mathematics

MDPI

Article

A Hybrid Arithmetic Optimization and Golden Sine Algorithm for Solving Industrial Engineering Design Problems

Qingxin Liu [1], Ni Li [2,3], Heming Jia [4,*], Qi Qi [1,*], Laith Abualigah [5,6] and Yuxiang Liu [7]

1 School of Computer Science and Technology, Hainan University, Haikou 570228, China; qxliu@hainanu.edu.cn
2 School of Mathematics and Statistics, Hainan Normal University, Haikou 571158, China; lini@hainnu.edu.cn
3 Key Laboratory of Data Science and Intelligence Education of Ministry of Education, Hainan Normal University, Haikou 571158, China
4 School of Information Engineering, Sanming University, Sanming 365004, China
5 Faculty of Computer Sciences and Informatics, Amman Arab University, Amman 11953, Jordan; laythdyabat@aau.edu.jo
6 School of Computer Science, Universiti Sains Malaysia, Gelugor 11800, Malaysia
7 College of Physics and Information Engineering, Fuzhou University, Fuzhou 350108, China; 211127132@fzu.edu.cn
* Correspondence: jiaheming@fjsmu.edu.cn (H.J.); qqi@hainanu.edu.cn (Q.Q.)

Abstract: Arithmetic Optimization Algorithm (AOA) is a physically inspired optimization algorithm that mimics arithmetic operators in mathematical calculation. Although the AOA has an acceptable exploration and exploitation ability, it also has some shortcomings such as low population diversity, premature convergence, and easy stagnation into local optimal solutions. The Golden Sine Algorithm (Gold-SA) has strong local searchability and fewer coefficients. To alleviate the above issues and improve the performance of AOA, in this paper, we present a hybrid AOA with Gold-SA called HAGSA for solving industrial engineering design problems. We divide the whole population into two subgroups and optimize them using AOA and Gold-SA during the searching process. By dividing these two subgroups, we can exchange and share profitable information and utilize their advantages to find a satisfactory global optimal solution. Furthermore, we used the Levy flight and proposed a new strategy called Brownian mutation to enhance the searchability of the hybrid algorithm. To evaluate the efficiency of the proposed work, HAGSA, we selected the CEC 2014 competition test suite as a benchmark function and compared HAGSA against other well-known algorithms. Moreover, five industrial engineering design problems were introduced to verify the ability of algorithms to solve real-world problems. The experimental results demonstrate that the proposed work HAGSA is significantly better than original AOA, Gold-SA, and other compared algorithms in terms of optimization accuracy and convergence speed.

Keywords: Meta-heuristics; arithmetic optimization algorithm; golden sine algorithm; hybrid optimization algorithm; industrial engineering design problem

MSC: 68T20

Citation: Liu, Q.; Li, N.; Jia, H.; Qi, Q.; Abualigah, L.; Liu, Y. A Hybrid Arithmetic Optimization and Golden Sine Algorithm for Solving Industrial Engineering Design Problems. *Mathematics* **2022**, *10*, 1567. https://doi.org/10.3390/math10091567

Academic Editor: Catalin Stoean

Received: 6 April 2022
Accepted: 29 April 2022
Published: 6 May 2022

1. Introduction

The main optimization process can be considered to obtain the best solution among all potential solutions according to the various NP-hard and engineering problems. Many real-world problems, such as image processing [1–3], engineering design [4–8], and job shop scheduling [9], can be expressed as optimization problems and solved using optimization techniques. In the past two decades, the complexity of real-world optimization problems has increased sharply. However, the traditional (mathematical) methods cannot find the optimal solution or near-optimal solution in many cases [10]. Therefore, many researchers have turned their attention to meta-heuristic algorithms (MAs). Unlike traditional techniques, MAs are flexible and reliable in solving complex optimization problems [11].

Over the past few decades, various MAs had been proposed according to natural phenomena, physical principles, biological behaviors, etc. [12]. MAs can be separated into three main categories (as shown in Figure 1): (1) swarm intelligence-based methods, (2) physics-based methods, and (3) evolution-based methods. The first kind of method mimics the biological entities in nature that have collaboration behavior to finish hunting, migrating, etc. [13]. Developed algorithms in this category are Whale Optimization Algorithm (WOA) [14], Particle Swarm Optimization (PSO) [15], Grey Wolf Optimizer (GWO) [16], Salp Swarm Algorithm (SSA) [17], Ant Lion Optimization (ALO) [18], Moth Flame Optimization (MFO) [19], Slime Mould Algorithm (SMA) [20], Harris Hawks Optimization (HHO) [21], Reptile Search Algorithm (RSA) [22], and Aquila Optimizer (AO) [23]. The second type of method mainly simulates the physical phenomena of the universe and methods designed based on these laws are Multi-Verse Optimizer (MVO) [24], Sine Cosine Algorithm (SCA) [25], Arithmetic Optimization Algorithm (AOA) [26], Golden Sine Algorithm (Gold-SA) [27], Henry Gas Solubility Optimization (HGSO) [28], Gravity Search Algorithm (GSA) [29], Atom Search Optimization (ASO) [30], and Equilibrium Optimizer (EO) [31]. The evolution-based methods stem from the biological evolution process in nature. Some of the well-known algorithms developed by this behavior are Genetic Algorithm (GA) [32], Bio-geography-Based Optimizer (BBO) [33], Differential Evolution (DE) [34], and Evolution Strategy (ES) [35]. However, considering the No-Free-Lunch (NFL) theorem [36], no specific optimization algorithm can solve all real-world problems, which motivates us to design more efficient methods to solve them well.

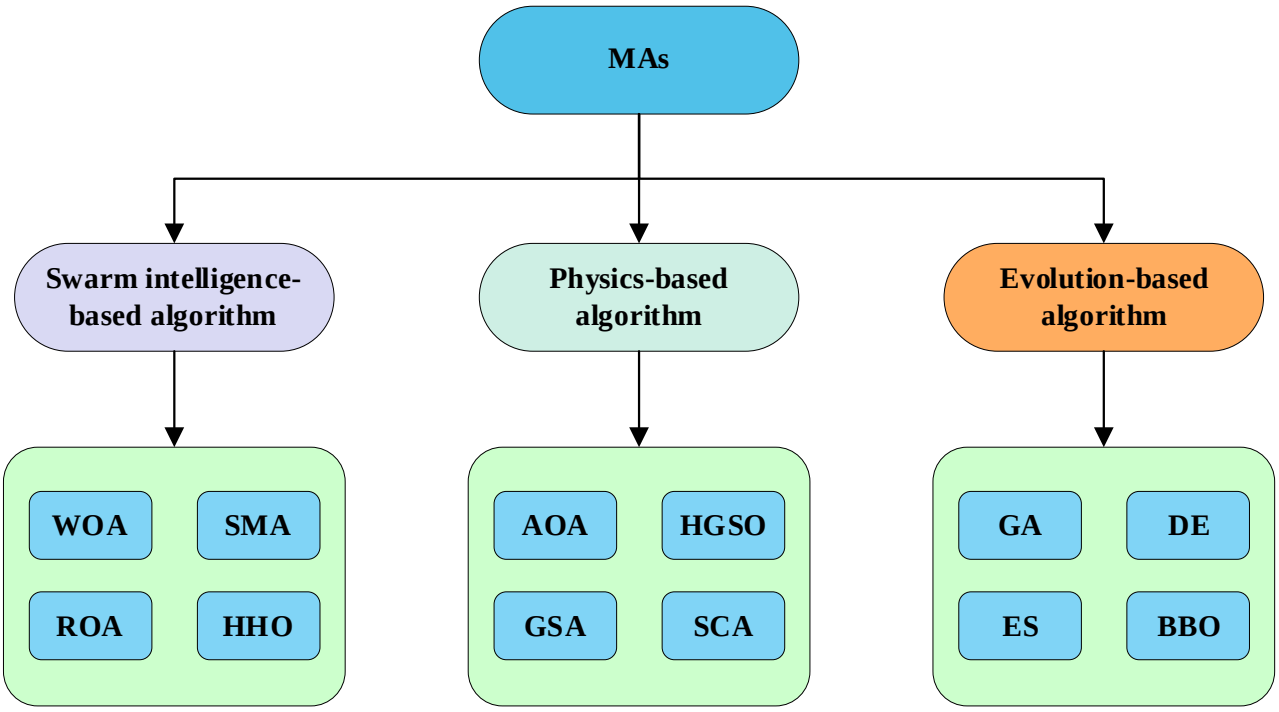

Figure 1. Classification of MAs.

The Arithmetic Optimization Algorithm (AOA) [26] is a physics-based and gradient-free method proposed by Abualigah et al. in 2021. It originated from the commonly used mathematical operators including Addition (+), Subtraction (−), Multiplication (×), and Division (÷). This approach integrates these four operators to realize different search mechanisms (exploration and exploitation) in the search space. Specifically, AOA uses the high distribution characteristics of (× and ÷) operators to realize the exploration approach. In the same way, the (+ and −) operators are used to obtain the high-dense results (exploitation approach). However, some researches denote that the original AOA has some defects, such as it easily suffering from a local optimal and slow convergence

speed. Therefore, many variant versions of AOA were proposed to improve its searchability. For example, Azizi et al. [37] proposed an improved AOA based on Levy flight to determine the steel structure's optimal fuzzy controller parameters. Agushaka et al. [38] proposed an improved version of AOA called nAOA, which integrated the high-density values and beta distribution to enhance searchability. An Adaptive AOA, called APAOA, was proposed by Wang et al. [39]. In the APAOA, the parallel communication strategy was used to balance the exploration and exploitation ability of the original AOA. Another improved AOA that utilized a hybrid mechanism, named DAOA, was proposed by Abualigah et al. [40]. In DAOA, the differential evolution technique was integrated to enhance the local search ability of AOA, and to help it to jump out of the local optimal solution. Elkasem et al. [41] presented an eagle strategy AOA called ESAOA. In this work, the eagle strategy is used to avoid premature convergence and increase the population's efficacy to obtain the optimal solution. Sharma et al. [42] introduced an opposition-based AOA namely OBAOA for identifying the parameters of PEMFCs. The opposition-based learning strategy is used to promote the algorithm to find the high-precision solution and improve the convergence rate. Abbassi et al. [43] developed an improved AOA to determine the solar cell parameters. In this work, the new operator called narrowed exploitation was used to narrow the search space and focus on the potential area to find the optimal or near-optimal solutions. Zhang et al. [44] proposed an improved AOA called IAO, which integrated the chaotic theory. The chaotic theory improves the algorithm to escape the optimal solution with a suitable convergence speed. Moreover, the IAO was used to optimize the weight of neural network.

Given the above discussion, some of the variants of AOA have strong searchability, but they cannot converge to the optimal solution at an appropriate time, i.e., they still easily fall into the local optimal solution. Furthermore, by considering the NFL theorem and increasingly complex real-world problems, the development of new and improved versions of MAs is ongoing. In general, a single optimizer also exposes some shortcomings; for example, it neglects to share useful information between populations, which may cause the algorithm to have insufficient search capability. Therefore, many researchers utilized the characteristic of two Mas, i.e., designing a hybrid algorithm to improve performance and applying it to solve complex real-world optimization problems. Unlike the single algorithm, the hybrid algorithm alleviates these shortcomings and increases diversity, and shares more helpful information within the population. Thus, the hybrid algorithm has more powerful searchability than the single algorithm. Gold-SA is a physics-based technique with a good exploitation ability to find the near-optimal solution. Furthermore, Gold-SA also has fewer parameters and is easy to program. Motivated by these considerations, in this paper, we propose an improved hybrid version of AOA called HAGSA that combines both AOA and Gold-SA. The proposed method uses Gold-SA to increase the population diversity and share more useful information between search agents. At the same time, Levy flight and a new strategy called Brownian mutation are used to enhance the exploration and exploitation capability of hybrid algorithms, respectively. To evaluate the effectiveness of the proposed method, we selected the CEC 2014 competition test suite as the benchmark function and compared the results with seven well-known methods, including AOA and Gold-SA. In addition, five classical engineering design problems, including the car side crash design problem, pressure vessel design problem, tension spring design problem, speed reducer design problem, and cantilever beam design problem, were also used to evaluate HAGSA's ability to solve real-world problems. Experimental results demonstrate that the proposed work can provide complete results and achieve a faster convergence speed compared to other optimizers. The main contributions of this paper are as follows:

- We propose a new hybrid algorithm based on the Arithmetic Optimization Algorithm and Golden Sine Algorithm (HAGSA).
- Levy flight and a new mechanism called Brownian mutation are carried out to enhance the exploration and exploitation ability of the hybrid algorithm.

- The performance of the proposed work is assessed on the CEC 2014 competition test suite and five classical engineering design problems.
- Several well-known MAs are compared with the proposed method.
- Experimental results indicate that HAGSA has more reliable performance than that of other well-known algorithms.

The remainder of this paper is structured as follows: Section 2 briefly illustrates the concepts of AOA and Gold-SA. Section 3 describes Levy flight, Brownian mutation, and the details of HAGSA. Section 4 presents and analyzes the experimental results of the proposed work. Finally, this paper's conclusion and potential research directions are discussed in Section 5.

2. Preliminaries

This section introduces the inspiration and mathematical model of the original AOA and Gold-SA, in turn.

2.1. Arithmetic Optimization Algorithm (AOA)

The theory of AOA is described in this section. The main inspiration of AOA originates from the use of arithmetic operators such as Addition (*A*), Subtraction (*S*), Multiplication (*M*), and Division (*D*) to solve optimization problems [33]. In the following subsections, we discuss the different influences of these operators on optimization problems and the search method of AOA, as shown in Figure 2.

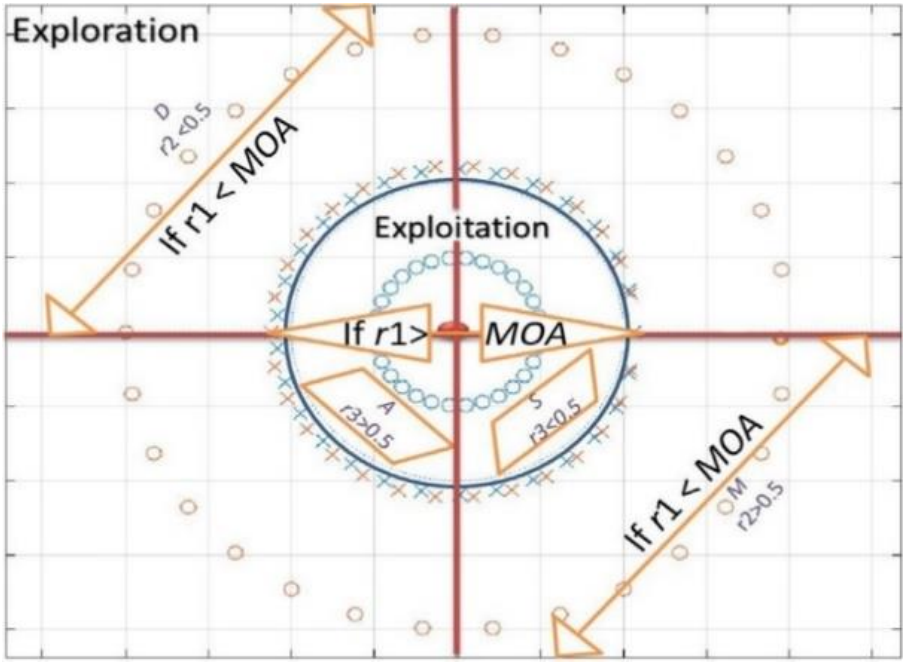

Figure 2. The different search phases of AOA.

