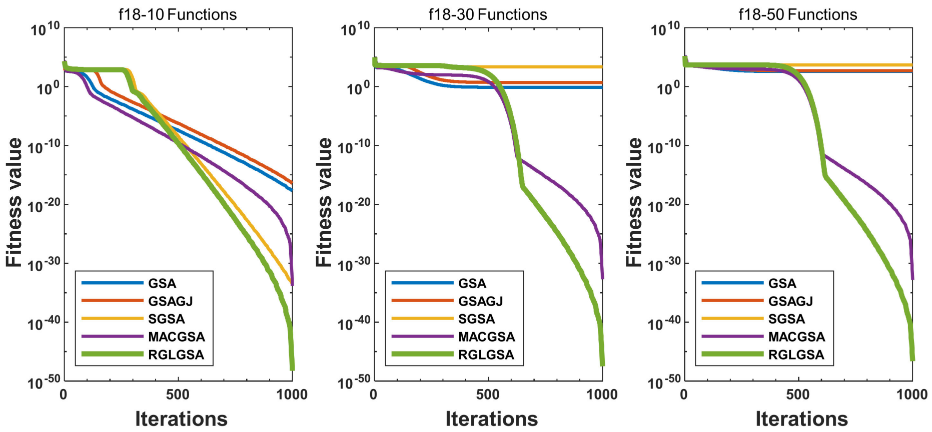


Figure 1. Convergence curves of function f16: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



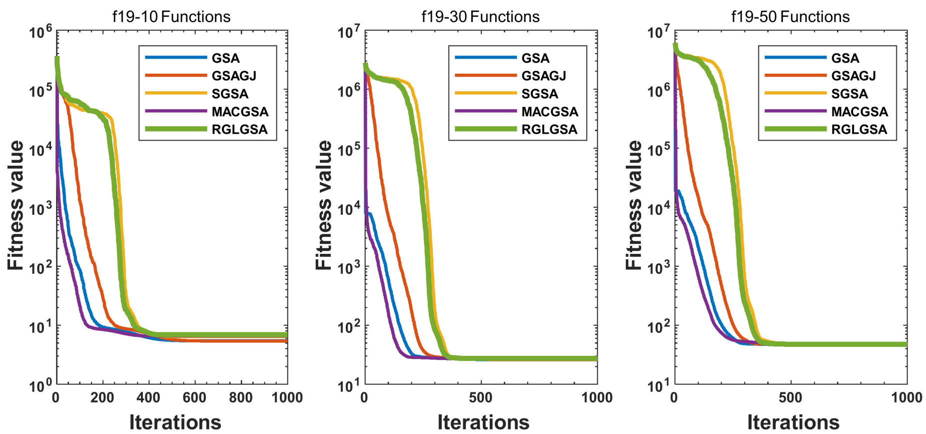
(**a**) (**b**) (**c**)

Figure 2. Convergence curves of function f17: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



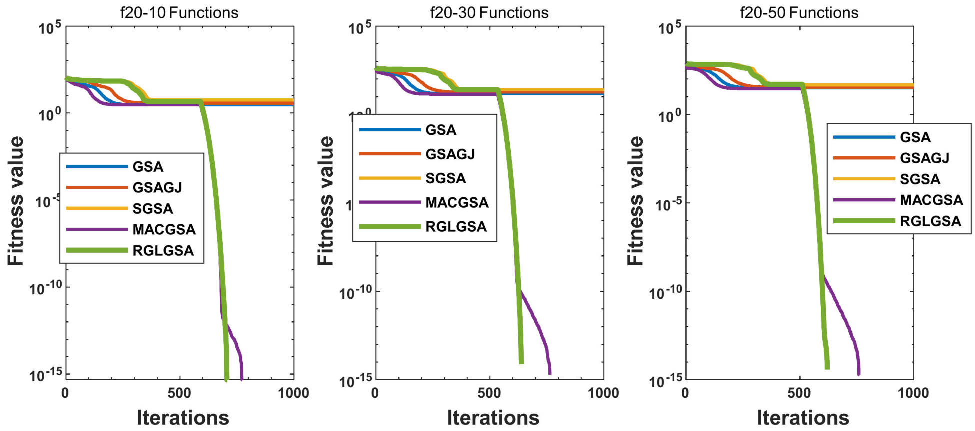
(**a**) (**b**) (**c**)

