Article

An Improved Novel Global Harmony Search Algorithm Based on Selective Acceptance

Hui Li ^{1,*}, Po-Chou Shih ^{1,*} , Xizhao Zhou ¹, Chunming Ye ¹ and Li Huang ^{1,2}

¹ Business School, University of Shanghai for Science and Technology, Shanghai 200093, China; 182911125@st.usst.edu.cn (H.L.); xzzhou@usst.edu.cn (X.Z.); yechm@usst.edu.cn (C.Y.); 181910086@st.usst.edu.cn (L.H.)

² School of Intelligent Manufacturing, Panzhihua University, Panzhihua 617000, China

* Correspondence: pcshih@usst.edu.cn; Tel.: +86-136-2183-4490 or +886-921-942-200

Received: 6 February 2020; Accepted: 6 March 2020; Published: 11 March 2020

Abstract: The novel global harmony search (NGHS) algorithm is proposed in 2010, and it is an improved harmony search (HS) algorithm which combines the particle swarm optimization (PSO) and the genetic algorithm (GA). One of the main differences between the HS and NGHS algorithms is that of using different mechanisms to renew the harmony memory (HM). In the HS algorithm, in each iteration, the new harmony is accepted and replaced the worst harmony in the HM while the fitness of the new harmony is better than the worst harmony in the HM. Conversely, in the NGHS algorithm, the new harmony replaces the worst harmony in the HM without any precondition. However, in addition to these two mechanisms, there is one old mechanism, the selective acceptance mechanism, which is used in the simulated annealing (SA) algorithm. Therefore, in this paper, we proposed the selective acceptance novel global harmony search (SANGHS) algorithm which combines the NGHS algorithm with a selective acceptance mechanism. The advantage of the SANGHS algorithm is that it balances the global exploration and local exploitation ability. Moreover, to verify the search ability of the SANGHS algorithm, we used the SANGHS algorithm in ten well-known benchmark continuous optimization problems and two engineering problems and compared the experimental results with other metaheuristic algorithms. The experimental results show that the SANGHS algorithm has better search ability than the other four harmony search algorithms in ten continuous optimization problems. In addition, in two engineering problems, the SANGHS algorithm also provided a competition solution compared with other state-of-the-art metaheuristic algorithms.

Keywords: metaheuristics; harmony search algorithm; selective acceptance

1. Introduction

In the past two decades, the interest in the research related to metaheuristic algorithms has increased significantly due to the fact that existing numerical methods do not allow for solving a number of complex optimization problems, such as discrete structural optimization [1], design of water distribution networks [2], and vehicle routing [3]. A metaheuristic algorithm is applied to simulate the randomness and rules in natural phenomena. For example, the genetic algorithm (GA) [4] imitates the process of the biological evolutionary and natural selection. The particle swarm optimization (PSO) [5] was inspired by the animal behavior, namely, the foraging behavior of bird flocking. In addition, there are many algorithms extending PSO and GA, such as the harmony search (HS) algorithm [6] and the honey bee mating optimization (HBMO) [7]. These algorithms have proved their ability to search for better solutions than numerical methods in extensive optimization problems [8–10]. However, compared with these algorithms, the HS algorithm which imitates an improvisation process of a harmony has two features. Firstly, it has fewer mathematical requirements and is easy to implement.

Secondly, each decision variable in a new solution vector could be generated independently by considering all of the existing solution vectors [11]. Due to these two features, the HS algorithm has stronger adaptability for many engineering problems and could generate competition results compared with the GA [11]. In addition, based on the above advantages, many modified versions of the HS algorithm are proposed to further improve the searching ability of the HS algorithm, such as the improved harmony search (IHS) algorithm [11], the self-adaptive global best harmony search (SGHS) algorithm [12], and the novel global harmony search (NGHS) algorithm [13]. These modified versions of the HS algorithm have better search ability than the original HS algorithm.

It is worth noting that, among these HS algorithms, the mechanism that is used to decide whether a new generated harmony (solution vector) is accepted and replaces the worst harmony in the harmony memory (HM) or not is different. The HM is a storage of the harmony. Specifically, the HS, IHS, and SGHS algorithms accept a new generated harmony only when it is better than the worst harmony in the current HM. This mechanism, we call it accept-the-better-only mechanism, reminded us of the tabu search (TS) algorithm [14] which also only accepts the better solution vector in the searching process. Conversely, in the NGHS algorithm, a new generated harmony is accepted and replaced the worst harmony in the HM without preconditions in each iteration. This mechanism, we call it accept-everything mechanism, is the same one used in the PSO and GA. In the PSO algorithm, in each iteration, after calculating the velocity of the particle, the particle will fly to the new position according to its velocity whether the fitness of the new position is better than the current position or not. However, while we were discussing these two acceptance mechanisms, we suddenly came up with an old algorithm, the simulated annealing (SA) algorithm [15]. In the SA algorithm, in each iteration, the new solution vector is accepted or not according to the acceptable probability (AP), which is calculated by the Boltzmann distribution. Thus, the SA algorithm inspired us to investigate whether the AP affects the search ability or not in the NGHS algorithm. Therefore, in this paper, we proposed the selective acceptance novel global harmony search (SANGHS) algorithm, which combined the selective acceptance mechanism with the NGHS algorithm. Moreover, we proposed a proportion formula to calculate the AP according to the convergence degree. To verify the searching performance of the SANGHS algorithm, we used the SANGHS algorithms in ten well-known optimization problems and two engineering problems and compared the experimental results with other 15 algorithms. The results revealed that the SANGHS algorithm outperformed other 15 algorithms.

The remainder of this paper is organized as follows. In Section 2, the HS, IHS, SGHS, and the NGHS algorithms are introduced. Section 3 describes the SANGHS algorithm. A considerable number of experiments were conducted to test the search ability of the proposed SANGHS algorithm, and the corresponding experiments results are analyzed in Section 4. Conclusions and future research directions are provided in Section 5.

2. HS, IHS, SGHS, and NGHS Algorithms

2.1. Harmony Search Algorithm

The harmony search was proposed by Geem in 2001 [6]. Unlike other metaheuristic algorithms used to simulate the reproductive or foraging processes in the biological world, such as PSO and GA, the harmony search algorithm is inspired by a jazz trio and simulates its improvisation process [6]. The HS algorithm introduces musical notes to represent decision variables in practical problems. The number of musical notes is determined by the number of decision variables. Each note is played by a musician. Harmony requires two or more notes to be played at the same time. Thereby, each harmony represents a set of decision values. The HS algorithm simulates this in an improvisation process; hence, if the new harmony is better than the worst harmony, the new harmony will be saved to the harmony memory, and the worst harmony will be discarded from the harmony memory. Conversely, the new harmony will not be committed, and the worst harmony will not be discarded.

The HS algorithm is superior to other heuristic algorithms in terms of solving continuous variable problems and combinatorial problems; for example, it is deemed to be more applicable than GA [6]. The HS algorithm works as follows [16].

- Step 1: Initialize the algorithm and problem parameters

The main parameters of the harmony search algorithm include the harmony memory size (HMS), the harmony memory considering rate (HMCR), the pitch adjusting rate (PAR), and the bandwidth (BW). HMS is the total number of harmonies, which can be stored in the harmony memory (HM). HMCR determines whether musicians create a new harmony based on a harmony in the HM or not. PAR determines whether musicians fine-tune the musical note of a harmony that was selected in the HM or not. The bandwidth BW is the range, in which musicians fine-tune the selected harmony. Another parameter is the maximum number of iterations (NI), which denotes how many improvisations musicians have made to define the best harmony.

In this step, the above parameters in the HS algorithm are determined. In addition, the current iteration (k) is set to 1. The HS algorithm is usually used in the D -dimensional optimization problem defined as minimization of $f(x)$ subject to $x_j \in [x_{jL}, x_{jU}]$ ($j = 1, 2, \dots, D$). Here, x_{jL} and x_{jU} are the lower and upper bounds for the decision variable x_j .