2.1.1. Initialization Phase

Like other meta-heuristic optimization algorithms, AOA is based on population behavior. The set of a population *X* containing *N* search agents is illustrated in Equation (1). In the matrix, each row indicates a search agent [33].

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \cdots & x_{N-1,j} & \cdots & x_{N-1,n} \\ x_{N,1} & \cdots & x_{N,j} & x_{N,n-1} & x_{N,n} \end{bmatrix} \quad (1)$$

After generating the population, the fitness of each search agent is computed, and the best one will be determined. Next, AOA decides to perform exploration or exploitation through the Math Optimizer Accelerated (*MOA*) value, which is defined as follows:

$$MOA(t) = Min + t \times \left(\frac{Max - Min}{T} \right) \quad (2)$$

where $MOA(t)$ indicates the value of *MOA* at the *t-th* iteration. *Min* and *Max* denote the minimum and maximum values of the accelerated function, respectively. t denotes the current iteration, and T denotes the maximum iteration. The search agent performs the exploration phase when $r_1 > MOA$, otherwise the exploitation phase will be executed.

2.1.2. Exploration Phase

In this section, the exploration phase of AOA is described. According to the main inspiration, the Division (*D*) and Multiplication (*M*) operators are introduced to achieve high distributed values or decisions [33]. The Division and Multiplication operators can be mathematically described as follows:

$$X_{i,j}(t+1) = \begin{cases} X_{best,j}(t) \times MOP \times ((UB_j - LB_j) \times \mu + LB_j), \ r_2 < 0.5 \\ X_{best,j}(t) \div (MOP + \varepsilon) \times ((UB_j - LB_j) \times \mu + LB_j), \ \text{otherwise} \end{cases} \quad (3)$$

where $X_{i,j}(t+1)$ denotes the jth position of the ith solution in the next iteration. $X_{best,j}(t)$ denotes the best solution obtained so far in the jth position. LB_j and UB_j denote the lower and upper boundaries, respectively, of the search space at the jth dimension. ε is a small integer number, and r_2 denotes the random value between 0 and 1. $\mu = 0.5$, which represents the control function. Moreover, the Math Optimizer can be calculated as follows:

$$MOP(t) = 1 - \frac{t^{1/\alpha}}{T^{1/\alpha}} \quad (4)$$

where α = 0.5 denotes the dynamic parameter, which determines the accuracy of the exploitation phase throughout iterations.

2.1.3. Exploitation Phase

In this section, we discuss the exploitation phase of AOA. In contrast to the *D* and *M* operator, AOA utilizes the Addition (*A*), and Subtraction (*S*) operators to derive high density solutions because (*S* and *A*) can easily approach the target region due to their low dispersion [33]. The mathematical formula can be described as follows:

$$X_{i,j}(t+1) = \begin{cases} X_{best,j}(t) - MOP \times ((UB_j - LB_j) \times \mu + LB_j), \ r_3 < 0.5 \\ X_{best,j}(t) + MOP \times ((UB_j - LB_j) \times \mu + LB_j), \ \text{otherwise} \end{cases} \quad (5)$$

where r_3 denotes a random value in the range 0 to 1.

The pseudo-code of AOA is illustrated in Algorithm 1.

Algorithm 1 pseudo-code of AOA [33]

Input: The parameter of AOA such as control function (μ), dynamic parameter (α), number of search agents (N), and maximum iteration (T)
Output: the best solution
Initialize the search agent randomly.
While ($t < T$) **do**
    Check if any search agent goes beyond the search space and amend it.
    Calculate fitness for the given search agent.
    Update the *MOA* and *MOP* using Equations (2) and (4), respectively.
    For $i = 1$ to N **do**
        For $j = 1$ to D **do**
            Update the random value r_1, r_2, r_3.
            If $r_1 > MOA$ **then**
                If $r_2 > 0.5$ **then**
                    Update position by Division ($\div$) operator in Equation (3).
                Else
                    Update position by Multiplication ($\times$) operator in Equation (3).
                End if
            Else
                If $r_3 > 0.5$ **then**
                    Update position by Addition (+) operator in Equation (5).
                Else
                    Update position by Subtraction ($-$) operator in Equation (5).
                End if
            End if
        End for
    End for
    $t = t + 1$.
End while

2.2. Golden Sine Algorithm (Gold-SA)

This section introduces the basic theory of the Golden Sine Algorithm (Gold-SA). The inspiration of Gold-SA is a sine function in mathematics, and the individuals explore the approximate optimal solution in the search space according to the golden ratio [27]. The range of the sine function is $[-1, 1]$, with period 2π. When the value of x_1 changes, the corresponding variable y_1 also changes. Combining the sine function and golden ratio helps to continuously reduce the search space and search in regions where the optimal values are more likely to be generated, thereby improving the convergence speed [27]. The calculation formula is as follows:

$$X_{i,j}(t+1) = X_{i,j}(t) \times |\sin(p_1)| - p_2 \times \sin(p_1) \times \left| d_1 \times X_{best,j}(t) - d_2 \times X_{i,j}(t) \right| \quad (6)$$

where p_1 is the random value between $[0, 2\pi]$, and p_2 is the random between $[0, \pi]$, and d_1 and d_2 are the coefficient factors, which are obtained by the following equation:

$$d_1 = a \times \tau + b \times (1 - \tau) \quad (7)$$

$$d_2 = a \times (1 - \tau) + b \times \tau \quad (8)$$

where a and b are the initial values, which are $-\pi$ and π. τ denotes the golden ratio, which is $(\sqrt{5} - 1)/2$. The pseudo-code of Gold-SA is shown in Algorithm 2.

Algorithm 2 pseudo-code of Gold-SA [27]

Input: The parameter of Gold-SA, such as the number of search agents (N), and maximum iteration (T).
Output: The best solution
Initialize the search agent randomly.
While ($t < T$) **do**
    Check if any search agent goes beyond the search space and amend it.
    Calculate fitness for the given search agent.
    For $i = 1$ to N **do**
        Update the random value p_1 and p_2, respectively.
        For $j = 1$ to D **do**
            Update position of search agent by the Equation (6).
        End for
    End for
    $t = t + 1$.
End while

3. The Proposed Algorithm

In this section, we describe the proposed method. First, Levy flight is presented. Second, we propose a new strategy called Brownian mutation. Then, the details of the proposed work, HAGSA, are discussed and analyzed.

3.1. Levy Flight

Numerous studies reveal that the flight trajectories of many flying animals are consistent with characteristics typical of Levy flight. Levy flight is a class of non-Gaussian random walk that follows the Levy distribution [41,42]. It performs occasional long-distance walking with frequent short-distance steps, as shown in Figure 3. The mathematical formula for Levy flight is as follows:

$$Levy = 0.01 \times \frac{r_4 \times \sigma}{|r_5|^{\frac{1}{\beta}}} \tag{9}$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{10}$$

where r_4 and r_5 are random values between [0, 1], and β is a constant equal to 1.5.

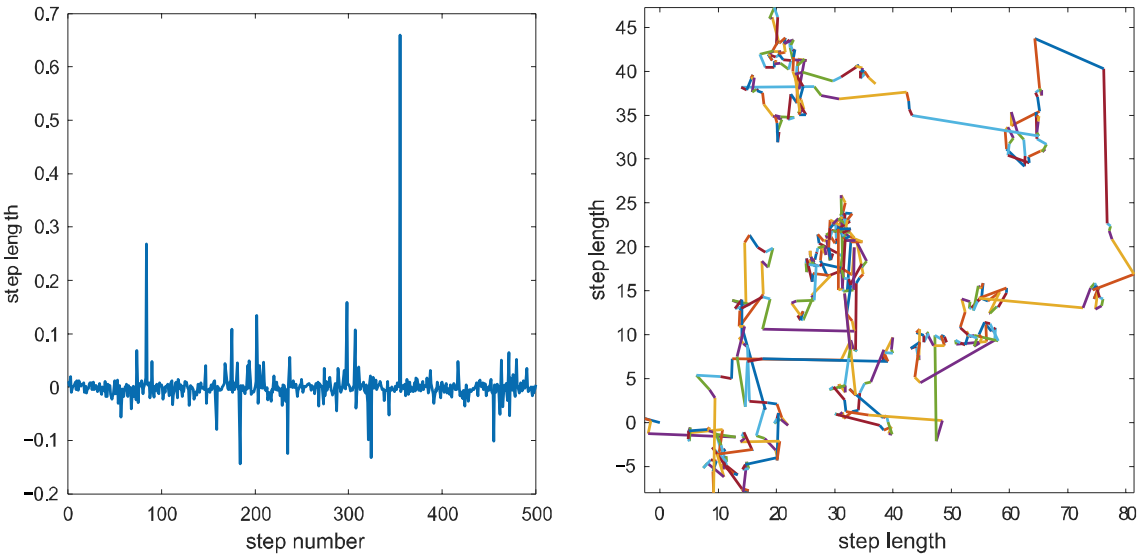

Figure 3. Levy distribution and 2D Levy trajectory.

3.2. Brownian Mutation

This paper proposes a Brownian mutation mechanism based on the mutation operator and Brownian motion. In 1995, differential evolution (DE) was proposed by Storn et al. [34], which was inspired by the mutation, crossover, and selection mechanisms in nature. Thus, DE obtains the optimal or near-optimal solution according to these operators. However, the crossover and mutation operators generate only one candidate solution in each iteration, limiting the population diversity and searchability of MAs [8]. Brownian motion (BM) is a stochastic process with a step size derived from a probability function defined by a normal distribution with $\mu = 0$ and $\sigma^2 = 1$ [43]. The formula of BM is listed as follows:

$$f_B(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{11}$$

where x indicates a point following this motion, and the distribution and 2D trajectory of BM as shown in Figure 4. We can see that BM's trajectory can explore distant areas of the neighborhood, which shows more efficiency than a uniform random search in the search space. Therefore, considering the high performance of Brownian motion and the limitation of the mutation operator, we propose Brownian mutation, which generates two trail vectors with the Brownian motion strategy. This method generates two candidate solutions V_1 and V_2 of the i-th search agent in parallel through Equations (12) and (13), respectively.

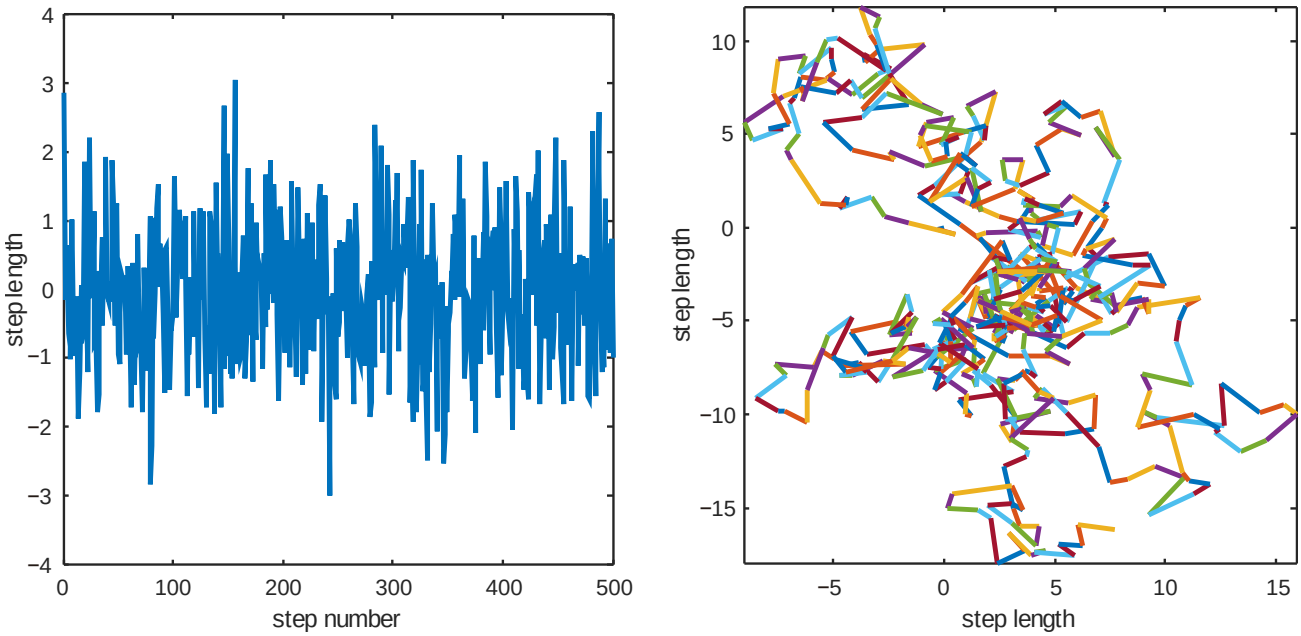

Figure 4. Brownian distribution and 2D Brownian trajectory.

The first mutation candidate solution V_1 is calculated as follows:

$$V_{1,j} = \begin{cases} X_{r6}(t) + Brownian \times (X_{r7}(t) - X_{r8}(t)), & \text{if } rand() < mr_1 \\ X_{i,j}, & \text{otherwise} \end{cases} \tag{12}$$

where r_6, r_7, and r_8 denote random values between 0 and 1. mr_1 is the mutation rate, and its value is 0.3. *Brownian* indicates the Brownian motion.

The second mutation candidate solution V_2 is calculated as follows:

$$V_{2,j} = \begin{cases} X_{best}(t) + Brownian \times (X_{r9}(t) - X_{r10}(t)), & \text{if } rand() < mr_2 \\ X_{i,j}, & \text{otherwise} \end{cases} \tag{13}$$

where r_9, r_{10}, and r_{11} denote random values between 0 and 1. mr_2 is the mutation rate equal to 0.5.

When two candidate solutions V_1 and V_2 are generated, they are first modified according to the lower and upper boundaries. Then, the best candidate solution V_{best} is selected using Equation (14) (lowest fitness as the criterion).

$$V_{best} = \begin{cases} V_1, \text{ if } f(V_1) < f(V_2) \\ V_2, \text{ otherwise} \end{cases} \tag{14}$$

Afterward, the best solution between V_{best} and X_i is selected as the ith search agent in the next iteration. The following equation describes this behavior:

$$X_i = \begin{cases} V_{best}, & \text{if } f(V_{best}) < f(X_i) \\ X_i, & \text{otherwise} \end{cases} \tag{15}$$

3.3. The Details of HAGSA

As mentioned above, single MAs have low diversity and cannot share useful information within the population. Moreover, the original AOA has shortcomings, such as easily stagnating into optimal local solutions and slow convergence speed. The Gold-SA has strong local searchability in the search space. Thus, to overcome the disadvantages of the original AOA and take full advantage of the benefits of Gold-SA, in this paper, we present a hybrid algorithm based on the AOA and Gold-SA, namely HAGSA. We divided the whole population into two subgroups, Group A and Group B, and optimized them using AOA and Gold-SA, respectively. Integrating both AOA and Gold-SA can increase population diversity and all the exchange pf useful search information between search agents. This operation aims to enable search agents to find the valuable solution in the search space based on two MAs (AOA and Gold-SA) in less time and increase the diversity throughout the entire iterations. Furthermore, to enhance the searchability of the hybrid algorithm, it was integrated with Levy flight and Brownian mutation. Levy flight can improve the hybrid algorithm's exploration ability, allowing search agents to explore more potential regions in the search space. Thus, the improved exploration phase can be calculated by Equation (16). Furthermore, the Brownian mutation is used to strengthen the exploitation capability of the hybrid algorithm and help the individuals escape the local optimal solution.

$$X_{i,j}(t+1) = \begin{cases} X_{best,j}(t) \times Levy(j) \times MOP \times ((UB_j - LB_j) \times \mu + LB_j), \ r_2 < 0.5 \\ X_{best,j}(t) \times Levy(j) \div (MOP + \varepsilon) \times ((UB_j - LB_j) \times \mu + LB_j), \text{ otherwise} \end{cases} \tag{16}$$

The pseudo-code of HAGSA is expressed in Algorithm 3, and the flowchart of the proposed work is shown in Figure 5.