Figure 3. Convergence curves of function f18: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



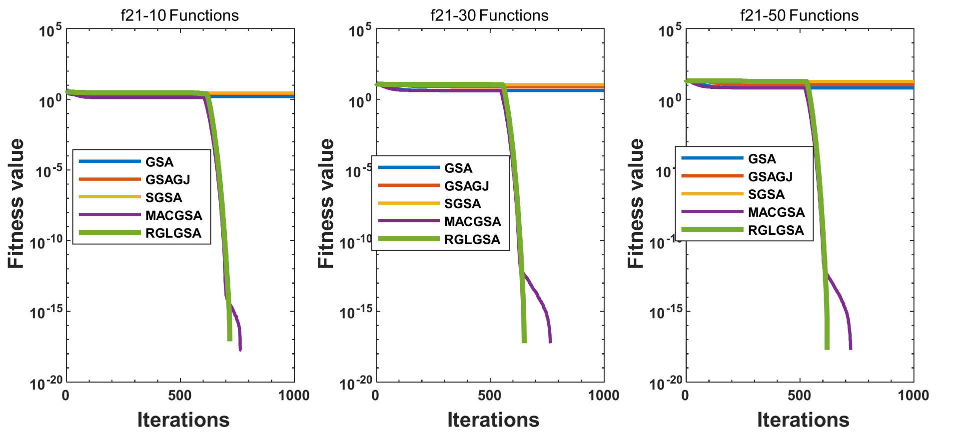
(**a**) (**b**) (**c**)

Figure 4. Convergence curves of function f19: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



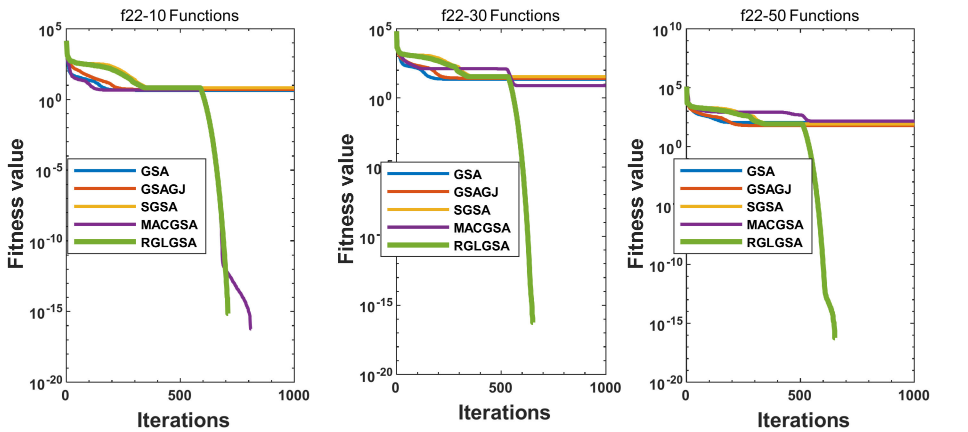
(**a**) (**b**) (**c**)

Figure 5. Convergence curves of function f20: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