- Step 2: Initialize the harmony memory (HM)

The number of harmony vectors stored in the HM is equal to HMS. A harmony vector is a set of decision variables. The initial values of the decision variable $x_{i,j}^{k=0}$ ($i = 1, 2, \dots, HMS$) are generated by Equation (1). In Equation (1), r is a random number in the range $[0,1]$. Here, $x_{i,j}^k$ represents the j th decision variable in the i th harmony vector, when the current iteration is k . The harmony memory (HM) is shown in Equation (2):

$$x_{i,j}^0 = x_{jL} + r \times (x_{jU} - x_{jL}) \quad (1)$$

$$HM = \begin{bmatrix} x_{1,1}^0 & x_{1,2}^0 & \cdots & x_{1,D}^0 \\ x_{2,1}^0 & x_{2,2}^0 & \cdots & x_{2,D}^0 \\ \vdots & \vdots & \ddots & \vdots \\ x_{HMS,1}^0 & x_{HMS,2}^0 & \cdots & x_{HMS,D}^0 \end{bmatrix} \quad (2)$$

- Step 3: Improvise a new harmony

A new harmony vector $x_{new,j}^k = (x_{new,1}^k, x_{new,2}^k, \dots, x_{new,D}^k)$ is improvised by three mechanisms: memory consideration, pitch adjustment, and random selections. It is defined as in Algorithm 1.

Algorithm 1 The Improvisation of New Harmony in HS

```

1: For  $j = 1$  to  $D$  do
2:   If  $r_1 \leq HMCR$  then
3:      $x_{new,j}^k = x_{i,j}^k$  %memory consideration
4:   If  $r_2 \leq PAR$  then
5:      $x_{new,j}^k = x_{new,j}^k - BW + r_3 \times 2 \times BW$  %pitch adjustment
6:   If  $x_{new,j}^k > x_{jU}$  then
7:      $x_{new,j}^k = x_{jU}$ 
8:   Else if  $x_{new,j}^k < x_{jL}$  then
9:      $x_{new,j}^k = x_{jL}$ 
10:  End
11: End
12: Else
13:   $x_{new,j}^k = x_{jL} + r_4 \times (x_{jU} - x_{jL})$  % random selection
14: End
15: End

```

Here, $x_{new,j}^k$ is the j th decision variable of the new harmony improvised in the current iteration k . In the memory consideration mechanism, i of the $x_{i,j}^k$ is an integer random selected in the range of $[1, HMS]$; r_1, r_2, r_3 , and r_4 are random numbers in the region of $[0,1]$.

- Step 4: Update the harmony memory

If the new harmony vector $x_{new,j}^k = (x_{new,1}^k, x_{new,2}^k, \dots, x_{new,D}^k)$ has better fitness compared with the worse fitness in the harmony memory, the new harmony vector $x_{new,j}^k$ will replace the worse harmony vector.

- Step 5: Check the stopping iteration

If the current iteration is equal to the stopping iteration (the maximum number of iterations is NI), the computation is terminated. Otherwise, set $k = k + 1$ and go back to step 3.

2.2. Improved Harmony Search Algorithm

In the HS algorithm, the pitch adjustment mechanism is designed to improve the local search ability. However, the PAR and the BW are fixed in the search process, which means that the local search ability does not change in the early iteration and the final iteration. This is a reason why the HS algorithm has a limit to balance the global search ability and local search ability in the different stages of the search process. To mitigate this deficiency, Mahdavi et al. [11] proposed a new strategy to dynamically adjust PAR and BW. At the early stage of optimization, the small PAR values with the large BW enhance the ability of the global search and increase the diversity of the solution vectors. The large PAR and the small BW values improve the ability of the local search and make the algorithm converge to the optimal solution vector in the late stage of optimization. Therefore, PAR and BW are adjusted according to Equations (3) and (4), respectively:

$$PAR^k = PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{NI} \times k \quad (3)$$

$$BW^k = BW_{max} \times e^{\ln(\frac{BW_{min}}{BW_{max}}) \times k / NI} \quad (4)$$

In Equation (3), PAR^k represents the pitch adjusting rate, when the current iteration is k . PAR_{min} and PAR_{max} are the minimum and maximum pitch adjusting rates in the process of optimization, respectively. Additionally, in Equation (4), BW^k represents the bandwidth, when the current iteration is k . BW_{max} and BW_{min} are the maximum and the minimum bandwidth in the process of optimization.

2.3. Self-Adaptive Global Best Harmony Search Algorithm

In 2010, the SGHS algorithm was proposed by Pan et al. [12]. The SGHS algorithm and the HS algorithm differ between each other in three aspects. Firstly, the SGHS algorithm implies that the pitch adjusting mechanism should be computed before the memory consideration mechanism. In addition, the memory consideration mechanism only selects the best harmony in the HM other than selecting randomly. Thereby, the new harmony is improvised using Algorithm 2. Secondly, the parameters HMCR and PAR are dynamically adjusted by the normal distribution, and BW is adjusted in each iteration according to Equation (5). Furthermore, HMCR is generated by the mean $HMCR_{mean}$ that is in the range of $[0.9, 1.0]$ and the static standard deviation (0.01). PAR is generated by the mean PAR_{mean} that is in the region of $[0.0, 1.0]$ and the static standard deviation (0.05). Finally, the SGHS algorithm introduces another parameter named the learning period (LP) to determine $HMCR_{mean}$ and PAR_{mean} . In a learning period, if the new harmony replaces the worse harmony in the HM, when the iteration is k , $HMCR^k$ and PAR^k are recorded. When the learning period is satisfied, $HMCR_{mean}$ and PAR_{mean} are recalculated by averaging the recorded $HMCR^k$ and PAR^k , respectively, then the recorded $HMCR^k$ and PAR^k are discarded, and another new learning period will begin.

Algorithm 2 The Improvisation of New Harmony in SGHS

```

1: For  $j = 1$  to  $D$  do
2:   If  $r_1 \leq HMCR$  then
3:      $x_{new,j}^k = x_{i,j}^k - BW + r_2 \times 2 \times BW$  %pitch adjustment
4:     If  $x_{new,j}^k > x_{jU}$  then
5:        $x_{new,j}^k = x_{jU}$ 
6:     Else if  $x_{new,j}^k < x_{jL}$  then
7:        $x_{new,j}^k = x_{jL}$ 
8:     End
9:     If  $r_3 \leq PAR$  then
10:       $x_{new,j}^k = x_{B,j}^k$  %memory consideration
11:    End
12:  Else
13:     $x_{new,j}^k = x_{jL} + r_4 \times (x_{jU} - x_{jL})$  % random selection
14:  End
15: End

```

Here, $x_{B,j}^k$ represents the j th decision variable of the best harmony in the HM, when the iteration is k . The meaning of other variables in Algorithm 2 is the same as in Algorithm 1:

$$BW^k = \begin{cases} BW_{max} - \frac{BW_{max} - BW_{min}}{NI} \times 2k & \text{if } k < NI/2, \\ BW_{min} & \text{if } k \geq NI/2, \end{cases} \quad (5)$$

2.4. Noval Global Harmony Search Algorithm

In 2010, the NGHS algorithm was proposed by Zou et al. [13,17,18]. The algorithm proposed in their study combines the HS algorithm, GA, and the particle swarm optimization (PSO) algorithm. GA imitates a process of the biological evolution [4]. The genetic mutation probability (p_m) in GA was introduced to escape from the local optimum. Particle swarm optimization (PSO) imitates the foraging process of bird flocking [5]. The position update in the NGHS algorithm was inspired by the main characteristic of PSO implying that the position of the particles is affected by the best position of all particles [13].

Figure 1 illustrates the principle of the position update. $step_j^k = |x_{best,j}^k - x_{worse,j}^k|$ is an adaptive step of the j th decision variable in the current iteration k . The range between P and R is referred to as a truth region, which is centered at the global best harmony. In other words, the j th decision variable is generated randomly in a range of the truth region. In the early iteration, all solution vectors in the harmony memory are sporadic, and the truth region is wide, which means that the new harmony is generated randomly in a large range. It is beneficial to enhance the global search ability. In the late iteration, all solution vectors in the HM are inclined to move to the global optimum. Therefore, all solution vectors are concentrated within a small solution space, and the truth region is narrow, which enhances the local search ability of NGHS [13,17,18]. The NGHS algorithm works as follows.

Figure 1. Schematic diagram of position update.

- Step 1: Initialize the algorithm and problem parameters

The genetic mutation probability (p_m), the current iteration k , and the maximum number of iterations (NI) are set in this step. However, the harmony memory considering rate (HMCR), pitch adjusting rate (PAR), and the bandwidth (BW) are excluded from the NGHS algorithm.