3.4. Computational Complexity Analysis

In the initialization phase, HAGSA produces the search agents randomly in the search space, so the computational complexity of this phase is O($N \times D$), where N denotes the number of population and D denotes the dimension size. Afterward, HAGSA evaluates each individual's fitness during the whole iteration with the complexity O($T \times N \times D$), where T indicates the number of iterations. Finally, we used AOA, Gold-SA, Levy flight, and Brownian mutation to obtain the best solution. Thus, the computational complexities of these phases are O($3 \times T \times N \times D$). In summary, the total computational complexity of HAGSA is O($T \times N \times D$).

Algorithm 3 pseudo-code of HAGSA

Input: The parameter such as control function (μ), dynamic parameter (α), number of search agent (N), and maximum iteration (T).
Output: best solution
Initialize the search agent randomly.
While ($t < T$) **do**
    Check if any search agent goes beyond the search space and amend it.
    Calculate fitness for the given search agent.
    Update the *MOA* and *MOP* using Equations (2) and (4), respectively.
    For $i = 1$ to N **do**
        For $j = 1$ to D **do**
            Update the random value r_1, r_2, r_3.
            If $i < N/2$ **then**
                If $r_1 > MOA$ **then**
                    If $r_2 > 0.5$ **then**
                        Update position by Division ($\div$) operator in Equation (16).
                    Else
                        Update position by Multiplication ($\times$) operator in Equation (16).
                    End if
                Else
                    If $r_3 > 0.5$ **then**
                        Update position by Addition (+) operator in Equation (5).
                    Else
                        Update position by Subtraction ($-$) operator in Equation (5).
                    End if
                End if
            Else
                Update position by Gold-SA operator in Equation (6).
            End if
            Generate candidate solution V_1 and V_2 by Equations (12) and (13).
            Check if V_1 and V_2 goes beyond the search space and amend it.
            Choose the best solution as V_{best} with the lower fitness from V_1 and V_2.
            If $f(V_{best}) < f(X_i)$ **then**
                $X_i = V_i$.
            End if
        End for
    End for
    $t = t + 1$.
End while

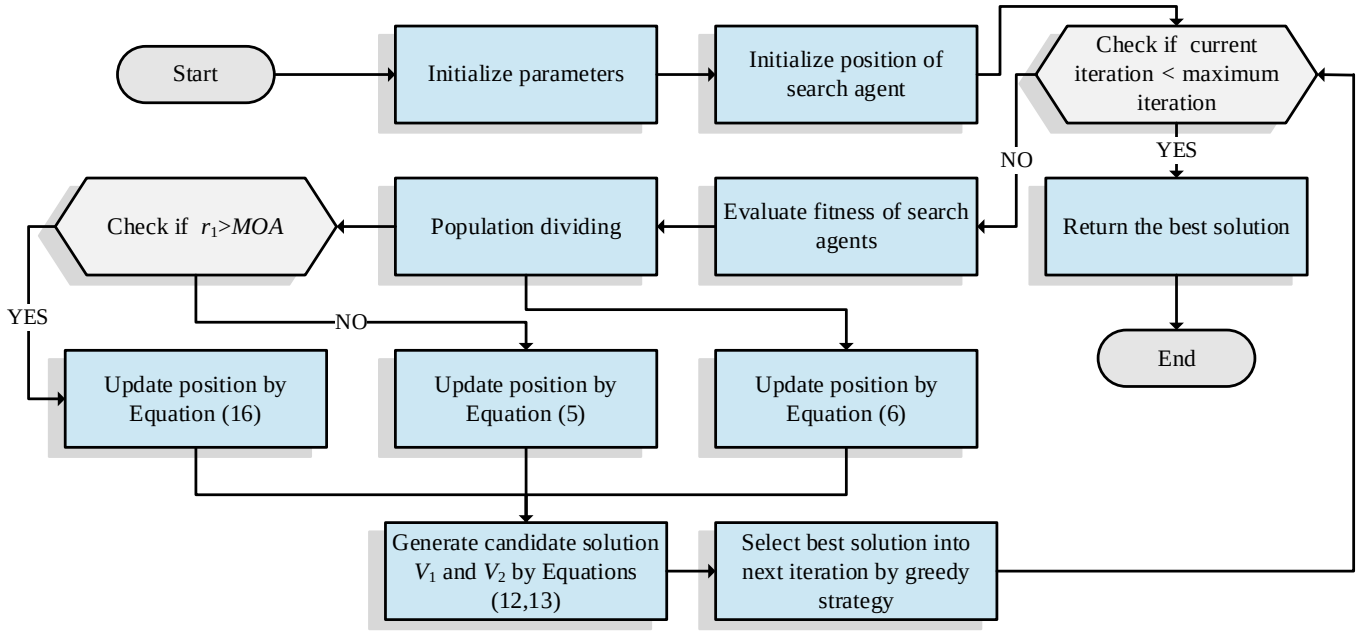

Figure 5. Flowchart of HAGSA.

4. Experimental Results and Discussion

This section evaluates the effectiveness of the proposed HAGSA algorithm using the CEC 2014 competition test suite and five industrial engineering design problems. First, the benchmark functions and experimental setup are described. Next, the statistical results of the CEC 2014 benchmark functions are analyzed and discussed. Finally, the five industrial engineering design problems are used to prove the advantages of HAGSA.

4.1. Definition of CEC 2014 Benchmark Functions

To validate the searchability of the proposed HAGSA, we considered the CEC 2014 competition test suite as a benchmark function to evaluate the performance of HAGSA and its peers, which include 30 extremely complex functions [44]. The details of the benchmark functions are listed in Table 1, where f_{min} denotes the theoretical optimal fitness. According to their characteristics, the CEC 2014 test suite can be categorized into four classes. C01–C03 are unimodal functions with only one global optimum without any local optima, and are suitable for evaluating algorithms' exploitation capability. C04–C15 are multimodal functions with only one global optimal value with many local optimal values, and can evaluate algorithms' exploration and local minima avoidance ability. C16–C22 are hybrid functions, including both unimodal and multimodal functions, and can simultaneously examine the exploration and exploitation capability of algorithms. C23–C30 are composition functions that maintain continuity around the local and global optima. All these functions are rotated and shifted, so their complexity increases dramatically. Figure 6 provides a 2D visualization of some functions of the CEC 2014 test suite to understand its characteristics.

Table 1. CEC 2014 benchmark functions.

Function Types	No.	Name of the Function	D	Range	$f_{\min}$
Unimodal	C01	Rotated High Conditioned Elliptic Function	30	[−100, 100]	100
	C02	Rotated Bent Cigar Function	30	[−100, 100]	200
	C03	Rotated Discus Function	30	[−100, 100]	300

Table 1. *Cont.*

Function Types	No.	Name of the Function	D	Range	f_{min}
Multimodal	C04	Shifted and Rotated Rosenbrock Function	30	[−100, 100]	400
	C05	Shifted and Rotated Ackley Function	30	[−100, 100]	500
	C06	Shifted and Rotated Weierstrass Function	30	[−100, 100]	600
	C07	Shifted and Rotated Griewank Function	30	[−100, 100]	700
	C08	Shifted Rastrigin Function	30	[−100, 100]	800
	C09	Shifted and Rotated Rastrigin Function	30	[−100, 100]	900
	C10	Shifted Schwefel Function	30	[−100, 100]	1000
	C11	Shifted and Rotated Schwefel Function	30	[−100, 100]	1100
	C12	Shifted and Rotated Katsuura Function	30	[−100, 100]	1200
	C13	Shifted and Rotated HappyCat Function	30	[−100, 100]	1300
	C14	Shifted and Rotated HGBat Function	30	[−100, 100]	1400
	C15	Shifted and Rotated Expanded Griewank plus Rosenbrock Function	30	[−100, 100]	1500
Hybrid	C16	Shifted and Rotated Expanded Scaffer F6 Function	30	[−100, 100]	1600
	C17	Hybrid Function 1(N = 3)	30	[−100, 100]	1700
	C18	Hybrid Function 2(N = 3)	30	[−100, 100]	1800
	C19	Hybrid Function 3(N = 4)	30	[−100, 100]	1900
	C20	Hybrid Function 4(N = 4)	30	[−100, 100]	2000
	C21	Hybrid Function 5(N = 5)	30	[−100, 100]	2100
	C22	Hybrid Function 6(N = 5)	30	[−100, 100]	2200
Composition	C23	Composition Function 1(N = 5)	30	[−100, 100]	2300
	C24	Composition Function 2(N = 3)	30	[−100, 100]	2400
	C25	Composition Function 3(N = 3)	30	[−100, 100]	2500
	C26	Composition Function 4(N = 5)	30	[−100, 100]	2600
	C27	Composition Function 5(N = 5)	30	[−100, 100]	2700
	C28	Composition Function 6(N = 5)	30	[−100, 100]	2800
	C29	Composition Function 7(N = 3)	30	[−100, 100]	2900
	C30	Composition Function 8(N = 3)	30	[−100, 100]	3000

4.2. Experimental Setup

As stated above, the CEC 2014 test suite was utilized to evaluate HAGSA's optimization performance. To demonstrate the validity of the experimental results, the proposed algorithm HAGSA was compared with the basic AOA [26], Gold-SA [27], Remora Optimization Algorithm (ROA) [45], Aquila Optimizer (AO) [23], Sine Cosine Algorithm (SCA) [25], Whale Optimization Algorithm (WOA) [14], Flower Pollination Algorithm (FPA) [46], Differential Evolution (DE) [8], and Genetic Algorithm (GA) [47]. We set the maximum iteration T = 500, population size N = 50, dimension size D = 30, and 30 independent runs. The best results are highlighted in bold. All the experiments were conducted on a PC with an Intel (R) Core (TM) i5-11300H CPU @ 3.10 GHz, 16 GB RAM, Windows 10, and MATLAB R2016b. Table 2 denotes the parameter setting of algorithms, and the details of the compared algorithms can be listed as follows:

- AOA: simulates four commonly used arithmetic operators as Division (÷), Multiplication (×), Subtraction (−), and Addition (+).
- Gold-SA: inspired by the sine function with the golden section search in mathematics compute.
- ROA: simulates remora's parasitism behavior on different hosts including whales and swordfish during the hunting process.
- AO: inspired by Aquila's four different hunting methods.
- SCA: simulates the distribution characteristics of sine and cosine functions.
- WOA: simulates the hunting behavior of humpback whales in oceans.
- FPA: simulates the pollination process of flowering plants in nature.
- DE: integrates the differential mutation, crossover, and selection mechanisms.

- GA: mimics the Darwinian evolution law and biological evolution of genetic mechanism in nature.

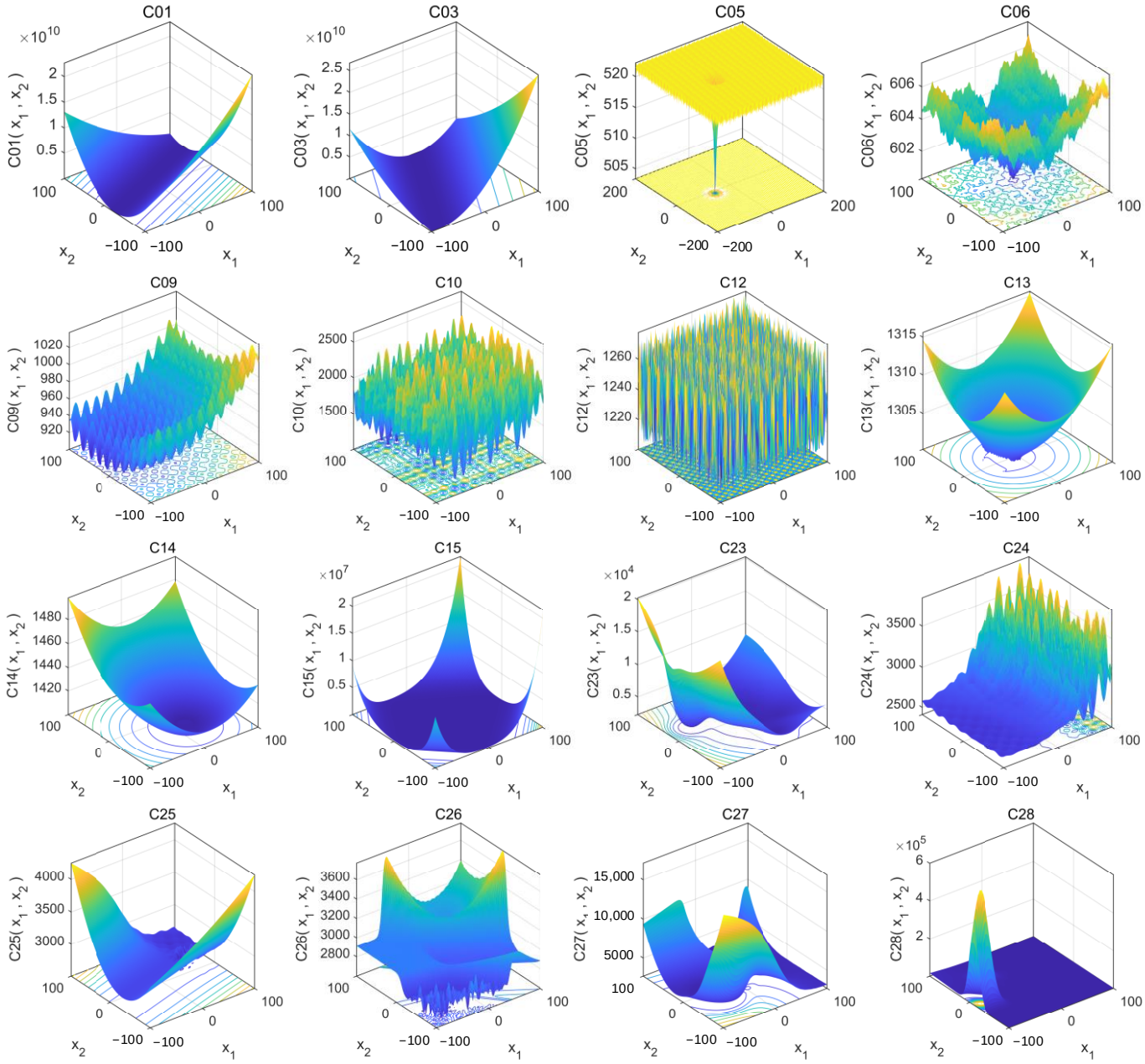

Figure 6. View of some CEC 2014 benchmark functions.