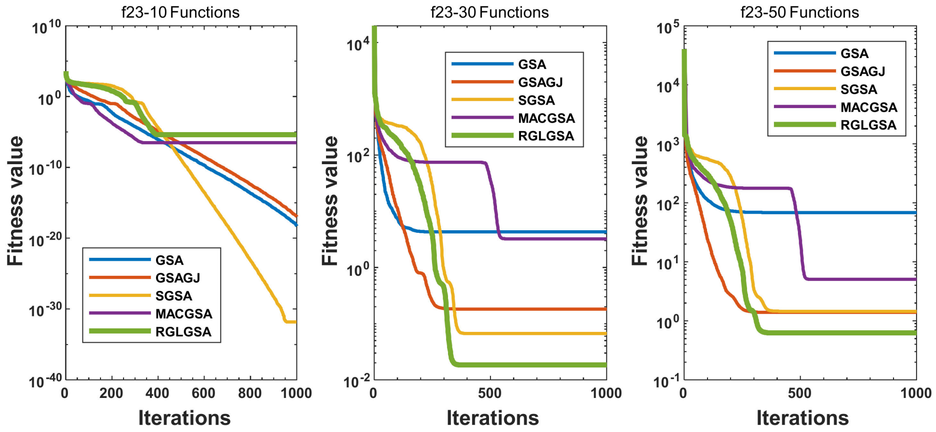
(**a**) (**b**) (**c**)

Figure 6. Convergence curves of function f21: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



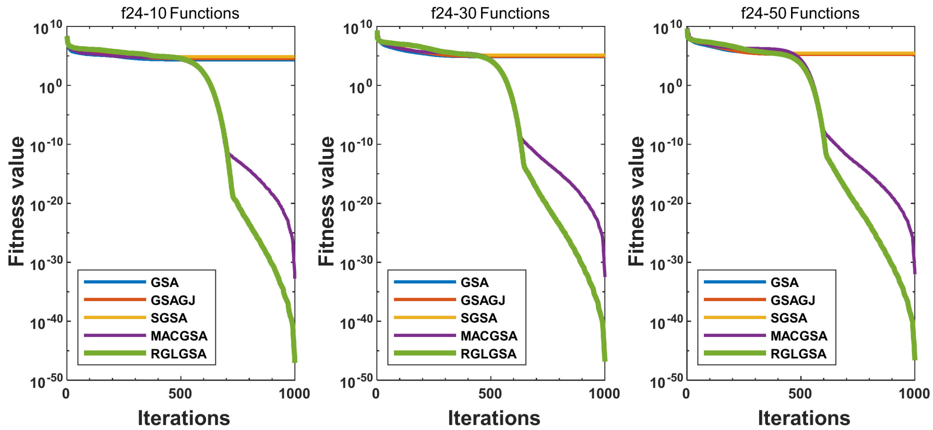
(**a**) (**b**) (**c**)

Figure 7. Convergence curves of function f22: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



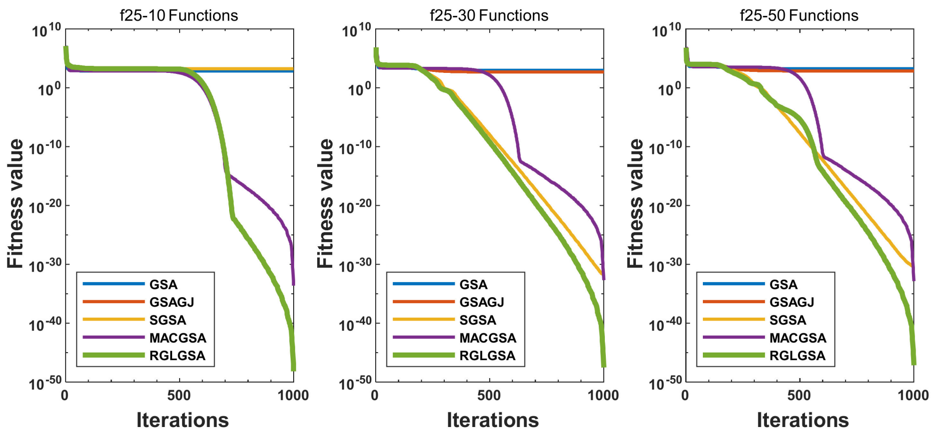
(**a**) (**b**) (**c**)

Figure 8. Convergence curves of function f23: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