- Step 2: Initialize the harmony memory (HM)

The initial values of the decision variable $x_{i,j}^{k=0}$ ($i = 1, \dots, HMS$) are generated by Equation (1). The harmony memory (HM) is shown in Equation (2).

- Step 3: Improvise a new harmony

A new harmony vector $x_{new,j}^k = (x_{new,1}^k, x_{new,2}^k, \dots, x_{new,D}^k)$ is improvised by Algorithm 3.

Algorithm 3 The Improvisation of New Harmony in NGHS

```

1: For  $j = 1$  to  $D$  do
2:    $x_R = 2 \times x_{best,j}^k - x_{worse,j}^k$ 
3:   If  $x_R > x_{jU}$  then
4:      $x_R = x_{jU}$ 
5:   Else if  $x_R < x_{jL}$  then
6:      $x_R = x_{jL}$ 
7:   End
8:    $x_{new,j}^k = x_{worse,j}^k + r_1 \times (x_R - x_{worse,j}^k)$  %position update
9:   If  $r_2 \leq p_m$  then
10:     $x_{new,j}^k = x_{jL} + r_3 \times (x_{jU} - x_{jL})$  % genetic mutation
11:  End
12: End

```

Here, $x_{best,j}^k$ and $x_{worse,j}^k$ are the best harmony and the worse harmony in the harmony memory, when the current iteration is k , respectively. r_1 , r_2 , and r_3 are random numbers in the range of $[0,1]$:

- Step 4: Update the harmony memory

The new harmony replaces the worse harmony in the HM, even if the worse harmony in the HM has better fitness than the new harmony.

- Step 5: Check the stopping iteration

If the current iteration k is equal to the stopping iteration (the maximum number of iterations is NI), the algorithm is terminated. Otherwise, set $k = k + 1$ and go back to step 3.

3. Selective Acceptance Novel Global Harmony Search Algorithm

Assad et al. [19] pointed that the effectiveness of evolutionary algorithms relies on the extent of balance between diversification, which implies the ability of an algorithm to explore the high performance regions of the search space, and intensification, which is the ability of an algorithm to search for better solution in the high performance regions during the optimizing process. In other words, the key to improving the searching ability of evolutionary algorithms is to enhance the extent of balance between diversification and intensification capabilities. It can be seen that, in Section 2, Step 3 (Improvise a new harmony) in the IHS, SGHS, or NGHS algorithm is gradually improved to balance the diversification and intensification capabilities. However, in Step 4, the mechanism of updating the HM is different in these algorithms. Accept-the-better-only mechanism is used in the HS, IHS, and the SGHS algorithm. The accept-everything mechanism is used in the NGHS algorithm. This difference inspired us to explore whether the mechanism in Step 4 could affect the balance of diversification and

intensification capabilities and improve the searching ability of the NGH algorithm. However, in the process of finding a good mechanism in Step 4, the mechanism that is used in the SA algorithm attracted our attention. The SA algorithm used Boltzmann distribution to calculate the acceptable probability dynamically in each iteration and determine to accept the new solution or not by the value of acceptable probability. Therefore, in this paper, we proposed the selective acceptance novel global harmony search (SANGHS) algorithm, which combined the selective acceptance mechanism with the NGH algorithm.

The main difference between the SANGHS and the NGH algorithms is that the SANGHS algorithm will accept and replace the worst harmony in the HM with the new harmony according to the selective acceptance mechanism. In this mechanism, the selective acceptance formula as shown in Equation (6), which is also dynamically adjusted in the optimization process as in the SA algorithm, is used to further balance the diversification and intensification capabilities. However, the selective acceptance formula is different from the Boltzmann distribution in the SA algorithm and does not need to use additional parameters. In Equation (6), AP^k represents the acceptable probability in the current iteration k . F_{new}^k represents the fitness value of the new harmony in the current iteration k . F_{best}^k and F_{worse}^k represent the fitness values of the best and the worst harmonies in the HM in the current iteration k , respectively. According to the selective acceptance formula in the SANGHS algorithm, if the fitness of the new harmony is better than the worst harmony in the HM, the AP is equal to 1. It means that the worst harmony will be replaced by the new harmony in the HM. However, if the fitness of the new harmony is worse than the worst harmony in the HM, the SANGHS algorithm will accept the new harmony according to the AP, which is calculated by the proportion formula. Proportion formula is the most important part of the selective acceptance formula. In proportion formula, $F_{worse}^k - F_{best}^k$ represents the convergence degree of all harmonies in the HM. In the early iterations, all harmonies in the HM are dispersed in a large solution space, and the value of $F_{worse}^k - F_{best}^k$ is a large value. Thus, it is more likely to accept a worse harmony and thereby expand the range of truth region in the SANGHS algorithm, which would contribute to improve the diversification capabilities. Conversely, in the late iterations, all harmonies in the HM have converged in a small solution space; the value of $F_{worse}^k - F_{best}^k$ is a small value. The HM is less likely to accept a worse harmony and thereby reduce the range of truth region, which makes the SANGHS algorithm have strong intensification capabilities:

$$AP^k = \begin{cases} 1 & \text{if } F_{new}^k \leq F_{worse}^k \\ \frac{(F_{worse}^k - F_{best}^k)}{(F_{new}^k - F_{best}^k)} & \text{if } F_{new}^k > F_{worse}^k \end{cases} \quad (6)$$

In general, the SANGHS algorithm contains the following steps:

- Step 1: Initialize the algorithm and problem parameters

The genetic mutation probability (p_m), the current iteration k , and the maximum iteration (NI) are set in this step.

- Step 2: Initialize the harmony memory (HM)

The initial values of the decision variable $x_{i,j}^{k=0}$ ($i = 1, 2, \dots, HMS$) are generated by Equation (1). The harmony memory (HM) is shown in Equation (2).

- Step 3: Improvise a new harmony

A new harmony vector $x_{new,j}^k = (x_{new,1}^k, x_{new,2}^k, \dots, x_{new,D}^k)$ is improvised by Algorithm 3. This step is the same as in the NGH algorithm.

- Step 4: Calculate AP

If the new harmony has a better fitness value than the worst harmony in the HM, AP is set to 1. Otherwise, AP is calculated by Equation (6).

- Step 5: Update the harmony memory

Firstly, generate a random number in $[0,1]$. If the random number is smaller than the AP, the new harmony replaces the worst harmony in the HM. Otherwise, the new harmony is discarded.

- Step 6: Check the stopping iteration

If the current iteration k is equal to the stopping iteration (the maximum number of iterations is NI), the algorithm is terminated. Otherwise, set $k = k + 1$ and go back to Step 3.

4. Experimental Results and Analysis

4.1. Computing Environment Settings

In this study, we used Python 3.5.1 (64-bit) as the compiler to implement the program to search for the trial solution. The solution-finding equipment comprised an Intel Core (TM) i5-8500 (3.0 GHz, 3.0 GHz) CPU, 8 GB of memory, and Windows 10 professional edition (64-bit) OS.

4.2. Ten Well-Known Benchmark Optimization Problems

To estimate the performance for the proposed acceptable probability mechanism, we considered ten well-known benchmark optimization problems [20,21] as follows. Among these problems, problems 1 to 4 are unimodal problems. Problems 5 to 10 are multimodal problems with several local optima, and the number of local optima is increased with the number of decision variables [12].

- (1) Sphere function (F1)

The Sphere function is defined as $\min f(x_i) = \sum_{i=1}^N x_i^2$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-100, 100]$.

- (2) Schwedel's problem 2.22 (F2)

This problem is defined as $\min f(x_i) = \sum_{i=1}^N |x_i| + \prod_{i=1}^N |x_i|$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-10, 10]$.

- (3) Axis Parallel Function (F3)

This problem is defined as $\min f(x_i) = \sum_{i=1}^N ix_i^2$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-5.12, 5.12]$.

- (4) Quartic Function (F4)

This problem is defined as $\min f(x_i) = \sum_{i=1}^N x_i^4$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-1.28, 1.28]$.

- (5) Ackley's function (F5)

The Ackley's function is defined as $\min f(x_i) = 20 + e - 20 \exp\left(-0.2 \sqrt{\sum_{i=1}^N \frac{x_i^2}{N}}\right) - \exp\left(\frac{\sum_{i=1}^N \cos(2\pi x_i)}{N}\right)$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-32, 32]$.