4.3. Statistical Results on CEC 2014 Benchmark Functions

Table 3 denotes the mean and standard deviation (std) values obtained by HAGSA and other competed algorithms for each CEC 2014 function with D = 30. According to Table 3, the statistical results illustrate that the HAGSA provides better searchability than its peers. For unimodal functions, HAGSA better obtains the global optimal solution on C01 and C03 than others. For multimodal functions, HAGSA outperforms all other well-known algorithms on nine functions, except functions C07–08, C11, and C14; FPA, DE, ROA, and AO find the global optimal solution for these functions, respectively. For hybrid functions, HAGSA achieves the best results for C16, C19, C20, and C22 among all algorithms. Finally, HAGSA also outperforms the results for composition functions compared to the original AOA, Gold-SA, and other compared algorithms on C23–25 and C28–C30, but not on C26.

Figure 7 shows HAGSA and competitor algorithms' ranking in various functions of the CEC 2014 test suite. In light of these results, HAGSA exhibits excellent performance by obtaining the best average over 21 functions.

Table 2. Parameter setting of each algorithm.

Algorithm	Parameters
AOA [26]	$\alpha = 5$; $\mu = 0.5$;
Gold-SA [27]	$c_1 = [1, 0]$; $c_2 \in [0, 1]$; $c_3 \in [0, 1]$
ROA [45]	$C = 0.1$
AO [23]	U = 0.00565; $r_1 = 10$; $\omega = 0.005$; $\alpha = 0.1$; $\delta = 0.1$;
SCA [25]	$a \in [2, 0]$
WOA [14]	$a_1 \in [2, 0]$; $a_2 \in [-1, -2]$; $b = 1$
FPA [46]	$p = 0.8$; $\beta = 1.5$
DE [8]	$F_{min} = 0.2$; $F_{max} = 0.8$; $CR = 0.1$
GA [47]	$Pc = 0.85$; $Pm = 0.01$

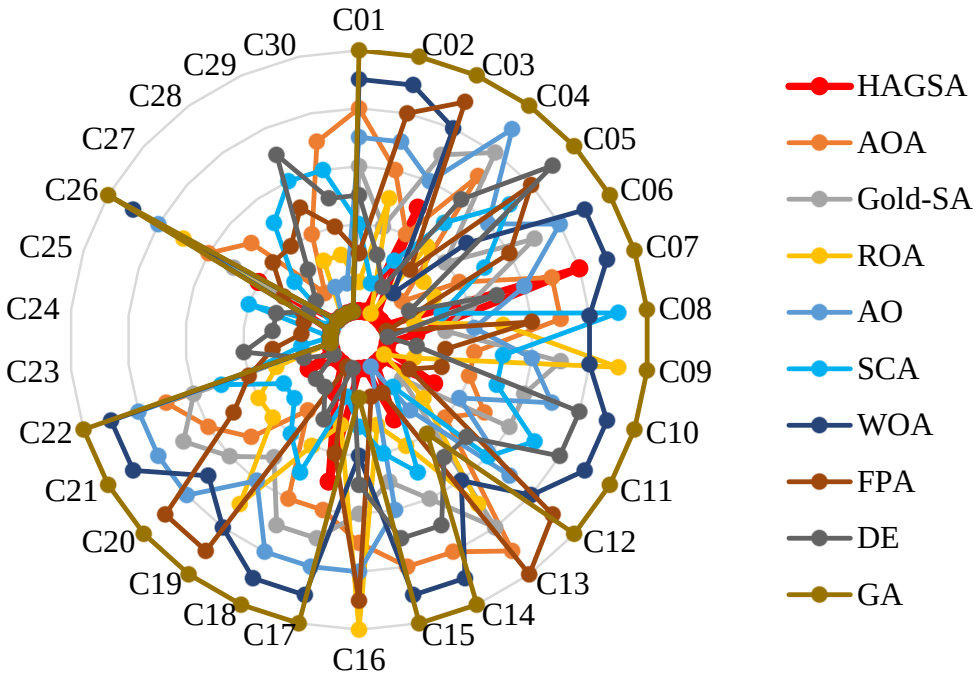

Figure 7. The radar graphs of algorithms on CEC 2014 benchmark functions.

4.4. Boxplot Behavior Analysis

The distribution characteristics of data can be displayed through boxplot analysis. The boxplot describes the data distribution as quartiles. The lowest and largest points of the edges of the boxplot are the minimum and maximum values obtained by the algorithm. The lower and upper quartiles are separated by the endpoints of the rectangle [5]. In this subsection, we use boxplot behavior to represent each algorithm's distribution of the obtained value. Each sample runs 30 times independently for each CEC 2014 benchmark function with D = 30. The boxplot behavior of each algorithm is shown in Figure 8. HAGSA has better stability for most benchmark functions and shows excellent performance compared to the others. In particular, for C01, C04, C05, C08, C09, C12, C13, and C15, the boxplot of the proposed HAGSA method is very narrow compared to others and shows lower values. For C06, C14, and C16, HAGSA achieves the lower values obtained than most algorithms. However, the performance is not obvious when solving C10, C17, C18, C19, C21, C23, C25, C27, and C30.

Table 3. The mean fitness and std obtained with the different algorithms on the CEC 2014 test suite.

Function		HAGSA	AOA	Gold-SA	ROA	AO	SCA	WOA	FPA	DE	GA
C01	Mean	$\mathbf{1.94 \times 10^{8}}$	1.08×10^{9}	6.73×10^{8}	3.59×10^{8}	7.85×10^{8}	5.11×10^{8}	1.97×10^{9}	4.63×10^{8}	5.30×10^{9}	2.77×10^{9}
	Std	7.50×10^{7}	3.49×10^{8}	2.21×10^{8}	1.63×10^{8}	3.92×10^{7}	1.26×10^{8}	3.25×10^{8}	1.97×10^{8}	2.23×10^{8}	1.07×10^{8}
C02	Mean	$\mathbf{2.40 \times 10^{10}}$	6.81×10^{10}	6.18×10^{10}	6.80×10^{10}	6.83×10^{10}	2.93×10^{10}	8.59×10^{10}	6.99×10^{10}	5.09×10^{10}	1.03×10^{11}
	Std	7.78×10^{9}	1.18×10^{10}	9.47×10^{9}	7.53×10^{9}	1.27×10^{9}	5.26×10^{9}	7.45×10^{9}	2.39×10^{9}	1.02×10^{10}	0.00
C03	Mean	8.55×10^{4}	8.19×10^{4}	8.73×10^{4}	$\mathbf{6.60 \times 10^{4}}$	8.72×10^{4}	7.58×10^{4}	9.20×10^{4}	1.26×10^{5}	7.01×10^{4}	1.42×10^{7}
	Std	2.10×10^{3}	6.52×10^{3}	2.50×10^{3}	7.55×10^{3}	7.66×10^{3}	1.61×10^{4}	1.22×10^{4}	6.25×10^{4}	1.56×10^{4}	1.25×10^{4}
C04	Mean	$\mathbf{1.45 \times 10^{4}}$	1.05×10^{4}	1.27×10^{4}	2.54×10^{3}	1.40×10^{4}	2.57×10^{3}	1.73×10^{4}	1.74×10^{3}	6.37×10^{3}	2.58×10^{4}
	Std	7.37×10^{2}	2.84×10^{3}	3.47×10^{3}	1.17×10^{3}	1.95×10^{2}	6.06×10^{2}	2.18×10^{3}	3.95×10^{2}	2.59×10^{3}	5.19×10^{2}
C05	Mean	$\mathbf{5.20 \times 10^{2}}$	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}	5.21×10^{2}
	Std	8.39×10^{2}	8.06×10^{2}	7.03×10^{2}	1.07×10^{-1}	8.92×10^{-2}	7.53×10^{-2}	8.15×10^{-2}	8.17×10^{-2}	6.61×10^{-2}	8.05×10^{-2}
C06	Mean	$\mathbf{6.17 \times 10^{2}}$	6.38×10^{2}	6.42×10^{2}	6.35×10^{2}	6.42×10^{2}	6.39×10^{2}	6.45×10^{2}	6.39×10^{2}	6.34×10^{2}	6.50×10^{2}
	Std	3.56	2.45	2.42	3.07	2.76	1.97	1.43	2.82	2.63	2.05
C07	Mean	1.47×10^{3}	1.34×10^{3}	1.13×10^{3}	9.16×10^{2}	1.19×10^{3}	9.50×10^{2}	1.56×10^{3}	$\mathbf{7.41 \times 10^{2}}$	1.16×10^{3}	1.75×10^{3}
	Std	7.04×10	1.06×10^{2}	9.78×10	9.09×10	1.24×10	$3. \times 10$	6.79×10	1.56×10	1.12×10^{2}	7.10×10
C08	Mean	1.09×10^{3}	1.14×10^{3}	1.12×10^{3}	1.13×10^{3}	1.13×10^{3}	1.19×10^{3}	1.18×10^{3}	1.13×10^{3}	$\mathbf{1.08 \times 10^{3}}$	1.31×10^{3}
	Std	2.42×10	3.04×10	3.10×10	2.57×10	2.02×10	2.22×10	1.37×10	4.63×10	2.32×10	2.23×10
C09	Mean	$\mathbf{1.14 \times 10^{3}}$	1.22×10^{3}	1.26×10^{3}	1.37×10^{3}	1.26×10^{3}	1.22×10^{3}	1.29×10^{3}	1.20×10^{3}	1.20×10^{3}	1.38×10^{3}
	Std	1.65×10	2.17×10	2.71×10	2.23×10	1.89×10	2.49×10	1.67×10	5.07×10	2.70×10	2.31×10^{-13}
C10	Mean	$\mathbf{6.12 \times 10^{3}}$	7.26×10^{3}	8.04×10^{3}	6.34×10^{3}	8.16×10^{3}	7.97×10^{3}	9.45×10^{3}	6.57×10^{3}	8.93×10^{3}	1.07×10^{4}
	Std	6.25×10^{2}	3.79×10^{2}	5.47×10^{2}	7.11×10^{2}	5.84×10^{2}	4.49×10^{2}	3.61×10^{2}	7.50×10^{2}	2.87×10^{2}	5.36×10^{2}
C11	Mean	7.56×10^{3}	7.85×10^{3}	8.90×10^{3}	$\mathbf{7.28 \times 10^{3}}$	7.81×10^{3}	8.96×10^{3}	1.01×10^{4}	7.47×10^{3}	9.31×10^{3}	1.10×10^{4}
	Std	7.10×10^{2}	4.20×10^{2}	5.68×10^{2}	6.88×10^{2}	6.68×10^{2}	2.55×10^{2}	3.79×10^{2}	7.89×10^{2}	4.51×10^{2}	4.69×10^{2}
C12	Mean	$\mathbf{1.20 \times 10^{3}}$	1.20×10^{3}	1.20×10^{3}	1.20×10^{3}	1.20×10^{3}	1.20×10^{3}	1.20×10^{3}	1.20×10^{3}	1.20×10^{3}	1.21×10^{3}
	Std	5.52×10	5.78×10^{-1}	5.35×10^{-1}	5.62×10^{-1}	5.98×10^{-1}	5.89×10^{-1}	6.48×10^{-1}	6.75×10^{-1}	5.80×10^{-1}	9.19×10^{-1}
C13	Mean	$\mathbf{1.30 \times 10^{3}}$	1.31×10^{3}	1.31×10^{3}	1.31×10^{3}	1.31×10^{3}	1.31×10^{3}	1.31×10^{3}	1.31×10^{3}	1.31×10^{3}	1.31×10^{3}
	Std	8.34×10^{-1}	9.07×10^{-1}	8.99×10^{-1}	8.64×10^{-1}	4.09×10^{-1}	3.93×10^{-1}	8.37×10^{-1}	9.22×10^{-1}	7.67×10^{-1}	4.48×10^{-1}