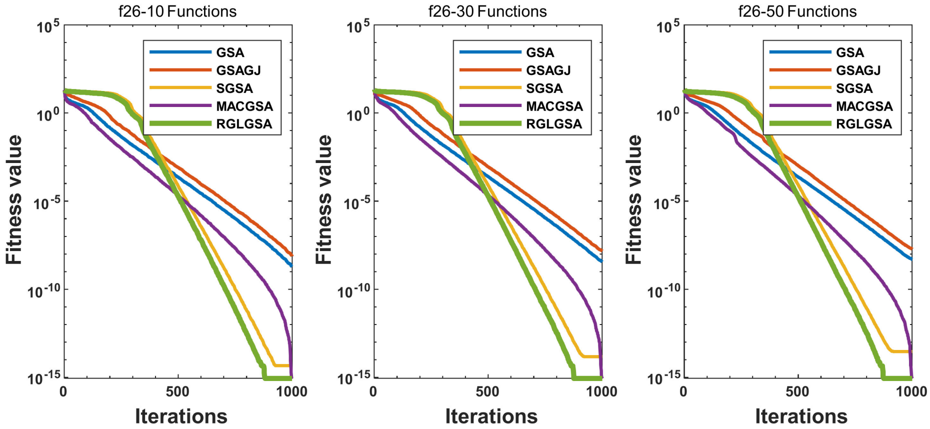
(**a**) (**b**) (**c**)

Figure 9. Convergence curves of function f24: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



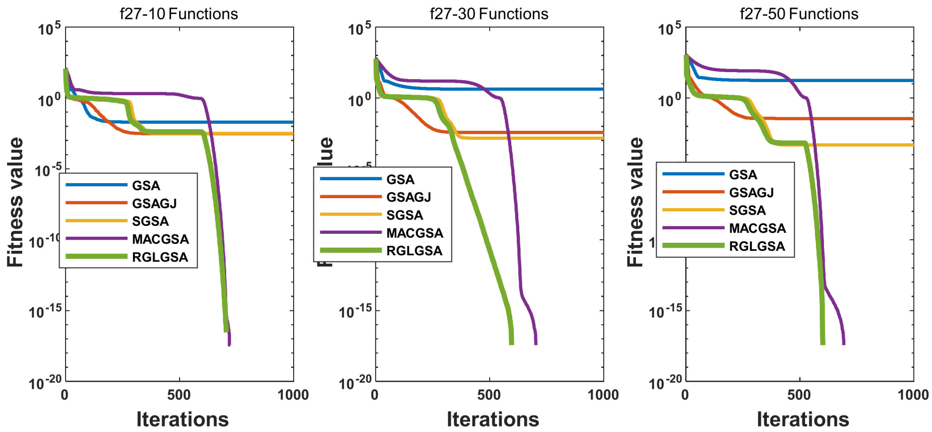
(**a**) (**b**) (**c**)

Figure 10. Convergence curves of function f25: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



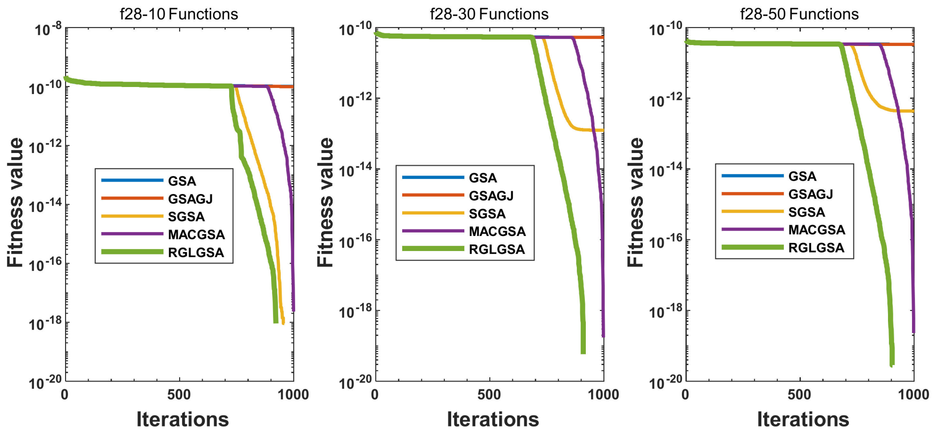
(**a**) (**b**) (**c**)