- (6) Rastrigin function (F6)

The Rastrigin function is defined as $\min f(x_i) = \sum_{i=1}^N (x_i^2 - 10 \cos(2\pi x_i) + 10)$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-5.12, 5.12]$.

- (7) Schwefel's problem 2.26 (F7)

This problem is defined as $\min f(x_i) = 418.9829N - \sum_{i=1}^N (x_i \sin(\sqrt{|x_i|}))$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-500, 500]$.

(8) Levy Function (F8)

This problem is defined as $\min f(x_i) = \sin^2(\pi w_1) - \sum_{i=1}^{N-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)] + (w_N - 1)^2 [1 + \sin^2(2\pi w_D)]$, where $w_i = 1 + \frac{x_i - 1}{4}$ and the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-10, 10]$.

(9) Bohachevsky function (F9)

This problem is defined as $\min f(x_i) = \sum_{i=1}^{N-1} [x_i^2 + 2x_{i+1}^2 - 0.3 \cos(3\pi x_i) - 0.4 \cos(4\pi x_{i+1}) + 0.7]$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-15, 15]$.

(10) Alpine 1 function (F10)

This problem is defined as $\min f(x_i) = \sum_{i=1}^N |x_i \sin(x_i) + 0.1x_i|$, where the global optimum is 0, and the search space of x_i ($i = 1, 2, \dots, N$) is $[-10, 10]$.

4.3. Comparison of Different Harmony Search Algorithms in Ten Benchmark Optimization Problems

To test the performance of the SANGHS algorithm for ten aforementioned optimization problems, we compared it with four harmony search algorithms, which were the HS, IHS, SGHS, and NGHS algorithms. The parameters of these algorithms are provided in Table 1 [12,17,22,23]. We tested three different dimensions equal to 10, 30, and 100. The maximum number of iterations (NI) is set to 30,000 with $D = 10$, 60,000 with $D = 30$, and 150,000 with $D = 100$. Each problem was tested in 30 independent experiments. The experimental results for all problems with three different dimensions obtained by the four harmony search algorithms and the SANGHS algorithm are shown in Tables 2–4. Mean and Std represent the average and the standard deviation of 30 independent experiments, respectively, as provided in Tables 2–4. Min and Max represent the minimum and maximum fitness values in 30 independent experiments.

Table 1. Parameters of compared harmony search (HS) algorithms in ten optimization problems.

Algorithm	HS	IHS	SGHS	NGHS	SANGHS
HMS ¹	5	5	5	5	5
HMCR ²	0.9	0.9	$HMCR_{mean} = 0.98$	–	–
PAR ³	0.3	$PAR_{min} = 0.01$ $PAR_{max} = 0.99$	$PAR_{mean} = 0.9$	–	–
BW ⁴	–	$BW_{min} = 0.0001$ $BW_{max} = (x_{jU} - x_{jL})/20$	$BW_{min} = 0.0005$ $BW_{max} = (x_{jU} - x_{jL})/10$	–	–
LP ⁵	–	–	100	–	–
P_m ⁶	–	–	–	0.005	0.005

¹ HMS: the harmony memory size; ² HMCR: the harmony memory considering rate; ³ PAR: the pitch adjusting rate;

⁴ BW: the bandwidth; ⁵ LP: the learning period; ⁶ P_m : the genetic mutation probability.

In Tables 2–4, several experimental results are given. Firstly, in all problems (F1 to F10) with three different dimensions ($D = 10, 30$, and 100), the SANGHS algorithm has smaller minimum (Min) and maximum (Max) fitness values than other four HS algorithms. For example, in problem 1 with $D = 10$, SANGHS has the smallest minimum fitness values (9.6086×10^{-111}) and the smallest maximum fitness values (1.3918×10^{-77}). Secondly, the SANGHS algorithm has a smallest average (Mean) fitness value. For example, in problem 1 with $D = 10$, SANGHS has the smallest average fitness values (4.6889×10^{-79}). This value is far less than the second one (8.2477×10^{-38}). The above two experimental results reveal that the SANGHS algorithm has a better search ability than other four harmony search algorithms. Lastly, in the above problems (F1 to F10) with $D = 10, 30$, and 100 , the SANGHS algorithm has a smallest standard deviation (Std) value. For example, in problem 1 with $D = 10$, SANGHS has the smallest standard deviation values (2.4976×10^{-78}), and this value is also far less than the second one (3.8833×10^{-37}). It means that the search ability of the SANGHS algorithm is more robust than other four harmony search algorithms.

Table 2. Experimental results of five algorithms for 10 problems with D = 10.

No.	Algorithm	Min	Max	Mean	Std	No.	Algorithm	Min	Max	Mean	Std
F1	HS	5.2096×10^{-8}	1.1157×10^{-6}	2.3562×10^{-7}	1.8902×10^{-7}	F6	HS	8.3112×10^{-6}	7.8214×10^{-5}	3.3076×10^{-5}	1.7982×10^{-5}
	IHS	6.7803×10^{-9}	2.3518×10^{-8}	1.3575×10^{-8}	3.7943×10^{-9}		IHS	1.1191×10^{-6}	4.0212×10^{-6}	2.6004×10^{-6}	6.4709×10^{-7}
	SGHS	6.8178×10^{-12}	1.4106×10^{-10}	3.2650×10^{-11}	2.5506×10^{-11}		SGHS	1.1716×10^{-9}	3.8150×10^{-1}	1.2717×10^{-2}	6.8481×10^{-2}
	NGHS	1.0698×10^{-45}	2.1604×10^{-36}	8.2477×10^{-38}	3.8833×10^{-37}		NGHS	0.0000×10^0	0.0000×10^0	0.0000×10^0	0.0000×10^0
	SANGHS	9.6086×10^{-111}	1.3918×10^{-77}	4.6889×10^{-79}	2.4976×10^{-78}		SANGHS	0.0000×10^0	0.0000×10^0	0.0000×10^0	0.0000×10^0
F2	HS	3.2109×10^{-4}	1.5971×10^{-3}	7.7726×10^{-4}	2.7372×10^{-4}	F7	HS	1.2731×10^{-4}	2.8515×10^{-2}	1.0739×10^{-3}	5.0957×10^{-3}
	IHS	2.0318×10^{-4}	3.6754×10^{-4}	2.9700×10^{-4}	3.2031×10^{-5}		IHS	1.2728×10^{-4}	1.2728×10^{-4}	1.2728×10^{-4}	4.0700×10^{-10}
	SGHS	7.3565×10^{-6}	1.8055×10^{-5}	1.2027×10^{-5}	2.7866×10^{-6}		SGHS	1.2728×10^{-4}	2.9759×10^{-4}	1.3517×10^{-4}	3.1705×10^{-5}
	NGHS	3.8140×10^{-27}	5.1898×10^{-21}	2.3612×10^{-22}	9.4731×10^{-22}		NGHS	1.2728×10^{-4}	1.2728×10^{-4}	1.2728×10^{-4}	8.5769×10^{-14}
	SANGHS	1.4267×10^{-82}	7.2433×10^{-54}	3.5052×10^{-55}	1.4044×10^{-54}		SANGHS	1.2728×10^{-4}	1.2728×10^{-4}	1.2728×10^{-4}	4.4839×10^{-14}
F3	HS	1.4468×10^{-7}	2.1934×10^{-6}	8.0446×10^{-7}	4.2722×10^{-7}	F8	HS	2.0732×10^{-8}	3.4077×10^{-7}	8.0031×10^{-8}	6.4199×10^{-8}
	IHS	3.1842×10^{-8}	9.8796×10^{-8}	6.2079×10^{-8}	1.8245×10^{-8}		IHS	3.3581×10^{-9}	1.0209×10^{-8}	6.7715×10^{-9}	1.7956×10^{-9}
	SGHS	2.7923×10^{-11}	7.2398×10^{-10}	1.7525×10^{-10}	1.4531×10^{-10}		SGHS	1.7990×10^{-12}	5.0988×10^{-11}	1.3622×10^{-11}	1.1253×10^{-11}
	NGHS	4.4709×10^{-46}	3.5344×10^{-39}	2.3102×10^{-40}	7.0896×10^{-40}		NGHS	1.4998×10^{-32}	1.4998×10^{-32}	1.4998×10^{-32}	5.4738×10^{-48}
	SANGHS	3.8636×10^{-113}	4.3798×10^{-83}	2.3947×10^{-84}	9.1559×10^{-84}		SANGHS	1.4998×10^{-32}	1.4998×10^{-32}	1.4998×10^{-32}	5.4738×10^{-48}
F4	HS	2.2666×10^{-17}	3.0752×10^{-14}	7.0687×10^{-15}	8.2181×10^{-15}	F9	HS	1.1130×10^{-6}	2.8915×10^{-5}	7.0153×10^{-6}	5.3517×10^{-6}
	IHS	9.5725×10^{-18}	6.5490×10^{-17}	2.9787×10^{-17}	1.3161×10^{-17}		IHS	3.0585×10^{-7}	9.1446×10^{-7}	5.8587×10^{-7}	1.4754×10^{-7}
	SGHS	2.5924×10^{-23}	4.7975×10^{-21}	6.9747×10^{-22}	1.1480×10^{-21}		SGHS	3.2077×10^{-10}	4.1452×10^{-9}	1.3097×10^{-9}	8.5282×10^{-10}
	NGHS	1.9010×10^{-86}	7.5124×10^{-67}	2.5042×10^{-68}	1.3485×10^{-67}		NGHS	5.7250×10^{-49}	2.0380×10^{-36}	7.8672×10^{-38}	3.6838×10^{-37}
	SANGHS	1.7493×10^{-169}	9.2806×10^{-124}	3.2904×10^{-125}	1.6640×10^{-124}		SANGHS	2.0444×10^{-113}	1.4776×10^{-69}	4.9253×10^{-71}	2.6524×10^{-70}
F5	HS	1.7118×10^{-4}	8.0222×10^{-4}	4.6344×10^{-4}	1.5000×10^{-4}	F10	HS	1.1494×10^{-4}	1.8959×10^{-3}	4.9240×10^{-4}	3.4263×10^{-4}
	IHS	1.0584×10^{-4}	1.6838×10^{-4}	1.4686×10^{-4}	1.5429×10^{-5}		IHS	2.0995×10^{-5}	3.2351×10^{-4}	5.9924×10^{-5}	7.1295×10^{-5}
	SGHS	2.8256×10^{-6}	1.2840×10^{-5}	6.6281×10^{-6}	2.4208×10^{-6}		SGHS	5.6467×10^{-7}	1.8978×10^{-5}	5.9487×10^{-6}	5.4367×10^{-6}
	NGHS	6.6613×10^{-15}	3.8636×10^{-14}	1.8267×10^{-14}	7.1015×10^{-15}		NGHS	1.4433×10^{-15}	1.5821×10^{-14}	6.1063×10^{-15}	3.0077×10^{-15}
	SANGHS	6.6613×10^{-15}	2.7978×10^{-14}	1.4122×10^{-14}	4.8842×10^{-15}		SANGHS	8.3267×10^{-16}	1.0658×10^{-14}	4.9442×10^{-15}	2.5741×10^{-15}