Table 3. *Cont.*

Function		HAGSA	AOA	Gold-SA	ROA	AO	SCA	WOA	FPA	DE	GA
C14	Mean	1.45×10^{3}	1.63×10^{3}	1.57×10^{3}	1.47×10^{3}	$\mathbf{1.41 \times 10^{3}}$	1.49×10^{3}	1.73×10^{3}	1.42×10^{3}	1.59×10^{3}	1.79×10^{3}
	Std	1.44×10	4.41×10	4.36×10	2.20×10	5.87×10	1.99×10	2.46×10	9.00	3.95×10	3.58×10
C15	Mean	$\mathbf{4.34 \times 10^{3}}$	2.50×10^{5}	4.92×10^{4}	9.04×10^{3}	8.92×10^{4}	2.54×10^{4}	5.38×10^{5}	4.74×10^{3}	1.03×10^{5}	9.16×10^{5}
	Std	2.25×10^{3}	1.31×10^{5}	3.55×10^{4}	8.15×10^{3}	3.54×10	1.62×10^{4}	1.55×10^{5}	2.24×10^{3}	1.35×10^{5}	4.74×10^{-10}
C16	Mean	$\mathbf{1.61 \times 10^{3}}$	1.61×10^{3}	1.61×10^{3}	1.61×10^{3}	1.61×10^{3}	1.61×10^{3}	1.61×10^{3}	1.61×10^{3}	1.61×10^{3}	1.61×10^{3}
	Std	3.71×10^{-1}	3.70×10^{-1}	3.21×10^{-1}	4.71×10^{-1}	4.08×10^{-1}	1.95×10^{-1}	2.08×10^{-1}	4.66×10^{-1}	2.37×10^{-1}	1.88×10^{-1}
C17	Mean	8.59×10^{7}	8.90×10^{7}	1.36×10^{8}	1.61×10^{7}	1.64×10^{8}	1.56×10^{7}	2.47×10^{8}	2.35×10^{7}	$\mathbf{9.17 \times 10^{6}}$	5.48×10^{8}
	Std	7.10×10^{6}	6.17×10^{7}	8.13×10^{7}	1.40×10^{7}	5.24×10^{6}	5.76×10^{6}	6.66×10^{7}	2.13×10^{7}	1.30×10^{7}	2.33×10^{8}
C18	Mean	1.36×10^{7}	2.44×10^{9}	2.78×10^{9}	2.90×10^{8}	3.80×10^{9}	4.77×10^{8}	7.52×10^{9}	$\mathbf{6.11 \times 10^{6}}$	2.65×10^{8}	1.20×10^{10}
	Std	2.06×10^{7}	2.04×10^{9}	1.67×10^{9}	6.85×10^{8}	2.05×10^{6}	2.52×10^{8}	2.30×10^{9}	7.19×10^{6}	3.73×10^{8}	3.82×10^{9}
C19	Mean	$\mathbf{2.01 \times 10^{3}}$	2.24×10^{3}	2.27×10^{3}	2.30×10^{3}	2.30×10^{3}	2.25×10^{3}	2.49×10^{3}	2.32×10^{3}	2.10×10^{3}	2.80×10^{3}
	Std	5.12×10	1.05×10^{2}	9.98×10^{1}	9.65×10	3.15×10	3.29×10	7.58×10	5.24×10	6.27×10	2.51×10
C20	Mean	$\mathbf{3.67 \times 10^{4}}$	1.86×10^{5}	2.45×10^{5}	9.06×10^{4}	4.34×10^{5}	5.90×10^{4}	3.43×10^{6}	4.97×10^{5}	2.75×10^{4}	1.07×10^{8}
	Std	3.96×10^{4}	9.23×10^{4}	1.27×10^{5}	6.04×10^{4}	5.11×10^{4}	2.94×10^{4}	4.51×10^{6}	7.78×10^{5}	2.08×10^{4}	2.87×10^{7}
C21	Mean	1.12×10^{6}	3.36×10^{7}	5.47×10^{7}	9.65×10^{6}	5.65×10^{7}	5.18×10^{6}	1.07×10^{8}	1.25×10^{7}	$\mathbf{5.17 \times 10^{5}}$	2.80×10^{8}
	Std	6.21×10^{5}	2.39×10^{7}	2.92×10^{7}	9.83×10^{6}	1.66×10^{6}	2.86×10^{6}	5.82×10^{7}	9.65×10^{6}	7.13×10^{5}	2.14×10^{8}
C22	Mean	$\mathbf{2.85 \times 10^{3}}$	4.93×10^{3}	4.69×10^{3}	3.28×10^{3}	6.49×10^{3}	3.37×10^{3}	3.08×10^{4}	3.32×10^{3}	3.08×10^{3}	1.68×10^{5}
	Std	2.12×10^{2}	2.12×10^{7}	1.78×10^{3}	7.41×10^{2}	2.78×10^{2}	1.72×10^{2}	3.05×10^{4}	2.89×10^{2}	2.52×10^{2}	7.01×10^{4}
C23	Mean	$\mathbf{2.50 \times 10^{3}}$	$\mathbf{2.50 \times 10^{3}}$	$\mathbf{2.50 \times 10^{3}}$	$\mathbf{2.50 \times 10^{3}}$	$\mathbf{2.50 \times 10^{3}}$	2.72×10^{3}	$\mathbf{2.50 \times 10^{3}}$	2.72×10^{3}	2.84×10^{3}	$\mathbf{2.50 \times 10^{3}}$
	Std	0.00	0.00	0.00	0.00	0.00	3.17×10	0.00	4.01×10	9.01×10	0.00
C24	Mean	$\mathbf{2.60 \times 10^{3}}$	$\mathbf{2.60 \times 10^{3}}$	$\mathbf{2.60 \times 10^{3}}$	$\mathbf{2.60 \times 10^{3}}$	$\mathbf{2.60 \times 10^{3}}$	$\mathbf{2.63 \times 10^{3}}$	$\mathbf{2.60 \times 10^{3}}$	2.61×10^{3}	2.69×10^{3}	$\mathbf{2.60 \times 10^{3}}$
	Std	0.00	8.87×10^{-2}	0.00	1.46×10^{-7}	2.34×10^{-5}	1.87×10	0.00	5.81	1.34×10	0.00
C25	Mean	$\mathbf{2.70 \times 10^{3}}$	$\mathbf{2.70 \times 10^{3}}$	$\mathbf{2.70 \times 10^{3}}$	$\mathbf{2.70 \times 10^{3}}$	$\mathbf{2.70 \times 10^{3}}$	2.75×10^{3}	$\mathbf{2.70 \times 10^{3}}$	2.72×10^{3}	2.73×10^{3}	$\mathbf{2.70 \times 10^{3}}$
	Std	0.00	0.00	0.00	0.00	0.00	1.15×10	0.00	1.83×10	8.74	0.00
C26	Mean	2.77×10^{3}	2.77×10^{3}	2.77×10^{3}	2.77×10^{3}	2.78×10^{3}	$\mathbf{2.70 \times 10^{3}}$	2.79×10^{3}	2.74×10^{3}	2.73×10^{3}	2.79×10^{3}
	Std	4.33×10	4.41×10	4.25×10	4.62×10	4.99×10	4.96×10^{-1}	2.35×10^{1}	8.00×10	4.33×10	2.39×10

Table 3. *Cont.*

Function		HAGSA	AOA	Gold-SA	ROA	AO	SCA	WOA	FPA	DE	GA
C27	Mean	$\mathbf{2.90 \times 10^3}$	4.05×10^3	$\mathbf{2.90 \times 10^3}$	$\mathbf{2.90 \times 10^3}$	$\mathbf{2.90 \times 10^3}$	3.91×10^3	$\mathbf{2.90 \times 10^3}$	3.99×10	3.86×10^3	$\mathbf{2.90 \times 10^3}$
	Std	0.00	3.71×10^2	0.00	0.00	3.98	2.65×10^2	0.00	2.42×10^2	2.46×10^2	0.00
C28	Mean	$\mathbf{3.00 \times 10^3}$	5.34×10^3	$\mathbf{3.00 \times 10^3}$	$\mathbf{3.00 \times 10^3}$	$\mathbf{3.00 \times 10^3}$	5.95×10^3	$\mathbf{3.00 \times 10^3}$	5.40×10^3	5.36×10^3	$\mathbf{3.00 \times 10^3}$
	Std	0.00	2.75×10^3	0.00	0.00	0.00	6.11×10^2	0.00	8.95×10^2	4.64×10^2	0.00
C29	Mean	$\mathbf{3.10 \times 10^3}$	4.32×10^8	$\mathbf{3.10 \times 10^3}$	7.17×10^6	1.46×10^4	4.43×10^7	$\mathbf{3.10 \times 10^3}$	1.79×10^7	6.87×10^7	$\mathbf{3.10 \times 10^3}$
	Std	0.00	1.80×10^8	0.00	7.00×10^6	6.28×10^4	1.75×10^7	0.00	1.63×10^7	5.50×10^7	0.00
C30	Mean	$\mathbf{3.20 \times 10^3}$	4.16×10^6	$\mathbf{3.20 \times 10^3}$	3.29×10^5	1.66×10^5	6.97×10^5	$\mathbf{3.20 \times 10^3}$	4.02×10^5	4.17×10^5	$\mathbf{3.20 \times 10^3}$
	Std	0.00	2.65×10^6	0.00	2.80×10^5	1.44×10^5	2.87×10^5	0.00	2.76×10^5	2.34×10^5	0.00

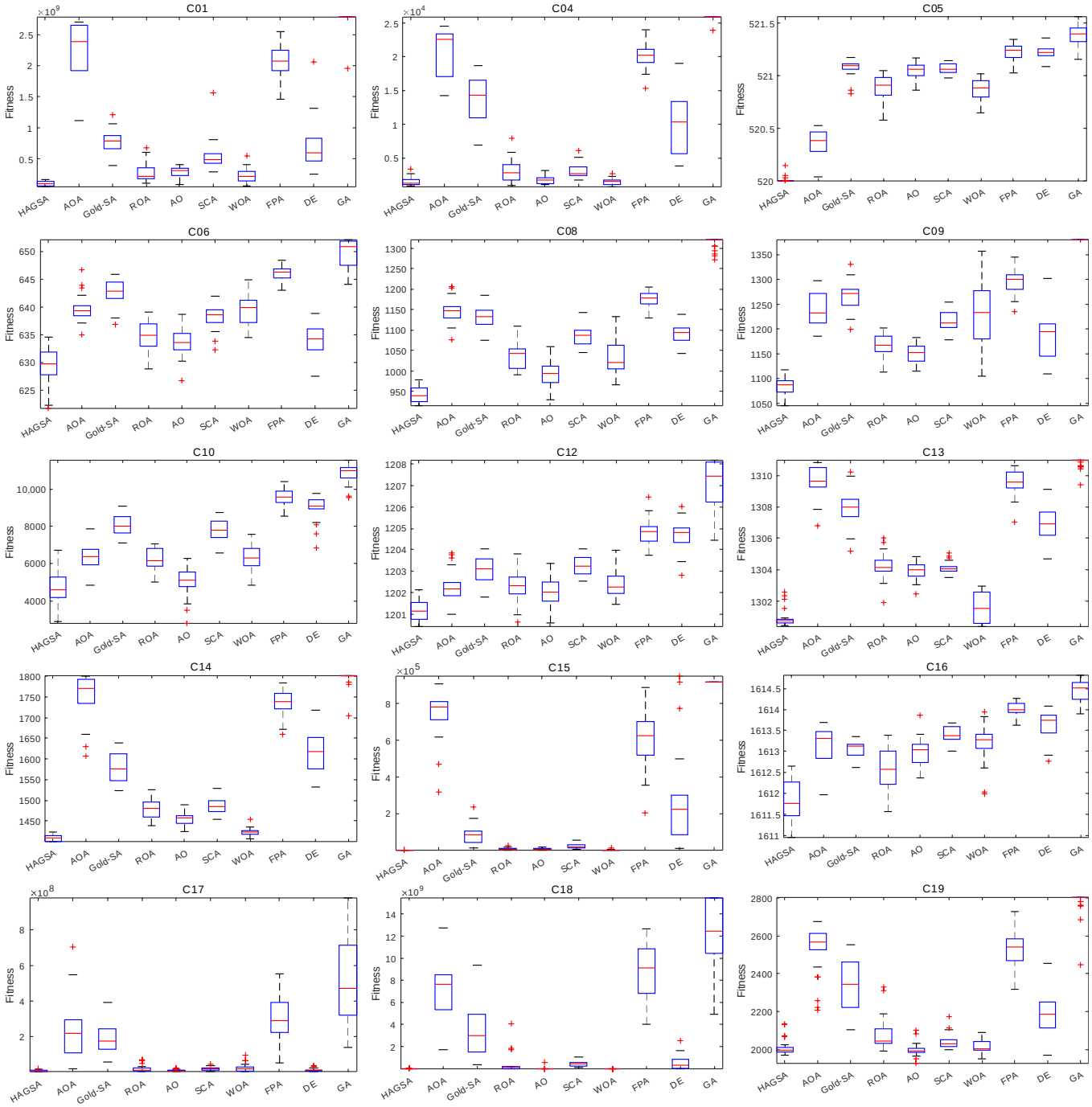

Figure 8. *Cont.*

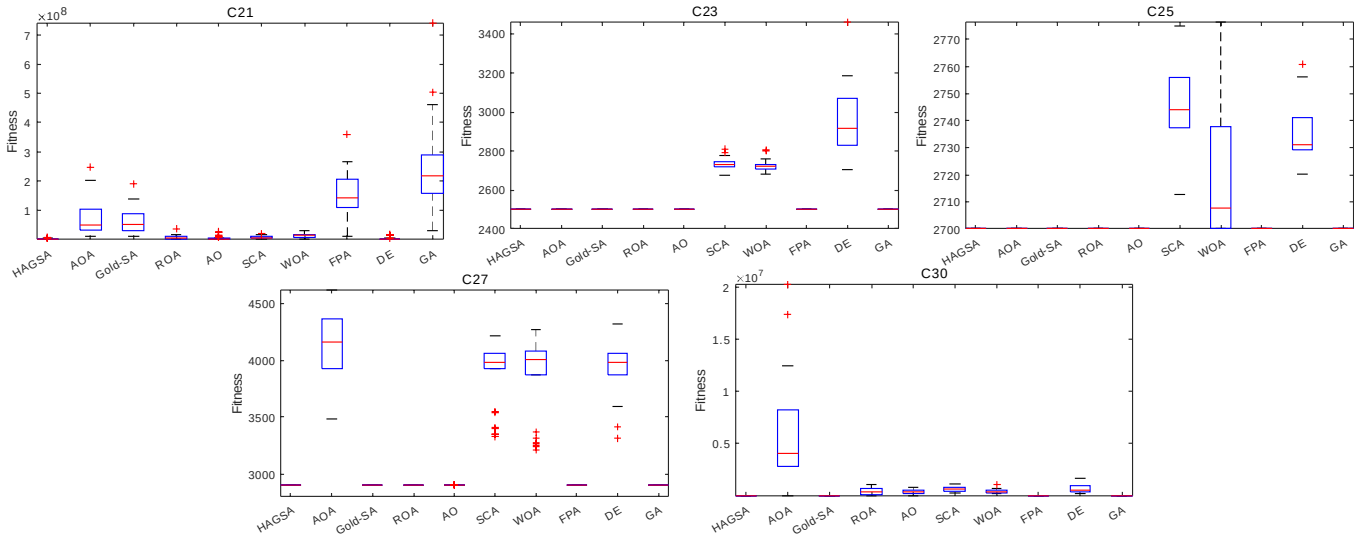

Figure 8. Boxplot behavior of algorithms on some functions.

4.5. Convergence Behavior Analysis

In this subsection, we analyze the convergence behavior of each algorithm used over some benchmark functions. Figure 9 shows the convergence behavior of HAGSA, AOA, Gold-SA, ROA, AO, SCA, WOA, FPA, DE, and GA for selected functions. As can be seen from this figure, HAGSA achieves excellent behavior for most functions, which suggests the convergence of the proposed method. For unimodal functions (C01 and C03), although the convergence speed is slower than WOA in the early iteration (for C01), the convergence accuracy is higher than WOA at the end of the iteration. For C03, HAGSA has the fastest convergence speed and highest convergence accuracy. On the multimodal functions, HAGSA still maintains the fastest convergence speed and highest accuracy on most functions. In particular, for C05 and C06, although the global optimal is not found, HAGSA still has excellent performance compared to the others. However, the optimal value of HAGSA is ranked third and the WOA and AO are ranked first and second, respectively, when solving C07. For C10 and C11, it can be seen that the convergence curve of HAGSA is accelerated in the later stage of iteration; this is due to the excellent ability to jump out of the local optimal as a result of Brownian mutation. On hybrid functions, the convergence accuracy is still good compared to the others. For C16, C20, C21, and C22, the proposed HAGSA algorithm demonstrates its better performance compared to the original AOA and Gold-SA. On composition functions, the improvement is not obvious compared to the original Gold-SA and other well-known algorithms such as GA and FPA.

4.6. Wilcoxon Rank-Sum Test

Because the results obtained by each algorithm are random, in this subsection, we utilize the Wilcoxon rank-sum test (WRS) to evaluate the statistical significance difference between two samples at a significance level of 5% [2]. Specifically, if the p-value is less than 0.05, it indicates the statistical difference is significant; otherwise, the difference is not obvious. Furthermore, NaN denotes there is no difference between the two samples. The statistical results of the Wilcoxon rank-sum test are listed in Table 4; from this table, we can see that the proposed HAGSA algorithm shows better significant performance than the other algorithms on most benchmark functions.

4.7. Computational Time Analysis

To show the computational cost of the proposed HAGSA, in this subsection, we record the computational time cost obtained by algorithms on the CEC 2014 test suite. The statistical results are listed in Table 5; although HAGSA has the same computational complexity as AOA and Gold-SA, the computational time cost of HAGSA is more than that

of AOA and Gold-SA. This is because HAGSA uses Brownian mutation to generate two candidates' solutions to enhance the algorithm's searchability and Levy flight is used to improve the exploitation ability of the hybrid algorithm. In addition, considering the NFL theorem, it is acceptable to increase computational time to obtain reliable solutions.

4.8. Industrial Engineering Design Problems

This subsection introduces five real-world industrial engineering design problems to evaluate the proposed algorithm's searchability, including the car side crash design problem, pressure vessel design problem, tension spring design problem, speed reducer design problem, and cantilever beam design problem. Unlike benchmark functions, these industrial engineering design problems have many inequality and equality constraints, which is a vital challenge to MAs. In addition, using these problems helps evaluate the potential of algorithms to solve real-world problems.