Figure 11. Convergence curves of function f26: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



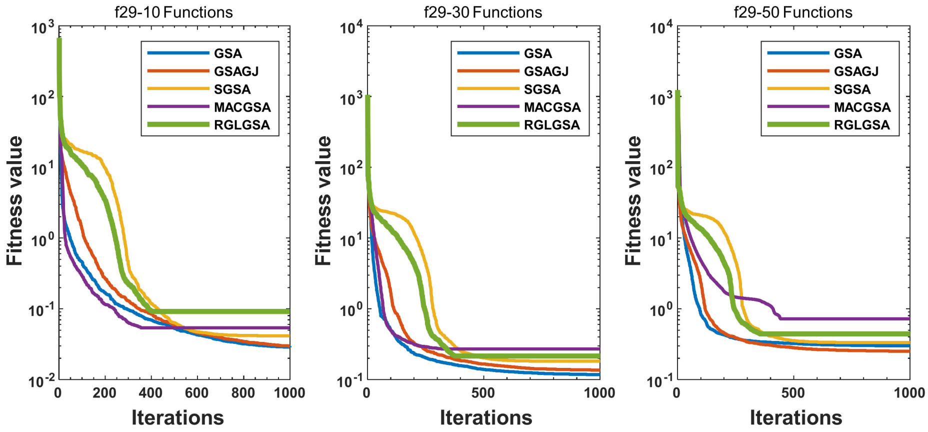
(**a**) (**b**) (**c**)

Figure 12. Convergence curves of function f27: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



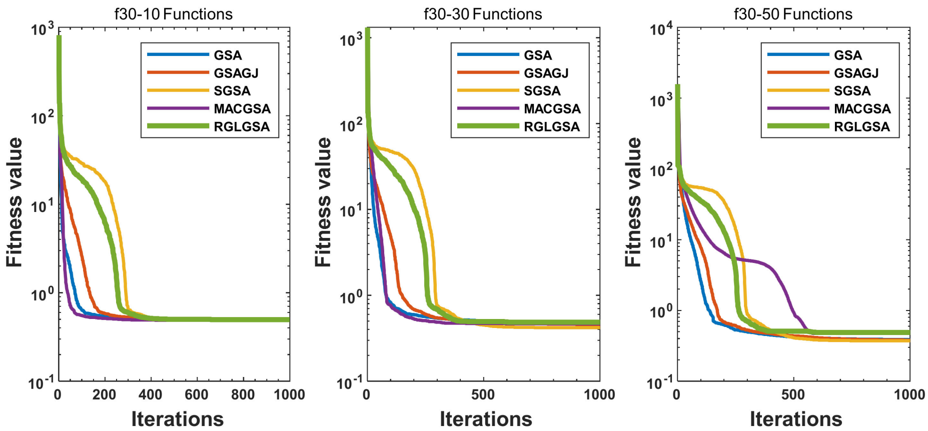
(**a**) (**b**) (**c**)

Figure 13. Convergence curves of function f28: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