Table 3. Experimental results of five algorithms for 10 problems with D = 30.

No.	Algorithm	Min	Max	Mean	Std	No.	Algorithm	Min	Max	Mean	Std
F1	HS	5.0536×10^{-1}	6.4171×10^0	3.3124×10^0	1.5861×10^0	F6	HS	2.6666×10^{-2}	2.0462×10^0	5.0288×10^{-1}	6.0096×10^{-1}
	IHS	2.1207×10^{-7}	4.9163×10^{-7}	3.5819×10^{-7}	7.1968×10^{-8}		IHS	4.0261×10^{-5}	4.6986×10^0	1.3178×10^0	1.0352×10^0
	SGHS	1.5978×10^{-9}	1.4781×10^{-8}	4.8841×10^{-9}	3.1202×10^{-9}		SGHS	7.9528×10^{-7}	9.9496×10^{-1}	4.8052×10^{-2}	1.8483×10^{-1}
	NGHS	1.2081×10^{-17}	6.1930×10^{-15}	6.6153×10^{-16}	1.3269×10^{-15}		NGHS	0.0000×10^0	4.7713×10^{-12}	1.7243×10^{-13}	8.5424×10^{-13}
	SANGHS	2.3793×10^{-66}	2.4404×10^{-37}	8.1347×10^{-39}	4.3807×10^{-38}		SANGHS	0.0000×10^0	0.0000×10^0	0.0000×10^0	0.0000×10^0
F2	HS	4.5085×10^{-2}	2.0043×10^{-1}	7.6210×10^{-2}	3.7203×10^{-2}	F7	HS	8.1324×10^0	5.2395×10^1	2.0713×10^1	9.5235×10^0
	IHS	1.7655×10^{-3}	4.4854×10^{-3}	2.3905×10^{-3}	4.6485×10^{-4}		IHS	3.8186×10^{-4}	1.7516×10^0	6.3827×10^{-2}	3.1460×10^{-1}
	SGHS	8.1564×10^{-5}	1.9686×10^{-4}	1.5091×10^{-4}	2.9858×10^{-5}		SGHS	3.8277×10^{-4}	7.8715×10^{-2}	1.7508×10^{-2}	1.9421×10^{-2}
	NGHS	3.2148×10^{-10}	8.7017×10^{-9}	2.0345×10^{-9}	1.7972×10^{-9}		NGHS	3.8183×10^{-4}	3.8183×10^{-4}	3.8183×10^{-4}	8.1680×10^{-12}
	SANGHS	1.3727×10^{-44}	7.3787×10^{-28}	2.4752×10^{-29}	1.3243×10^{-28}		SANGHS	3.8183×10^{-4}	3.8183×10^{-4}	3.8183×10^{-4}	1.0918×10^{-13}
F3	HS	1.5858×10^{-3}	4.8414×10^{-3}	2.9867×10^{-3}	7.1956×10^{-4}	F8	HS	5.1397×10^{-5}	1.6213×10^{-4}	1.0240×10^{-4}	2.2064×10^{-5}
	IHS	8.1295×10^{-6}	1.3227×10^{-3}	1.7068×10^{-4}	3.1223×10^{-4}		IHS	1.1278×10^{-7}	4.7712×10^{-6}	8.2210×10^{-7}	1.0175×10^{-6}
	SGHS	8.4500×10^{-9}	5.2306×10^{-8}	2.1700×10^{-8}	1.1042×10^{-8}		SGHS	3.0351×10^{-10}	1.4228×10^{-9}	7.3604×10^{-10}	2.3859×10^{-10}
	NGHS	3.4329×10^{-19}	3.4291×10^{-16}	1.8533×10^{-17}	6.0640×10^{-17}		NGHS	9.6185×10^{-20}	7.8833×10^{-17}	8.7557×10^{-18}	1.8706×10^{-17}
	SANGHS	9.6420×10^{-70}	2.0955×10^{-48}	7.4888×10^{-50}	3.7563×10^{-49}		SANGHS	1.4998×10^{-32}	1.4998×10^{-32}	1.4998×10^{-32}	5.4738×10^{-48}
F4	HS	9.2705×10^{-10}	4.8468×10^{-9}	3.1521×10^{-9}	9.6498×10^{-10}	F9	HS	7.5871×10^{-3}	3.2014×10^0	1.0965×10^0	1.0189×10^0
	IHS	2.3766×10^{-15}	2.5488×10^{-14}	8.1771×10^{-15}	4.4175×10^{-15}		IHS	8.6839×10^{-6}	2.1440×10^0	5.3391×10^{-1}	5.8893×10^{-1}
	SGHS	1.9509×10^{-20}	1.6666×10^{-18}	4.2398×10^{-19}	4.2829×10^{-19}		SGHS	3.1568×10^{-8}	1.2410×10^{-7}	7.9116×10^{-8}	2.4154×10^{-8}
	NGHS	1.8030×10^{-40}	6.0162×10^{-35}	4.2247×10^{-36}	1.1809×10^{-35}		NGHS	4.2058×10^{-19}	1.3275×10^{-14}	1.0821×10^{-15}	2.8017×10^{-15}
	SANGHS	2.6142×10^{-94}	3.9928×10^{-69}	1.3328×10^{-70}	7.1669×10^{-70}		SANGHS	6.1353×10^{-65}	6.3460×10^{-33}	2.3282×10^{-34}	1.1387×10^{-33}
F5	HS	1.5649×10^{-2}	1.7072×10^0	6.7498×10^{-1}	3.9769×10^{-1}	F10	HS	2.4004×10^{-2}	1.9367×10^{-1}	1.0981×10^{-1}	4.6542×10^{-2}
	IHS	3.2084×10^{-4}	1.3089×10^0	3.6401×10^{-1}	4.4719×10^{-1}		IHS	2.1852×10^{-2}	9.7546×10^{-2}	5.3262×10^{-2}	1.9894×10^{-2}
	SGHS	2.5647×10^{-5}	4.6874×10^{-5}	3.2244×10^{-5}	5.0051×10^{-6}		SGHS	1.2359×10^{-5}	1.4067×10^{-3}	1.0235×10^{-4}	2.4394×10^{-4}
	NGHS	7.4210×10^{-10}	1.4597×10^{-8}	4.6791×10^{-9}	3.5652×10^{-9}		NGHS	1.0805×10^{-10}	3.0740×10^{-9}	8.6347×10^{-10}	6.8488×10^{-10}
	SANGHS	3.8636×10^{-14}	2.9798×10^{-13}	6.2321×10^{-14}	4.5175×10^{-14}		SANGHS	4.4409×10^{-15}	2.1261×10^{-14}	1.1208×10^{-14}	4.4960×10^{-15}