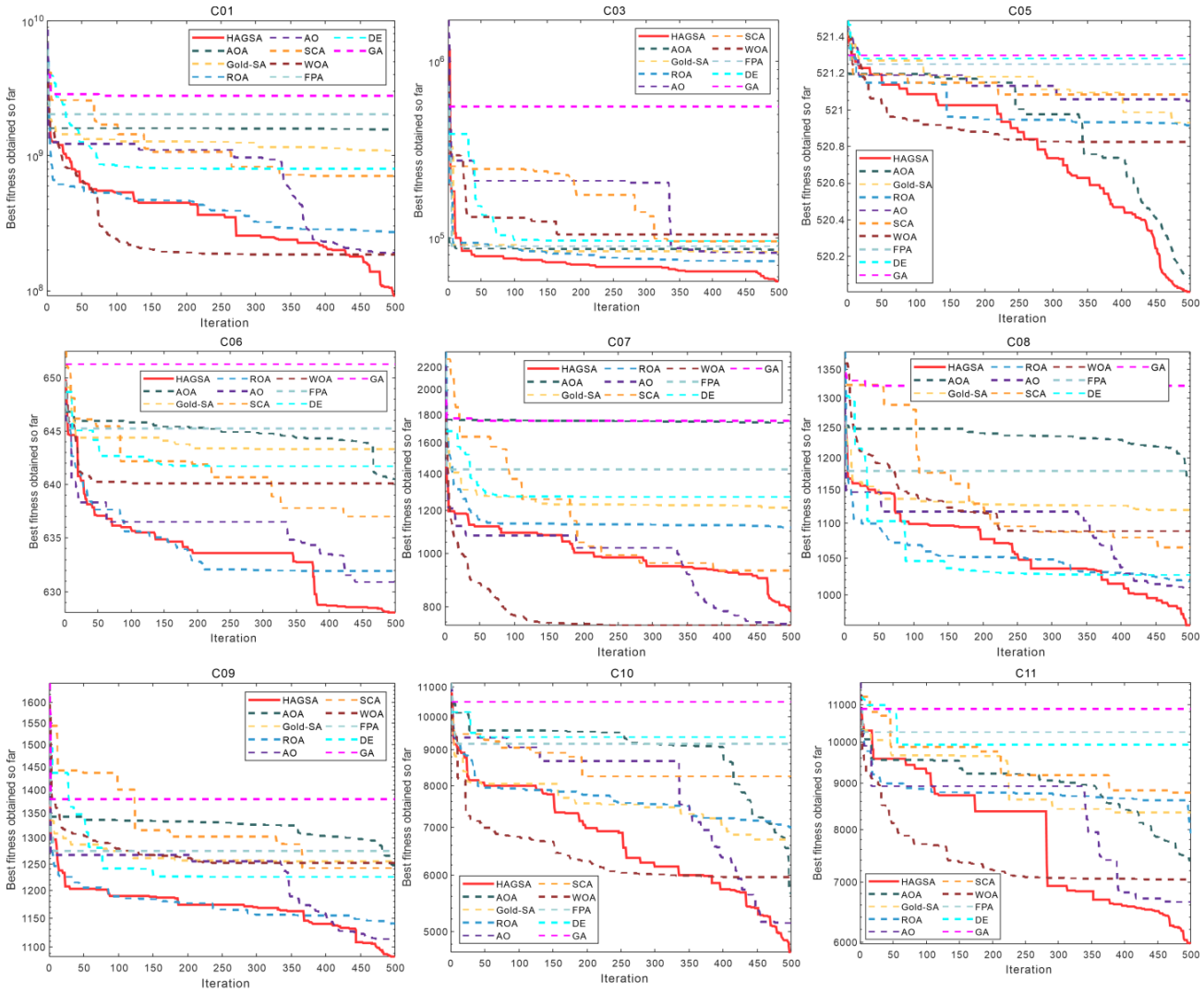

Figure 9. *Cont.*

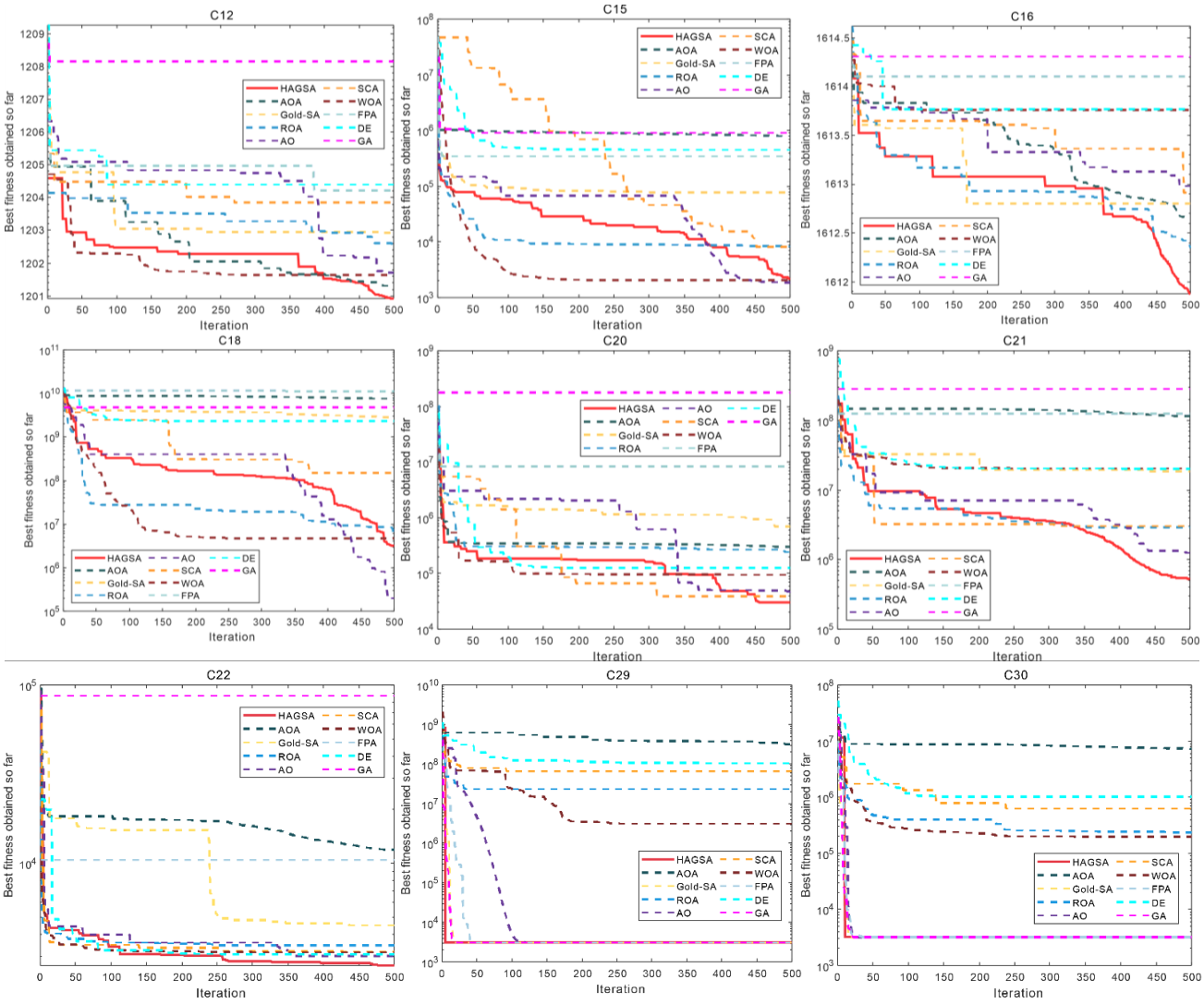

Figure 9. Convergence curve of algorithms on some functions.

4.8.1. Car Side Crash Design Problem

This problem aims to maintain the side impact crash performance and minimize the vehicle weight [48]. It has 11 parameters that need to be optimized; also, ten constraints were integrated into this problem. The model of this problem can be established as follows:

Consider $x = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ x_{11}]$

Minimize $f(x) = \text{Weight}$,

$$\text{Subject to} \begin{cases} g_1(x) = F_a(\text{load in abdomen}) \leq 1\ \text{kN}, \\ g_2(x) = V \times Cu\ (\text{dummyupperchest}) \leq 0.32\ \text{m/s}, \\ g_3(x) = V \times Cm\ (\text{dummymiddlechest}) \leq 0.32\ \text{m/s}, \\ g_4(x) = V \times Cl\ (\text{dummylowerchest}) \leq 0.32\ \text{m/s}, \\ g_5(x) = \Delta_{\text{ur}}\ (\text{upperribdeflection}) \leq 32\ \text{mm}, \\ g_6(x) = \Delta_{\text{mr}}\ (\text{middleribdeflection}) \leq 32\ \text{mm}, \\ g_7(x) = \Delta_{\text{lr}}\ (\text{lowerribdeflection}) \leq 32\ \text{mm}, \\ g_8(x) = F\ (\text{Publicforce})_p \leq 4\ \text{kN}, \\ g_9(x) = V_{\text{MBP}}(\text{Velocity of V} - \text{Pillarat middle point}) \leq 9.9\ \text{mm/ms}, \\ g_{10}(x) = V_{\text{FD}}(\text{Velocity of front door at V} - \text{Pillar}) \leq 15.7\ \text{mm/ms}, \end{cases}$$

$$\text{Variable range} \begin{cases} 0.5 \leq x_1 - x_7 \leq 1.5,\ 0.192 < x_8, x_9 < 0.345, \\ -30 \leq x_{10}, x_{11} \leq 30, \end{cases}$$

Table 4. Statistical results of Wilcoxon rank-sum test obtained by each algorithm.

Function	HAGSA vs.								
	AOA	**Gold-SA**	**ROA**	**AO**	**SCA**	**WOA**	**FPA**	**DE**	**GA**
C01	3.02×10^{-11}	3.02×10^{-11}	2.71×10^{-2}	2.13×10^{-4}	4.08×10^{-11}	2.64×10^{-1}	3.02×10^{-11}	1.29×10^{-9}	2.37×10^{-12}
C02	3.02×10^{-11}	3.02×10^{-11}	7.01×10^{-2}	3.02×10^{-11}	5.83×10^{-13}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.21×10^{-12}
C03	3.02×10^{-11}	3.02×10^{-11}	3.32×10^{-6}	2.49×10^{-6}	2.00×10^{-5}	6.52×10^{-9}	3.02×10^{-11}	3.82×10^{-10}	3.02×10^{-11}
C04	3.02×10^{-11}	3.02×10^{-11}	2.06×10^{-2}	2.61×10^{-10}	4.71×10^{-4}	1.31×10^{-8}	3.02×10^{-11}	3.69×10^{-11}	1.21×10^{-12}
C05	1.78×10^{-10}	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}
C06	4.62×10^{-10}	3.02×10^{-11}	2.75×10^{-3}	7.73×10^{-2}	2.23×10^{-9}	1.29×10^{-11}	3.02×10^{-11}	1.30×10^{-1}	2.95×10^{-11}
C07	3.02×10^{-11}	6.07×10^{-11}	1.45×10^{-1}	3.02×10^{-11}	2.13×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.16×10^{-12}
C08	3.02×10^{-11}	3.02×10^{-11}	1.73×10^{-6}	1.25×10^{-7}	3.02×10^{-11}	2.84×10^{-4}	3.02×10^{-11}	5.49×10^{-11}	9.40×10^{-12}
C09	3.02×10^{-11}	3.02×10^{-11}	1.44×10^{-2}	7.70×10^{-8}	3.34×10^{-11}	4.42×10^{-6}	3.02×10^{-11}	6.12×10^{-10}	1.21×10^{-12}
C10	2.23×10^{-9}	3.02×10^{-11}	5.09×10^{-8}	2.97×10^{-1}	3.02×10^{-11}	1.46×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
C11	3.50×10^{-9}	3.02×10^{-11}	1.37×10^{-3}	5.01×10^{-1}	3.34×10^{-11}	2.96×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
C12	6.91×10^{-4}	2.39×10^{-8}	1.76×10^{-2}	2.90×10^{-1}	1.69×10^{-9}	5.27×10^{-5}	4.50×10^{-11}	3.02×10^{-11}	2.80×10^{-11}
C13	3.02×10^{-11}	3.69×10^{-11}	1.38×10^{-2}	6.07×10^{-11}	1.68×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	7.88×10^{-12}
C14	3.02×10^{-11}	3.02×10^{-11}	4.22×10^{-4}	1.33×10^{-10}	1.39×10^{-6}	1.09×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	1.72×10^{-12}
C15	3.02×10^{-11}	3.02×10^{-11}	5.49×10^{-1}	3.02×10^{-11}	1.69×10^{-9}	2.37×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	1.21×10^{-12}
C16	1.56×10^{-8}	4.18×10^{-9}	1.64×10^{-5}	2.23×10^{-9}	4.50×10^{-11}	3.20×10^{-9}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
C17	3.02×10^{-11}	3.02×10^{-11}	6.97×10^{-3}	3.95×10^{-1}	7.22×10^{-6}	2.43×10^{-5}	3.02×10^{-11}	2.32×10^{-2}	3.00×10^{-11}
C18	3.02×10^{-11}	3.02×10^{-11}	4.86×10^{-3}	1.21×10^{-10}	6.70×10^{-11}	3.96×10^{-8}	3.02×10^{-11}	4.08×10^{-11}	2.63×10^{-11}
C19	3.02×10^{-11}	3.34×10^{-11}	2.39×10^{-4}	4.35×10^{-5}	5.61×10^{-5}	3.55×10^{-1}	3.02×10^{-11}	4.57×10^{-9}	1.72×10^{-12}
C20	3.02×10^{-11}	1.41×10^{-9}	1.00×10^{-3}	9.83×10^{-8}	1.91×10^{-2}	2.20×10^{-7}	3.69×10^{-11}	7.96×10^{-3}	3.02×10^{-11}
C21	3.02×10^{-11}	3.02×10^{-11}	3.18×10^{-4}	4.17×10^{-2}	2.28×10^{-5}	1.07×10^{-7}	3.02×10^{-11}	3.38×10^{-2}	3.02×10^{-11}
C22	5.49×10^{-11}	1.46×10^{-10}	5.32×10^{-3}	3.03×10^{-2}	7.70×10^{-8}	1.64×10^{-5}	3.02×10^{-11}	4.06×10^{-2}	3.02×10^{-11}
C23	1.21×10^{-12}	NaN	NaN	NaN	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN
C24	1.21×10^{-12}	NaN	1.61×10^{-1}	6.62×10^{-4}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN
C25	1.21×10^{-12}	NaN	NaN	NaN	1.21×10^{-12}	1.93×10^{-9}	NaN	1.21×10^{-12}	NaN
C26	8.11×10^{-8}	3.55×10^{-1}	2.86×10^{-4}	4.56×10^{-2}	3.98×10^{-6}	9.59×10^{-9}	8.00×10^{-1}	7.40×10^{-3}	1.89E-02
C27	1.21×10^{-12}	NaN	NaN	4.19×10^{-2}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN
C28	1.21×10^{-12}	NaN	NaN	NaN	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN
C29	1.21×10^{-12}	NaN	6.61×10^{-5}	1.61×10^{-1}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN
C30	1.21×10^{-12}	NaN	6.25×10^{-10}	1.31×10^{-7}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	NaN

Table 5. The computational time for HAGSA and its peers.