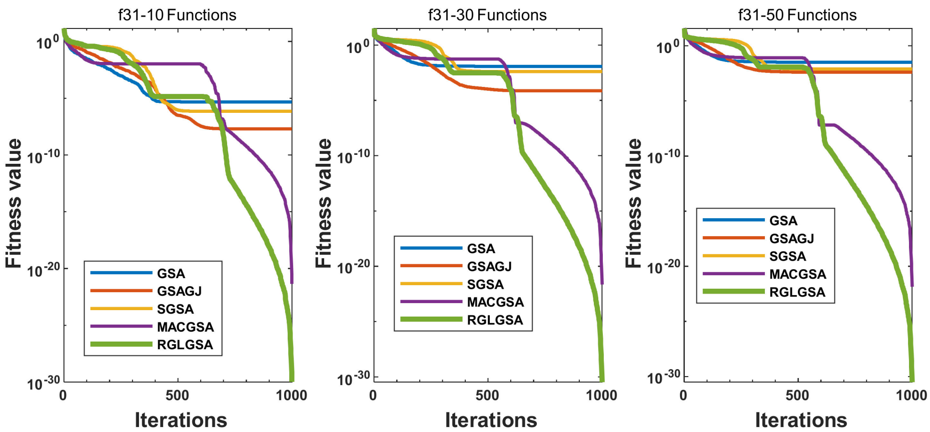
(**a**) (**b**) (**c**)

Figure 14. Convergence curves of function f29: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



(**a**) (**b**) (**c**)

Figure 15. Convergence curves of function f30: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.



(**a**) (**b**) (**c**)

Figure 16. Convergence curves of function f31: (**a**) 10 dimensions, (**b**) 30 dimensions, and (**c**) 50 dimensions.

For multimodal functions f19, f29 and f30, in the first 400 iterations, the convergence speed of the RLSGSA is roughly the same as those of the other four algorithms. Because of the characteristics of the functions and algorithms, after 400 iterations, the five algorithms converge to a local extremum, and the differences between the solutions obtained by the algorithms are not significant. For multimodal functions f20, f21, f27, and f28, GSA, GSAGJ, and SGSA converge to the local extreme value after a certain number of iterations and stop searching. The MACGSA and RGLGSA do not fall into the local extreme value but continue to evolve to find the globally optimal solution 0. However, the RGLGSA converges faster than the MACGSA. For multimodal function f22, in the first 500 iterations, the convergence speed of the RLSGSA is roughly the same as that of the other four algorithms. After 500 generations, the other four algorithms fall into local extrema, and the RLSGSA finds the global optimal solution 0 after about 650 generations. For multimodal function f31, in the first 300 iterations, the convergence rates of the five algorithms are similar. After 300 generations, the GSA, GSAGJ, and SGSA have stopped iterative evolutions, and the algorithms have fallen into locally optimal solutions. The MACGSA and RGLGSA continue to evolve after 600 generations, and finally, the MACGSA finds a better solution at a faster speed. For multimodal function f23, in 10 dimensions, the GSA and GSAGJ stop iterations after about 400 generations. After 400 generations, the convergence speed of the RGLGSA is faster than that of the SGSA and MACGSA. At 30 and 50 dimensions, the convergence accuracies of the five algorithms are not high. The GSAGJ has the fastest convergence speed, followed by the RGLGSA, but the convergence accuracy of the RGLGSA is higher than that of the GSAGJ. For multimodal function f26, the convergence speed of the MACGSA is higher before generation 500, the convergence speed of the RGLGSA is higher after generation 500, and the optimal value is obtained around generation 880. As a whole, the convergence speed of the RGLGSA is higher than those of the other comparison GSAs.

In general, compared with other GSA methods, the proposed adaptive GSA based on the adaptive strategies of population density, gravitational constant, and location update has a greatly improved optimization accuracy and stability, and performs well with respect to convergence.

6. Conclusions

In this paper, we proposed an improved GSA that is based on adaptive strategies. To address the shortcomings of the basic GSA, this algorithm introduces adaptive strategies based on crowd density, the gravitational constant, and location update into the basic universal GSA simultaneously. Moreover, it dynamically adjusts the distance between particles and the step size of the particle iteration, strengthens the information exchange between particles, and greatly increases the diversity of particles in the population. Therefore, it can effectively overcome the disadvantages of the basic GSA, which tends to fall into local extreme values. The simulation results show that the improved algorithm is superior to other classically improved GSAs in terms of search accuracy, convergence speed, stability, and other factors. It hence is an effective extension to the algorithm.