Table 4. Experimental results of five algorithms for 10 problems with D = 100.

No.	Algorithm	Min	Max	Mean	Std	No.	Algorithm	Min	Max	Mean	Std
F1	HS	8.7644×10^3	1.6480×10^4	1.2203×10^4	1.5403×10^3	F6	HS	2.2313×10^2	2.9442×10^2	2.5974×10^2	2.1788×10^1
	IHS	9.1067×10^3	1.4800×10^4	1.2428×10^4	1.0779×10^3		IHS	1.9487×10^2	2.8076×10^2	2.4522×10^2	2.0779×10^1
	SGHS	8.7681×10^{-1}	2.6029×10^0	1.3666×10^0	4.2374×10^{-1}		SGHS	1.5762×10^0	8.5459×10^0	4.0935×10^0	1.5444×10^0
	NGHS	4.8032×10^{-4}	1.0576×10^{-3}	7.3694×10^{-4}	1.6006×10^{-4}		NGHS	1.5643×10^{-3}	2.0192×10^0	4.7179×10^{-1}	6.1547×10^{-1}
	SANGHS	5.1238×10^{-23}	7.0729×10^{-15}	4.3965×10^{-16}	1.3365×10^{-15}		SANGHS	0.0000×10^0	1.3856×10^{-13}	6.5133×10^{-15}	2.4832×10^{-14}
F2	HS	5.0156×10^1	7.3451×10^1	6.1118×10^1	4.9536×10^0	F7	HS	4.5600×10^3	7.0794×10^3	5.8498×10^3	5.8361×10^2
	IHS	5.1843×10^1	6.8563×10^1	6.0248×10^1	4.3861×10^0		IHS	4.4499×10^3	6.5893×10^3	5.5203×10^3	4.9043×10^2
	SGHS	8.1631×10^{-2}	4.1138×10^{-1}	2.4863×10^{-1}	8.1339×10^{-2}		SGHS	8.9612×10^0	3.0848×10^1	1.7210×10^1	5.3463×10^0
	NGHS	9.9950×10^{-3}	1.6514×10^{-2}	1.3619×10^{-2}	1.5409×10^{-3}		NGHS	3.9466×10^{-3}	1.1539×10^{-1}	1.6436×10^{-2}	2.4948×10^{-2}
	SANGHS	2.2360×10^{-12}	2.2218×10^{-3}	1.9072×10^{-4}	5.5550×10^{-4}		SANGHS	1.2728×10^{-3}	1.2728×10^{-3}	1.2728×10^{-3}	4.8314×10^{-12}
F3	HS	9.4087×10^2	1.5941×10^3	1.1913×10^3	1.6577×10^2	F8	HS	2.4107×10^1	3.3570×10^1	2.9300×10^1	2.8794×10^0
	IHS	9.2595×10^2	1.5737×10^3	1.1718×10^3	1.4788×10^2		IHS	2.3183×10^1	3.6516×10^1	2.9135×10^1	3.1786×10^1
	SGHS	6.4869×10^{-5}	5.3603×10^{-3}	8.8126×10^{-4}	1.2161×10^{-3}		SGHS	4.4458×10^{-7}	4.0334×10^{-4}	2.2728×10^{-5}	8.2115×10^{-5}
	NGHS	3.3424×10^{-5}	1.1541×10^{-4}	6.4369×10^{-5}	1.9242×10^{-5}		NGHS	1.4748×10^{-6}	9.2715×10^{-6}	3.6238×10^{-6}	1.4865×10^{-6}
	SANGHS	7.2707×10^{-22}	1.4056×10^{-13}	7.1676×10^{-15}	2.6377×10^{-14}		SANGHS	4.8189×10^{-31}	9.5129×10^{-17}	3.2482×10^{-18}	1.7065×10^{-17}
F4	HS	1.3307×10^{-1}	2.6181×10^{-1}	1.9974×10^{-1}	3.4152×10^{-2}	F9	HS	7.0005×10^2	1.1205×10^3	9.0061×10^2	1.0327×10^2
	IHS	1.3301×10^{-1}	2.8520×10^{-1}	2.0499×10^{-1}	3.3111×10^{-2}		IHS	7.3029×10^2	1.1882×10^3	9.1884×10^2	1.0628×10^2
	SGHS	4.0943×10^{-15}	1.1631×10^{-14}	7.5656×10^{-15}	2.0292×10^{-15}		SGHS	2.5895×10^{-1}	4.6678×10^0	2.0793×10^0	1.0991×10^0
	NGHS	8.7725×10^{-16}	7.2065×10^{-15}	2.3953×10^{-15}	1.7313×10^{-15}		NGHS	3.4107×10^{-4}	1.4657×10^{-3}	7.7456×10^{-4}	2.5196×10^{-4}
	SANGHS	6.1762×10^{-46}	4.0280×10^{-31}	1.4178×10^{-32}	7.2252×10^{-32}		SANGHS	2.8823×10^{-22}	1.6936×10^{-15}	1.6201×10^{-16}	3.2735×10^{-16}
F5	HS	1.0843×10^1	1.2640×10^1	1.1775×10^1	4.3937×10^{-1}	F10	HS	1.9740×10^1	2.6059×10^1	2.2513×10^1	1.6612×10^0
	IHS	1.1199×10^1	1.2544×10^1	1.1869×10^1	3.3008×10^{-1}		IHS	1.6309×10^1	2.6621×10^1	2.1674×10^1	2.1494×10^0
	SGHS	3.2093×10^{-2}	6.5191×10^{-1}	2.4740×10^{-1}	1.3506×10^{-1}		SGHS	9.7570×10^{-2}	2.0103×10^{-1}	1.4311×10^{-1}	2.6382×10^{-2}
	NGHS	2.4194×10^{-3}	5.1098×10^{-3}	3.4730×10^{-3}	5.5824×10^{-4}		NGHS	3.1071×10^{-3}	7.3450×10^{-3}	5.1084×10^{-3}	1.1414×10^{-3}
	SANGHS	2.1347×10^{-12}	2.4401×10^{-9}	2.9843×10^{-10}	6.6605×10^{-10}		SANGHS	1.7615×10^{-13}	8.6512×10^{-4}	3.8048×10^{-5}	1.5973×10^{-4}

In addition, to further explore the difference in the performance of the SANGHS algorithm and other four harmony search algorithms, the experiment results were also analyzed through the Wilcoxon rank-sum test [24,25] as shown in Table 5. In Table 5, the p -value, which is greater than 0.05, is bold. Based on the p -value, the SANGHS algorithm is significantly better than the other four HS algorithms in all problems with three different dimensions, except in problem 10 (F10) with $D = 10$ (6.3886×10^{-2}).

Figure 2 shows the typical convergence graphs of five different algorithms for ten problems with $D = 30$. In Figure 2a–d which are unimodal problems, as the number of iterations increases, the SANGHS algorithm converges faster than other four harmony search algorithms. It means that the SANGHS algorithm has a strong local search ability in unimodal problems. In Figure 2e,f,h,j which are multimodal problems, we can easily observe that the SANGHS algorithm is more effective in escaping from the local optima and have a better convergence ability compared with the other four harmony search algorithms. In other words, the SANGHS algorithm has a better balance between the global search ability and the local search ability in multimodal problems.