Function	HAGSA	AOA	Gold-SA	ROA	AO	SCA	WOA	FPA	DE	GA
C01	0.5375	0.1722	**0.1260**	0.3587	0.3303	0.1756	0.1482	0.2102	0.2743	0.1516
C02	0.5998	0.1487	**0.0918**	0.2918	0.2854	0.1491	0.1332	0.1697	0.2010	0.1048
C03	0.5519	0.1659	0.1094	0.2395	0.2817	0.1588	0.1391	0.1558	0.2050	**0.1043**
C04	0.5085	0.1585	**0.0929**	0.2545	0.2499	0.1490	0.1803	0.1472	0.1971	0.1027
C05	0.5959	0.1564	0.1334	0.3615	0.3404	0.1521	0.1794	0.1705	0.2365	**0.1136**
C06	6.7234	1.1244	1.4928	5.5571	2.3889	1.5203	1.5837	1.5670	3.1240	**1.3135**
C07	0.6473	0.1605	0.1203	0.2872	0.3283	0.1850	0.1192	0.1661	0.2290	**0.1047**
C08	0.4786	0.1447	**0.1027**	0.2972	0.2493	0.1391	0.1212	0.1707	0.1799	0.1051
C09	0.6048	0.1847	0.1061	0.3045	0.2735	0.1681	0.1289	0.1791	0.2101	**0.1046**
C10	0.8256	0.1980	**0.1440**	0.4217	0.4360	0.2012	0.1474	0.2088	0.3306	0.1520
C11	0.8986	0.2100	**0.1596**	0.7439	0.4011	0.2126	0.1802	0.2147	0.3569	0.2055
C12	1.2724	0.3245	**0.2602**	1.1208	0.6199	0.3206	0.2946	0.3260	0.7370	0.3250
C13	0.5331	0.1451	**0.0933**	0.2555	0.2930	0.1541	0.1137	0.1541	0.2013	0.0944
C14	0.5130	0.1508	**0.1108**	0.3054	0.2816	0.1956	0.1180	0.1509	0.1855	0.1184
C15	0.4946	0.1610	0.1320	0.3372	0.3080	0.1914	0.1421	0.1771	0.2245	**0.1277**
C16	0.5078	0.1538	**0.0978**	0.3274	0.3163	0.1599	0.1164	0.1952	0.2397	0.1200
C17	0.6081	0.1684	0.1537	0.4210	0.3202	0.1730	0.1852	0.1790	0.2857	**0.1479**
C18	0.4803	0.1439	**0.1028**	0.3162	0.3332	0.2505	0.1138	0.2067	0.2115	0.1057
C19	1.6777	0.3450	0.3121	1.2646	0.7030	0.5136	0.3195	0.5490	0.8568	**0.2681**
C20	0.4975	0.1584	**0.0987**	0.3122	0.3295	0.1577	0.1371	0.1958	0.2195	0.1226
C21	0.5846	0.2016	**0.1269**	0.4338	0.3211	0.1794	0.1587	0.1744	0.2701	0.1310
C22	0.6756	0.1951	**0.1308**	0.5880	0.3688	0.1932	0.1534	0.2029	0.3152	0.1749
C23	1.8811	0.3695	**0.3146**	1.3598	0.7576	0.3692	0.4163	0.3930	0.9236	0.3200
C24	1.4512	0.2916	**0.2459**	1.1242	0.5935	0.4264	0.2725	0.4199	0.9495	0.2777
C25	1.6396	0.3334	**0.2878**	1.3109	0.7071	0.3465	0.3106	0.4518	1.0307	0.3435
C26	6.4800	**1.5258**	2.0984	5.0402	3.0212	2.0210	1.7759	1.7384	4.3710	1.6796
C27	6.3308	1.5334	**1.2872**	4.7350	2.8645	1.8617	1.8453	1.7505	4.3188	1.5384
C28	1.7570	0.4684	0.3955	1.1151	0.8401	0.4585	0.5362	0.6638	1.1272	**0.3811**
C29	2.0752	**0.4942**	0.6205	1.5271	0.9543	0.7252	0.6315	0.6343	1.4255	0.6466
C30	1.2367	**0.3321**	0.2759	0.8986	0.6684	0.3431	0.3148	0.3581	0.7941	0.4417

Table 6 shows the best results obtained by all algorithms. As shown in this table, the results of the proposed HAGSA are superior to those of other optimization techniques, and ROA and AO approaches are ranked second and third, respectively.

Table 6. Statistical results of car side crash design problem.

Algorithm	Optimum Variables											Optimum Cost
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	
HAGSA	**0.5**	**1.253**	**0.5**	**1.109**	**0.5**	**0.5**	**0.501**	**0.344**	**0.192**	**3.904**	**6.381**	**22.9765**
AOA	0.5	1.262	0.5	1.156	0.5	0.772	0.5	0.310	0.192	0.365	1.162	23.2139
Gold-SA	0.5	1.278	0.612	1.102	0.544	1.323	0.5	0.345	0.345	0.170	0.294	23.9711
ROA	0.5	1.235	0.5	1.166	0.5	1.110	0.5	0.341	0.192	0.275	2.926	23.0801
AO	0.724	1.175	0.502	1.200	0.5	0.792	0.5	0.308	0.192	0.739	2.837	23.1694
SCA	0.567	1.334	0.540	1.167	0.5	1.109	0.5	0.233	0.263	0.301	2.393	24.3513
WOA	0.953	1.106	0.5	1.206	0.524	0.559	0.501	0.282	0.298	0.246	7.326	24.6495
FPA	0.532	1.322	0.515	1.143	0.616	0.516	0.534	0.197	0.197	0.710	1.892	24.1309
DE	0.505	1.446	0.521	1.182	0.5	1.466	0.5	0.312	0.192	1.008	13.266	24.7181
GA	1.073	1.0465	0.595	1.096	0.714	0.502	0.521	0.322	0.264	5.549	8.215	25.4504

4.8.2. Pressure Vessel Design Problem

The pressure vessel design problem is shown in Figure 10. The goal of this problem is to minimize the total cost [49]. It has four design parameters: shell thickness (T_s), ball thickness (T_h), shell radius (R), and shell length (L). The constraints and objective function can be expressed as follows:

Consider $x = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L]$

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

$$\text{Subject to} \begin{cases} g_1(x) = -x_1 + 0.0193x_3 \le 0,\ g_2(x) = -x_3 + 0.00954x_3 \le 0, \\ g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \le 0,\ g_4(x) = x_4 - 240 \le 0 \end{cases}$$

$$\text{Variable range} \begin{cases} 0 \le x_1 \le 99,\ 0 \le x_2 \le 99, \\ 10 \le x_3 \le 200,\ 10 \le x_4 \le 200 \end{cases}$$

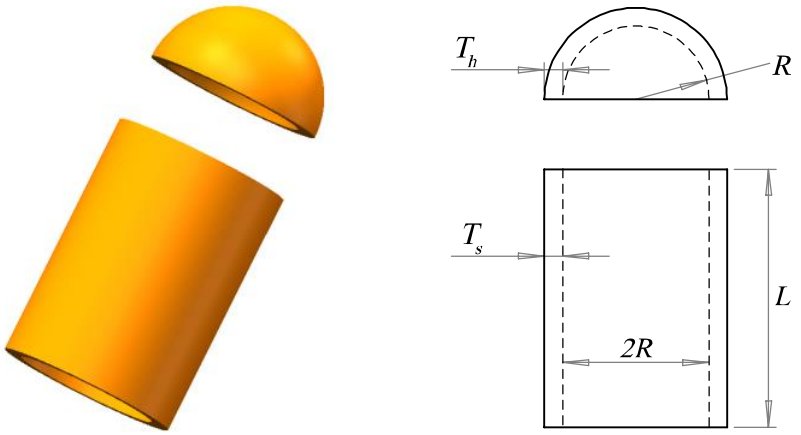

Figure 10. Pressure vessel design problem.

Table 7 shows the statistical results obtained by HAGSA and other comparison algorithms including AOA, Gold-SA, ROA, AO, SCA, WOA, FPA, DE, and GA. As can be seen from this table, HAGSA achieves competitive results in this design problem, and the results of ROA and AO are ranked second and third, respectively.

Table 7. Statistical results of the pressure vessel design problem.

Algorithm	Optimum Variables				Optimum Cost
	T_s	T_h	R	L	
HAGSA	0.8304795	0.3770664	44.00935	154.9557	5982.8355
AOA	0.8395475	0.4113845	44.27936	156.8883	6068.3284
Gold-SA	0.7140179	0.4619435	40.49522	197.7362	6090.4062
ROA	0.8610026	0.3934984	44.96907	144.2921	6023.0145
AO	0.8030047	0.4524486	43.65139	158.3146	6024.2153
SCA	0.963087	0.476939	51.4412	87.3095	6246.7789
WOA	0.937726	0.473373	49.9436	98.8134	6195.7655
FPA	0.971843	0.478402	52.5479	81.3225	6393.2109
DE	1.009677	0.498834	54.0470	69.2270	6398.6641
GA	1.025422	0.484037	54.7458	64.6720	6439.9228

4.8.3. Tension Spring Design Problem

The main goal of this problem is to find the optimal parameters to minimize the production cost [50]. There are three parameters: wire diameter (d), mean diameter of the spring (D), and number of active coils (N), as shown in Figure 11. The mathematical model is expressed as follows:

Consider $x = [x_1\ x_2\ x_3] = [d\ D\ N]$

Minimize $f(x) = (x_3 + 2)x_2 x_1^2$

$$\text{Subject to} \begin{cases} g_1(x) = 1 - \frac{x_2^3 x_3}{71{,}785x_1^4} \le 0,\ g_2(x) = \frac{4x_2^2 - x_1 x_2}{12{,}566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \le 0, \\ g_3(x) = 1 - \frac{140.45x_1}{x_2^2 x_3} \le 0,\ g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \le 0 \end{cases}$$

$$\text{Variable range} \begin{cases} 0.05 \le x_1 \le 2.00,\ 0.25 \le x_2 \le 1.30 \\ 2.00 \le x_3 \le 15.00 \end{cases}$$

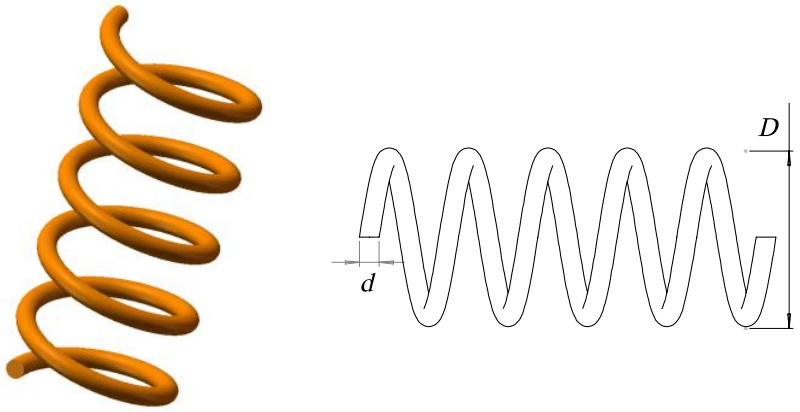

Figure 11. Tension spring design problem.

The statistical results of the tension spring design problem were obtained by HAGSA and other comparison algorithms as listed in Table 8. As can be seen from this table, the best cost of this design problem is 0.011196, and the three parameters are 0.050411, 0.37384, and 9.7854, respectively.

Table 8. Statistical results of the tension spring design problem.

Algorithm	Optimum Variables			Optimum Cost
	d	*D*	*N*	
HAGSA	0.050411	0.37384	9.7854	0.011196
AOA	0.051791	0.388	9.5556	0.012026
Gold-SA	0.060683	0.67982	3.1063	0.012783
ROA	0.059221	0.6308	3.5188	0.012209
AO	0.05	0.337193	13.0905	0.012721
SCA	0.061365	0.70355	2.9232	0.013043
WOA	0.0502069	0.351224	12.336	0.012692
FPA	0.10187	1.093	9.5387	0.130890
DE	0.06766	0.907935	2.0871	0.016985
GA	0.05401	0.465113	9.6797	0.015848

4.8.4. Speed Reducer Design Problem

This problem aims to construct a speed reducer with a minimum weight under constraints [51]. There are seven parameters: face width, the module of teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings, the diameter of the first shafts, and the diameter of second shafts. Figure 12 shows the design of this problem, and its mathematical formula is as follows:

Consider $x = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7]$

Minimize $f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3)$,

$$\text{Subject to} \begin{cases} g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \le 0,\ g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0, \\ g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0,\ g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0, \\ g_5(x) = \frac{\sqrt{(\frac{745x_4}{x_2x_3})^2 + 16.9\times10^6}}{110.0x_6^3} - 1 \le 0,\ g_6(x) = \frac{\sqrt{(\frac{745x_4}{x_2x_3})^2 + 157.5\times10^6}}{85.0x_6^3} - 1 \le 0, \\ g_7(x) = \frac{x_2x_3}{40} - 1 \le 0,\ g_8(x) = \frac{5x_2}{x_1} - 1 \le 0, \\ g_9(x) = \frac{x_1}{12x_2} - 1 \le 0,\ g_{10}(x) = \frac{1.5x_6+1.9}{x_4} - 1 \le 0, \\ g_{11}(x) = \frac{1.1x_7+1.9}{x_5} - 1 \le 0, \end{cases}$$

$$\text{Variable range} \begin{cases} 2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8, \\ 17 \le x_3 \le 28,\ 7.3 \le x_4 \le 8.3, \\ 7.8 \le x_5 \le 8.3,\ 2.9 \le x_6 \le 3.9, \\ 5.0 \le x_7 \le 5.5 \end{cases}$$

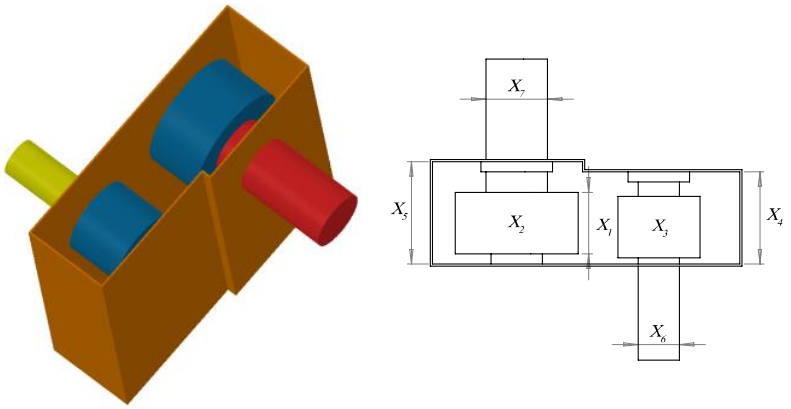

Figure 12. Speed reducer problem.

The proposed HAGSA is compared with AOA, Gold-SA, ROA, AO, SCA, WOA, FPA, DE, and GA. The statistical results are shown in Table 9. As can be seen, HAGSA is excellent for solving speed reducer design problems, and the results obtained by HAGSA are ranked first. The results of AOA and ROA are ranked second and third, respectively.

Table 9. Statistical results of the speed reducer design problem.