No Free Lunch Theory is a very important theorem in the field of optimization research, which reflects that no optimization algorithm can outperform other algorithms in average performance on all optimization problems. The improved algorithm in this paper also has this problem, and the experimental analysis cannot fully cover many optimization problems, which is convincing. The future research direction is mainly to optimize the overall performance of the algorithm, so that the performance of the improved algorithm is better than that of other algorithms on as many optimization problems as possible.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data is available upon reasonable request from the author.

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

References

1. Das, P.K.; Behera, H.S.; Jena, P.K.; Panigrahi, B.K. An intelligent multi-robot path planning in a dynamic environment using improved gravitational search algorithm. *Int. J. Autom. Comput.* **2021**, *18*, 1032–1044. [CrossRef]
2. Xu, H.; Jiang, S.; Zhang, A. Path planning for unmanned aerial vehicle using a Mix-strategy-based gravitational search algorithm. *IEEE Access.* **2021**, *9*, 57033–57045. [CrossRef]
3. Wang, Y.; Tan, Z.P.; Chen, Y.C. An adaptive gravitational search algorithm for multilevel image thresholding. *J. Supercomput.* **2021**, *77*, 10590–10607. [CrossRef]
4. Tan, Z.; Zhang, D. A fuzzy adaptive gravitational search algorithm for two-dimensional multilevel thresholding image segmentation. *J. Ambient Intell. Hum. Comput.* **2020**, *11*, 4983–4994. [CrossRef]
5. Alenazy, W.M.; Alqahtani, A.S. An automatic facial expression recognition system employing convolutional neural network with multi-strategy gravitational search algorithm. *IETE Tech. Rev.* **2022**, *39*, 72–85. [CrossRef]
6. García-Ródenas, R.; Linares, L.J.; López-Gómez, J.A. Memetic algorithms for training feedforward neural networks: An approach based on gravitational search algorithm. *Neural. Comput. Appl.* **2021**, *33*, 2561–2588. [CrossRef]
7. Nagra, A.A.; Alyas, T.; Hamid, M.; Tabassum, N.; Ahmad, A. Training a feedforward neural network using hybrid gravitational search algorithm with dynamic multiswarm particle swarm optimization. *BioMed Res. Int.* **2022**, *2022*, 2636515. [CrossRef] [PubMed]
8. Nobahari, H.; Alizad, M.; Nasrollahi, S. A nonlinear model predictive controller based on the gravitational search A nonlinear model predictive controller based on the gravitational search algorithm. *Optim. Control Appl. Meth.* **2021**, *42*, 1734–1761. [CrossRef]
9. Zhang, X.Q.; Wu, X.L.; Zhu, G.Y.; Lu, X.; Wang, K. A seasonal Arima model based on the gravitational search algorithm (GSA) for runoff prediction. *Water Supply* **2022**, *22*, 6959–6977. [CrossRef]
10. Singh, A.; Singh, N. Gravitational search algorithm-driven missing links prediction in social networks. *Concurr. Computat. Pract. Experience* **2022**, *34*, e6901. [CrossRef]
11. Younes, Z.; Alhamrouni, I.; Mekhilef, S.; Reyasudin, M. A memory-based gravitational search algorithm for solving economic dispatch problem in micro-grid. *Ain. Shams Eng. J.* **2021**, *12*, 1985–1994. [CrossRef]
12. Cao, C.W.; Zhang, Y.; Gu, X.S.; Li, D.; Li, J. An improved gravitational search algorithm to the hybrid flowshop with unrelated parallel machines scheduling problem. *Int. J. Prod. Res.* **2021**, *59*, 5592–5608. [CrossRef]