Table 5. p -value of Wilcoxon rank-sum test in ten problems.

D	No.	SANGHS vs. HS	SANGHS vs. IHS	SANGHS vs. SGHS	SANGHS vs. NGHS
10	F1	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F2	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F3	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F4	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F5	7.8779×10^{-12}	7.8779×10^{-12}	7.8779×10^{-12}	1.1293×10^{-02}
	F6	6.0589×10^{-13}	6.0589×10^{-13}	6.0589×10^{-13}	–
	F7	9.1371×10^{-12}	9.1371×10^{-12}	9.1371×10^{-12}	8.1647×10^{-04}
	F8	6.0589×10^{-13}	6.0589×10^{-13}	6.0589×10^{-13}	–
	F9	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F10	1.5071×10^{-11}	1.5071×10^{-11}	1.5071×10^{-11}	6.3886×10^{-02}
30	F1	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F2	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F3	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F4	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F5	1.4019×10^{-11}	1.4019×10^{-11}	1.4019×10^{-11}	1.4019×10^{-11}
	F6	6.0589×10^{-13}	6.0589×10^{-13}	6.0589×10^{-13}	2.2246×10^{-12}
	F7	1.3835×10^{-11}	1.3835×10^{-11}	1.3835×10^{-11}	1.3661×10^{-11}
	F8	6.0589×10^{-13}	6.0589×10^{-13}	6.0589×10^{-13}	6.0589×10^{-13}
	F9	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F10	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
100	F1	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F2	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F3	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F4	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F5	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F6	1.0142×10^{-11}	1.0142×10^{-11}	1.0142×10^{-11}	1.0142×10^{-11}
	F7	1.4688×10^{-11}	1.4688×10^{-11}	1.4688×10^{-11}	1.4688×10^{-11}
	F8	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F9	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}
	F10	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}	1.5099×10^{-11}

Figure 2. Typical convergence graph of five different algorithms for ten problems with $D = 30$. (a) problem 1; (b) problem 2; (c) problem 3; (d) problem 4; (e) problem 5; (f) problem 6; (g) problem 7; (h) problem 8; (i) problem 9; (j) problem 10.

4.4. Two Benchmarks' Engineering Optimization Problems

In this section, to verify the searching performance of the SANGHS algorithm further, we used the SANGHS algorithm to solve two benchmark engineering problems and compared the experimental results with previous references [26–38].

4.4.1. Tension/Compression Spring Design Optimization Problem

The first engineering problem is the tension/compression spring design engineering problem. The main purpose of this problem is to minimize the spring weight by determining three design variables when four constraints are satisfied. Three design variables are the wire diameter (d), mean coil diameter (D), and the number of active coils (N). Four constraints are about the shear stress, surge frequency, and the minimum deflection. One objective function, three design variables, and four constraints are shown in Figure 3.

We used the HS, IHS, SGHS, NGHS, and the SANGHS algorithms to search the best result for this problem. Table 6 shows the parameters of these five HS algorithms. The best results obtained by these five algorithms are shown in Table 7. Moreover, as shown in Table 7, to further verify the searching performance of SANGHS algorithm, the best result of SANGHS algorithm is also compared with other state-of-the-art algorithms which have been applied to solve this problem.

Figure 3. Tension/compression spring design.

Table 6. Parameters of compared harmony search (HS) algorithms in the tension/compression spring design engineering problem.

Algorithm	HS	IHS [11]	SGHS	NGHS	SANGHS
HMS ¹	4	4	4	4	4
HMCR ²	0.8	0.95	HMCR _{mean} = 0.98	–	–
PAR ³	0.3	PAR _{min} = 0.35 PAR _{max} = 0.99	PAR _{mean} = 0.9	–	–
BW ⁴	–	BW _{min} = 0.0005 BW _{max} = 0.05	BW _{min} = 0.0005 BW _{max} = (x _{jU} - x _{jL})/10	–	–
LP ⁵	–	–	100	–	–
P _m ⁶	–	–	–	0.006	0.008
NI ⁷	50,000	50,000	50,000	50,000	50,000

¹ HMS: the harmony memory size; ² HMCR: the harmony memory considering rate; ³ PAR: the pitch adjusting rate; ⁴ BW: the bandwidth; ⁵ LP: the learning period; ⁶ P_m: the genetic mutation probability; ⁷ NI: the maximum number of iterations.

Table 7. Experimental results in the tension/compression spring design engineering problem.

Algorithm	d	D	N	Optimal Cost
HHO [26]	0.05179639	0.35930536	11.13885900	0.0126654
GWO [27]	0.05169000	0.35673700	11.28885000	0.0126663
RO [28]	0.05137000	0.34909600	11.76279000	0.0126786
CDE [29]	0.05160900	0.35471400	11.41083100	0.0126702
CPSO [30]	0.05172800	0.35764400	11.24454300	0.0126747
GA3 [31]	0.05198900	0.36396500	10.89052200	0.0126810
ES [32]	0.05164300	0.35536000	11.39792600	0.0126980
GSA [33]	0.05027600	0.32368000	13.52541000	0.0127022
MFO [34]	0.05199446	0.36410932	10.86842186	0.0126669
CWCA [35]	0.05170910	0.35710734	11.27082577	0.0126716
LAFBA [36]	0.05166300	0.35607400	11.33340000	0.0126720
HS	0.05152035	0.35266977	11.53040649	0.0126659
IHS	0.05129291	0.34725895	11.86597566	0.0126683
SGHS	0.05149684	0.35211045	11.56429455	0.0126659
NGHS	0.05148116	0.35173679	11.58707075	0.0126660
SANGHS	0.05162828	0.35525732	11.37510196	0.0126653

In Table 7, it can be seen that, compared with the HS, IHS, SGHS, and the NGHS algorithm, the SANGHS algorithm provides a best fitness value for the tension/compression spring design engineering problem. It is worth noting that the SANGHS algorithm also provides a very competitive result compared with the other state-of-the-art algorithms in previous works.

4.4.2. Welded Beam Design Optimization Problem

The second problem is the welded beam design engineering problem. In this problem, there are four variables that need to be determined that are the thickness of the weld (x_1), the length of the clamped bar (x_2), the height of the bar (x_3), and the thickness of the bar (x_4). The main purpose of this problem is to minimize the manufacturing cost when seven constraints are satisfied. The constraints include limits on the shear stress, bending stress, buckling load and end deflection, and several size constraints. The welded beam and its design variables are shown in Figure 4. These seven constraints and the objective function of manufacturing cost are defined as follows.

$$\begin{aligned}
 &\text{Consider} && x = [x_1, x_2, x_3, x_4], \\
 &\text{Minimize} && f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2), \\
 &\text{Subject to} && g_1(x) = \tau(x) - \tau_{\max} \leq 0, \\
 & && g_2(x) = \sigma(x) - \sigma_{\max} \leq 0, \\
 & && g_3(x) = \delta(x) - \delta_{\max} \leq 0, \\
 & && g_4(x) = x_1 - x_4 \leq 0, \\
 & && g_5(x) = P - P_c(x) \leq 0, \\
 & && g_6(x) = 0.125 - x_1 \leq 0, \\
 & && g_7(x) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0, \\
 &\text{Variable range} && 0.10 \leq x_1 \leq 2.00, 0.1 \leq x_2 \leq 10, 0.10 \leq x_3 \leq 10.0, 0.10 \leq x_4 \leq 2 \\
 &\text{where} && \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, \\
 & && M = P\left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}, \sigma(x) = \frac{6PL}{x_4x_2^2}, \delta(x) = \frac{4PL^2}{Ex_3^2x_4}, \\
 & && J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}, P_c(x) = \frac{4.013E\sqrt{\frac{x_2^2x_4^3}{2g}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \\
 & && P = 6000\text{lb}, L = 14\text{in}, \delta_{\max} = 0.25\text{in}, E = 30 \times 10^6\text{psi}, \\
 & && G = 12 \times 10^6\text{psi}, \tau_{\max} = 13600\text{psi}, \sigma_{\max} = 30000\text{psi},
 \end{aligned}$$

Figure 4. Welded beam design.