Algorithm	Optimum Variables							Optimum Cost
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
HAGSA	**3.49767**	**0.7**	**17**	**7.3**	**7.8001**	**3.34982**	**5.28559**	**2995.4897**
AOA	3.50776	0.7	17	7.77685	7.96133	3.35075	5.28557	3007.0806
Gold-SA	3.49441	0.7	17	7.3	7.8	3.42383	5.2872	3016.2163
ROA	3.50776	0.7	17	7.77685	7.96133	3.35075	5.28557	3007.0806
AO	3.49748	0.7	17	8.07645	7.8	3.35162	5.28573	3002.8462
SCA	3.6	0.7	17	8.3	8.3	3.43032	5.30013	3085.2732
WOA	3.5247	0.7	17	8.14441	8.05897	3.35091	5.28568	3019.883
FPA	3.6	0.7	17	7.3	7.8	3.41261	5.28143	3056.8032
DE	3.5119	0.7	17	8.3	8.3	3.37356	5.38151	3088.6759
GA	3.4896	0.7	17	7.71388	7.8	3.65614	5.29218	3094.3185

4.8.5. Cantilever Beam Design

The design of the cantilever beam is shown in Figure 13, and the goal of this problem is to minimize the total weight [52]. There are five parameters that need to be optimized. The objective function and constraints of this problem are as follows:

Consider $x = [x_1\ x_2\ x_3\ x_4\ x_5]$

Minimize $f(x) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5)$

Subject to $g(x) = \frac{60}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0$

Variable range $0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100$

The statistical results obtained by HAGSA, AOA, Gold-SA, ROA, AO, SCA, WOA, FPA, DE, and GA are shown in Table 10. From this table, HAGSA shows a lower cost than that of other optimization techniques, and the results of ROA and AO are ranked second and third, respectively.

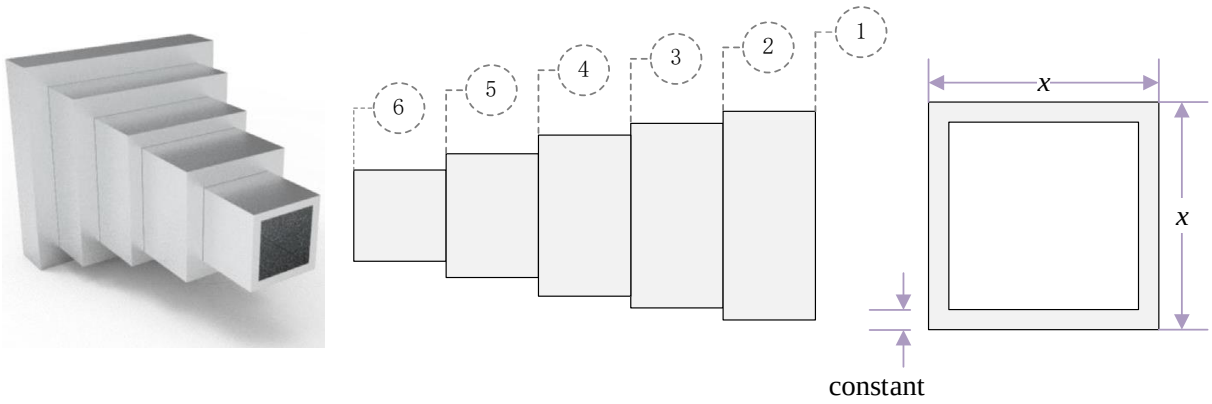

Figure 13. Cantilever beam structure.

Table 10. Statistical results of the cantilever beam design problem.

Algorithm	Optimum Variables					Optimum Cost
	x_1	x_2	x_3	x_4	x_5	
HAGSA	**5.9271**	**5.3962**	**4.5081**	**3.476**	**2.1726**	**1.3404**
AOA	6.4746	5.515	4.1138	3.7827	1.8724	1.3577
Gold-SA	5.7908	5.0142	4.9397	3.4175	2.5713	1.3562
ROA	5.8567	5.4316	4.4342	3.6542	2.1263	1.3418
AO	5.8219	5.4572	4.4551	3.5517	2.2198	1.342
SCA	5.781	5.5669	4.9992	3.5049	2.5094	1.3954
WOA	6.6424	5.0184	4.8451	3.0428	2.287	1.3626
FPA	5.7763	6.4239	4.6938	3.6501	1.6685	1.3861
DE	7.1323	4.9612	4.2559	3.3748	2.5797	1.3918
GA	6.5195	4.1943	5.7643	4.1847	2.2862	1.4320

5. Conclusions and Future Work

Considering the characteristic of AOA and Gold-SA, this paper proposes a hybrid optimization algorithm, namely HAGSA. First, Gold-SA is utilized to alleviate the shortcomings of AOA, such as low population diversity, premature convergence, and easy stagnation into local optimal solutions. Second, Levy flight and a new strategy called Brownian mutation are used to enhance the searchability of the hybrid algorithm.

We first used the CEC 2014 competition test suite to validate the optimization performance of HAGSA and its peers. The experimental results demonstrate that HAGSA outperforms other competitors in terms of optimization accuracy, convergence speed, robustness, and statistical difference. In addition, five industrial engineering design problems were carried out to test the ability of HAGSA to solve real-world problems. The experimental results also show that HAGSA is significantly better than its peers. Therefore, it is believed that HAGSA is a valuable method and can provide high-quality solutions to solve these kinds of problems. Although HAGSA has significant improvements over the original AOA and Gold-SA, its time consumption is a potential issue. This is because the BM strategy produces two candidate solutions and uses fitness evaluation to select the best solution. Thus, determining how to reduce the computational time under the premise of ensuring performance needs further research. In future works, we will: (1) improve the BM strategy to reduce the computational time without degrading HAGSA's performance; (2) seek to hybridize other MAs to improve AOA's optimization performance; and (3) apply HAGSA to solve combinatorial optimization problems (e.g., the traveling salesman problem, knapsack problem, and graph coloring problem). In addition, multilevel thresholding image segmentation would also be an interesting and meaningful research area.

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

Funding: This research was funded by the Innovative Research Project for Graduate Students of Hainan Province under grant No. Qhys2021-190, the National Natural Science Foundation of China under grant No. 11861030, the Hainan Provincial Natural Science Foundation of China under grants No. 621RC511 and No. 2019RC176, the Natural Science Foundation of Fujian Province under grant No. 2021J011128, the Sanming University Introduces High Level Talents to Start Scientific Research Funding Support Project under grant No. 20YG14, and the Sanming University National Natural Science Foundation Breeding Project under grant No. PYT2105.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding authors.

Acknowledgments: We acknowledge the anonymous reviewers for their constructive comments.

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

References

1. Esparza, E.R.; Calzada, L.A.Z.; Oliva, D.; Heidari, A.A.; Zaldivar, D.; Cisneros, M.P.; Foong, L.K. An efficient harris hawks-inspired image segmentation method. *Expert Syst. Appl.* **2020**, *155*, 113428. [CrossRef]
2. Liu, Q.; Li, N.; Jia, H.; Qi, Q.; Abualigah, L. Modified remora optimization algorithm for global optimization and multilevel thresholding image segmentation. *Mathematics* **2022**, *10*, 1014. [CrossRef]
3. Ewees, A.A.; Abualigah, L.; Yousri, D.; Sahlol, A.T.; Al-qaness, A.A.; Alshathri, S.; Elaziz, M.A. Modified artificial ecosystem-based optimization for multilevel thresholding image segmentation. *Mathematics* **2021**, *9*, 2363. [CrossRef]
4. Wang, S.; Liu, Q.; Liu, Y.; Jia, H.; Liu, L.; Zheng, R.; Wu, D. A hybrid SSA and SMA with mutation opposition-based learning for constrained engineering problems. *Comput. Intell. Neurosci.* **2021**, *2021*, 6379469. [CrossRef] [PubMed]
5. Houssein, E.H.; Mahdy, M.A.; Blondin, M.J.; Shebl, D.; Mohamed, W.M. Hybrid slime mould algorithm with adaptive guided differential evolution algorithm for combinatorial and global optimization problems. *Expert Syst. Appl.* **2021**, *174*, 114689. [CrossRef]
6. Wang, S.; Jia, H.; Liu, Q.; Zheng, R. An improved hybrid aquila optimizer and harris hawks optimization for global optimization. *Math. Biosci. Eng.* **2021**, *18*, 7076–7109. [CrossRef]
7. Wu, D.; Wang, S.; Liu, Q.; Abualigah, L.; Jia, H. An Improved Teaching-Learning-Based Optimization Algorithm with Reinforcement Learning Strategy for Solving Optimization Problems. *Comput. Intell. Neurosci.* **2022**, *2022*, 1535957. [CrossRef]
8. Zhang, H.; Wang, Z.; Chen, W.; Heidari, A.A.; Wang, M.; Zhao, X.; Liang, G.; Chen, H.; Zhang, X. Ensemble mutation-driven salp swarm algorithm with restart mechanism: Framework and fundamental analysis. *Expert Syst. Appl.* **2021**, *165*, 113897. [CrossRef]
9. Giovanni, L.D.; Pezzella, F. An improved genetic algorithm for the distributed and flexible Job-shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *200*, 395–408. [CrossRef]
10. Wu, B.; Zhou, J.; Ji, X.; Yin, Y.; Shen, X. An ameliorated teaching–learning-based optimization algorithm based study of image segmentation for multilevel thresholding using Kapur's entropy and Otsu's between class variance. *Inf. Sci.* **2020**, *533*, 72–107. [CrossRef]
11. Wang, S.; Jia, H.; Abualigah, L.; Liu, Q.; Zheng, R. An improved hybrid aquila optimizer and harris hawks algorithm for solving industrial engineering optimization problems. *Processes* **2021**, *9*, 1551. [CrossRef]
12. Lin, S.; Jia, H.; Abualigah, L.; Altalhi, M. Enhanced slime mould algorithm for multilevel thresholding image segmentation using entropy measures. *Entropy* **2021**, *23*, 1700. [CrossRef] [PubMed]
13. Su, H.; Zhao, D.; Yu, F.; Heidari, A.A.; Zhang, Y.; Chen, H.; Li, C.; Pan, J.; Quan, S. Horizontal and vertical search artificial bee colony for image segmentation of COVID-19 X-ray images. *Comput. Biol. Med.* **2021**, *142*, 105181. [CrossRef]
14. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
15. Khare, A.; Rangnekar, S. A review of particle swarm optimization and its applications in solar photovoltaic system. *Appl. Soft Comput.* **2013**, *13*, 2997–3006. [CrossRef]
16. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]

17. 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]
18. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
19. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [CrossRef]
20. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [CrossRef]
21. 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]
22. Abualigah, L.; Elaziz, M.A.; Sumari, P.; Geem, Z.; Gandomi, A.H. Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
23. Abualigah, L.; Yousri, D.; Abd, E.M.; Ewees, A.A. Aquila optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
24. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2015**, *27*, 495–513. [CrossRef]
25. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
26. Abualigah, L.; Diabat, A.; Mirjalili, S.; Elaziz, A.E.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
27. Tanyildizi, E.; Demir, G. Golden sine algorithm: A novel math-inspired algorithm. Golden sine algorithm: A novel math-inspired algorithm. *Adv. Electr. Comput. Eng.* **2017**, *17*, 71–78. [CrossRef]
28. Neggaz, N.; Houssein, E.H.; Hussain, K. An efficient henry gas solubility optimization for feature selection. *Expert Syst. Appl.* **2020**, *152*, 113364. [CrossRef]
29. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
30. Sun, P.; Liu, H.; Zhang, Y.; Meng, Q.; Tu, L.; Zhao, J. An improved atom search optimization with dynamic opposite learning and heterogeneous comprehensive learning. *Appl. Soft Comput.* **2021**, *103*, 107140. [CrossRef]
31. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl.-Based Syst.* **2021**, *191*, 105190. [CrossRef]
32. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
33. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
34. Slowik, A.; Kwasnicka, H. Evolutionary algorithms and their applications to engineering problems. *Neural Comput. Appl.* **2020**, *32*, 12363–12379. [CrossRef]
35. Hansen, N.; Ostermeier, A. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evol. Comput.* **2001**, *9*, 159–195. [CrossRef]
36. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
37. Azizi, M.; Talatahari, S. Improved arithmetic optimization algorithm for design optimization of fuzzy controllers in steel building structures with nonlinear behavior considering near fault ground motion effects. *Artif. Intell. Rev.* **2021**. [CrossRef]
38. Agushaka, J.O.; Ezugwu, A.E. Advanced arithmetic optimization algorithm for solving mechanical engineering design problems. *PLoS ONE* **2021**, *16*, 0255703. [CrossRef]
39. Wang, R.; Wang, W.; Xu, L.; Pan, J.; Chu, S. An adaptive parallel arithmetic optimization algorithm for robot path planning. *J. Adv. Transport.* **2021**, *2021*, 3606895. [CrossRef]
40. Abualigah, L.; Diabat, A.; Sumari, P.; Gandomi, A.H. A novel evolutionary arithmetic optimization algorithm for multilevel thresholding segmentation of COVID-19 CT images. *Processes* **2021**, *9*, 1155. [CrossRef]
41. Liu, Y.; Cao, B. A novel ant colony optimization algorithm with Levy flight. *IEEE Access* **2020**, *8*, 67205–67213. [CrossRef]
42. Iacca, G.; Junior, V.C.S.; Melo, V.V. An improved jaya optimization algorithm with Levy flight. *Expert Syst. Appl.* **2021**, *165*, 113902. [CrossRef]
43. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
44. Li, M.; Zhao, H.; Weng, X.; Han, T. A novel nature-inspired algorithm for optimization: Virus colony search. *Adv. Eng. Softw.* **2016**, *92*, 65–88. [CrossRef]
45. Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115665. [CrossRef]
46. Zhou, Y.; Wang, R.; Luo, Q. Elite opposition-based flower pollination algorithm. *Neurocomputing* **2016**, *188*, 294–310. [CrossRef]
47. Ewees, A.A.; Elaziz, M.A.; Houssein, E.H. Improved grasshopper optimization algorithm using opposition-based learning. *Expert Syst. Appl.* **2018**, *112*, 156–172. [CrossRef]
48. Yildiz, B.S.; Pholdee, N.; Bureerat, S.; Yildiz, A.R.; Sait, S.M. Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems. *Eng. Comput.* **2021**. [CrossRef]
49. Houssein, E.H.; Helmy, B.E.; Rezk, H.; Nassef, A.M. An efficient orthogonal opposition-based learning slime mould algorithm for maximum power point tracking. *Neural Comput. Appl.* **2022**, *34*, 3671–3695. [CrossRef]

50. Taheri, A.; Rahimizadeh, K.; Rao, R.V. An efficient balanced teaching-learning-based optimization algorithm with individual restarting strategy for solving global optimization problems. *Inf. Sci.* **2021**, *576*, 68–104. [CrossRef]
51. Ahmadianfar, I.; Heidari, A.A.; Gandomi, A.H.; Chu, X.; Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on runge kutta method. *Expert Syst. Appl.* **2021**, *181*, 115079. [CrossRef]
52. Cheng, Z.; Song, H.; Wang, J.; Zhang, H.; Chang, T.; Zhang, M. Hybrid firefly algorithm with grouping attraction for constrained optimization problem. *Knowl. Based Syst.* **2021**, *220*, 106937. [CrossRef]