13. Habibi, M.; Broumandia, A.; Harounabadi, A. An Intelligent Traffic Light Scheduling Algorithm by using fuzzy logic and gravitational search algorithm and considering emergency vehicles. *Int. J. Nonlinear Anal. Appl.* **2021**, *11*, 475–482.
14. Thakur, A.S.; Biswas, T.; Kuila, P. Binary quantum-inspired gravitational search algorithm-based multi-criteria scheduling for multi-processor computing systems. *J. Supercomput.* **2021**, *77*, 796–817. [CrossRef]
15. He, J.F.; Wang, T.; Li, Y.J.; Deng, Y.L.; Wang, S.B. Dynamic chaotic gravitational search algorithm-based kinetic parameter estimation of hepatocellular carcinoma on F-18-FDG PET/CT. *BMC Med. Imaging* **2022**, *22*, 20. [CrossRef]
16. Enikeeva, L.V.; Potemkin, D.I.; Uskov, S.I.; Snytnikov, P.V.; Enikeev, M.R.; Gubaydullin, I.M. Gravitational search algorithm for determining the optimal kinetic parameters of propane pre-reforming reaction. *React. Kinet. Mech. Cat.* **2021**, *132*, 111–122. [CrossRef]
17. Ismail, A.M.; Mohamad, M.S.; Abdul Majid, H.A.; Abas, K.H.; Deris, S.; Zaki, N.; Mohd Hashim, S.Z.; Ibrahim, Z.; Remli, M.A. An improved hybrid of particle swarm optimization and the gravitational search algorithm to produce a kinetic parameter estimation of aspartate biochemical pathways. *Biosystems* **2017**, *162*, 81–89. [CrossRef]
18. Zhenkai, X. Modification and Application of Gravitational Search Algorithm. Master's Thesis, University of Shanghai for Science & Technology, Shanghai, China, 2014.
19. Yuhao, X. The Improvement and Application of Gravitational Search Algorithm. Master's Thesis, Henan University, Kaifeng, China, 2018.
20. Jing, Y.; Fang, L.; Peng, D. Research and simulation of the gravitational search algorithms with immunity. *Acta Armamentarii* **2012**, *33*, 1528–1533.
21. Xiaogang, L.I.U.; Ouyang, Z. Application of improved gravitational search algorithm in function optimization. *J. Shenyang Univ. Technol.* **2021**, *43*, 193–197.
22. Liu, Y.; Ma, L. Improved gravitational search algorithm based on free search differential evolution. *J. Syst. Eng. Electron.* **2013**, *24*, 690–698. [CrossRef]
23. Shehadeh, H.A. A hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) for global optimization. *Neural. Comput. Appl.* **2021**, *33*, 11739–11752. [CrossRef]
24. Huang, Y.; Wang, J.R.; Guo, F. Economic load dispatch using improved gravitational search algorithm. In Proceedings of the 2016 2nd ISPRS International Conference on Computer Vision in Remote Sensing (CVRS), Xiamen, China, 28–30 April 2015; Volume 9901.

25. Wang, R.P.; Su, F.; Hao, T.G.; Li, J.L. RGSA: A new improved gravitational search algorithm. In Proceedings of the 2016 3rd International Conference on Modelling, Simulation and Applied Mathematics (MSAM), Shanghai, China, 22–23 July 2018; Volume 160, pp. 202–207.
26. Rather, S.A.; Bala, P.S. Lévy flight and chaos theory-based gravitational search algorithm for mechanical and structural engineering design optimization. *Open Comput. Sci.* **2021**, *11*, 509–529. [CrossRef]
27. Chen, X.; Shuo, L.I.U. Improvement of universal gravitation search algorithm based on reverse evaluation mechanism. *J. Wuhan Polytech. Univ.* **2016**, *35*, 60–63.
28. Teng, F. Research on Improved Artificial Fish Swarm Algorithm and Its Application in Logistics Location Optimization. Ph.D. Thesis, Tianjin University, Tianjin, China, 2016.
29. Xu, Y.; Wang, S. Enhanced version of gravitational search algorithm: Weighted GSA. *Comput. Eng. Appl.* **2011**, *47*, 188–192.