We also used the HS, IHS, SGHS, NGHS, and the SANGHS algorithms to search the best result for this problem. Table 8 shows the parameters of these five HS algorithms. The best results obtained by these five algorithms are shown in Table 9. Moreover, as shown in Table 9, to further verify the searching performance of SANGHS algorithm, the best result of the SANGHS algorithm is also compared with other state-of-the-art algorithms that have been applied to solve this problem.

In Table 9, it can be seen that, compared with the HS, IHS, SGHS, and the NGHS algorithm, the SANGHS algorithm provides a minimum manufacturing cost for the welded beam design engineering problem. It is worth noting that the SANGHS algorithm also provides a very competitive result compared with other state-of-the-art algorithms in previous works.

Table 8. Parameters of compared harmony search (HS) algorithms in the welded beam design engineering problem.

Algorithm	HS	IHS [11]	SGHS	NGHS	SANGHS
HMS ¹	8	8	8	8	8
HMCR ²	0.8	0.95	HMCR _{mean} = 0.98	–	–
PAR ³	0.3	PAR _{min} = 0.45 PAR _{max} = 0.99	PAR _{mean} = 0.9	–	–
BW ⁴	–	BW _{min} = 0.0005 BW _{max} = 2.5	BW _{min} = 0.0005 BW _{max} = (x _{ju} – x _{jl})/10	–	–
LP ⁵	–	–	100	–	–
P _m ⁶	–	–	–	0.015	0.014
NI ⁷	200,000	200,000	200,000	200,000	200,000

¹ HMS: the harmony memory size; ² HMCR: the harmony memory considering rate; ³ PAR: the pitch adjusting rate;

⁴ BW: the bandwidth; ⁵ LP: the learning period; ⁶ P_m: the genetic mutation probability; ⁷ NI: the maximum number of iterations.

Table 9. Experimental results in the welded beam design engineering problem.

Algorithms	h	l	t	b	Optimal Cost
HHO [26]	0.20403900	3.53106100	9.02746300	0.20614700	1.73198796
RO [28]	0.20368700	3.52846700	9.00423300	0.20724100	1.73534560
CDE [29]	0.20313700	3.54299800	9.03349800	0.20617900	1.73346218
ES [32]	0.19974200	3.61206000	9.03750000	0.20608200	1.73729731
GSA [33]	0.18212900	3.85697900	10.00000000	0.20237600	1.87994703
LAFBA [36]	0.18470619	3.64265569	9.13489736	0.20525405	1.72870000
GA1 [37]	0.24890000	6.17300000	8.17890000	0.25330000	2.43116000
RANDOM [38]	0.45750000	4.73130000	5.08530000	0.66000000	4.11855504
DAVID [38]	0.24340000	6.25520000	8.29150000	0.24440000	2.38410685
SIMPLEX [38]	0.27920000	5.62560000	7.75120000	0.27960000	2.53072583
APPOX [38]	0.24440000	6.21890000	8.29150000	0.24440000	2.38145339
HS	0.20570604	3.47100746	9.03659628	0.20573099	1.72489123
IHS	0.20573721	3.47045570	9.03644813	0.20574489	1.72494527
SGHS	0.20547623	3.47814281	9.03104463	0.20598515	1.72646956
NGHS	0.20573640	3.47040079	9.03647449	0.20573644	1.72487683
SANGHS	0.20572954	3.47049090	9.03662388	0.20572964	1.72485245

5. Conclusions

In this paper, we presented a novel algorithm, called the SANGHS algorithm, which combines the NGHS algorithm with a selective acceptance mechanism. In the SANGHS algorithm, the acceptable probability (AP) is generated by a selective acceptance formula in each iteration. Based on the value of AP, the SANGHS algorithm will determine whether to accept the worse trial solution or not. Moreover, to verify the searching performance of the SANGHS algorithm, we used the SANGHS algorithm in ten well-known benchmark continuous optimization problems and two engineering

problems and compared the experimental results with other state-of-the-art metaheuristic algorithms. The experimental results provide two findings that are worth noting.

Firstly, it was found that the SANGHS algorithm could result in improved searching performance as opposed to the NGHS algorithm. In the SANGHS algorithm, the selective acceptance mechanism dynamically adjusts the acceptable probability according to the selective acceptance formula. The most important part of the selective acceptance formula is proportion formula, and the main concept of the proportion formula is convergence degree in the HM. Hence, in Equation (6), $F_{worse}^k - F_{best}^k$ represents the convergence degree of all harmonies in the HM. In the early iterations, all solutions have not yet converged in the HM. Therefore, the value of $F_{worse}^k - F_{best}^k$ will be a larger value, and then there is a larger acceptable probability to accept a worse harmony. In the next iteration, there is a higher probability to generate a new harmony in a larger truth region. In other words, there is a strong global exploration ability in the early iterations. However, in the late iterations, the value of $F_{worse}^k - F_{best}^k$ will be a smaller value while all solutions have converged in the HM. Therefore, there is a smaller acceptable probability to accept a worse harmony. In the next iteration, there is a higher probability to generate a new harmony in a smaller truth region. In other words, there is a strong local exploitation ability in the later iterations. According to the experimental results, the proposed proportion mechanism can balance the global exploration and local exploitation ability in the NGHS algorithm.

Finally, in ten considered benchmark optimization problems with three different dimensions, based on the results of the Wilcoxon rank-sum test and the convergence graphs, the searching performance of the SANGHS algorithm was better than other four HS algorithms significantly. Especially in Figure 2, we can easily observe that the SANGHS algorithm had the better searching performance and convergence ability than other algorithms in all problems. Conversely, the HS, IHS, and the SGHS algorithms easily fall within the local optimum, and unlikely escape it. Moreover, in two engineering optimization problems, the SANGHS algorithm could not only search for a better solution compared with other four HS algorithms, but also could provide a competition result to compare with other state-of-the-art algorithms. In conclusion, the SANGHS algorithm is a more efficient and effective algorithm.

Author Contributions: Conceptualization, P.-C.S. and H.L.; Formal analysis, H.L. and L.H.; Methodology, H.L.; Software, H.L.; Supervision, X.Z. and C.Y.; Validation, H.L. and P.-C.S.; Writing—original draft, H.L.; Writing—review and editing, P.-C.S. and H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, Grant No. 71840003.

Acknowledgments: The authors would like to thank all the reviewers for their constructive comments.

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

References

1. Lee, K.S.; Geem, Z.W.; Lee, S.H.; Bae, K.W. The harmony search heuristic algorithm for discrete structural optimization. *Eng. Optim.* **2005**, *37*, 663–684. [\[CrossRef\]](#)
2. Geem, Z.W. Optimal cost design of water distribution networks using harmony search. *Eng. Optim.* **2006**, *38*, 259–280. [\[CrossRef\]](#)
3. Geem, Z.W.; Lee, K.S.; Park, Y. Application of Harmony Search to Vehicle Routing. *Am. J. Appl. Sci.* **2005**, *12*, 1552–1557. [\[CrossRef\]](#)
4. Metawaa, N.; Hassana, M.K.; Elhoseny, M. Genetic algorithm based model for optimizing bank lending decisions. *Expert Syst. Appl.* **2017**, *80*, 75–82. [\[CrossRef\]](#)
5. Chen, K.H.; Chen, L.F.; Su, C.T. A new particle swarm feature selection method for classification. *J. Intell. Inf. Syst.* **2014**, *42*, 507–530. [\[CrossRef\]](#)
6. Lee, K.S.; Geem, Z.W. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [\[CrossRef\]](#)
7. Afshar, A.; Bozorg Haddad, O.; Mariño, M.A.; Adams, B.J. Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. *J. Franklin Inst.* **2007**, *344*, 452–462. [\[CrossRef\]](#)

8. Sanchis, J.; Martínez, M.A.; Blasco, X. Integrated multiobjective optimization and a priori preferences using genetic algorithms. *Inf. Sci.* **2008**, *178*, 931–951. [\[CrossRef\]](#)
9. Zhang, J.; Zhang, Y.; Gao, R. Genetic algorithms for optimal design of vehicle suspensions. In Proceedings of the IEEE International Conference on Engineering of Intelligent Systems, Islamabad, Pakistan, 22–23 April 2006.
10. Marinakis, Y.; Marinaki, M.; Dounias, G. Particle swarm optimization for pap-smear diagnosis. *Expert Syst. Appl.* **2008**, *35*, 1645–1656. [\[CrossRef\]](#)
11. Mahdavi, M.; Fesanghary, M.; Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **2007**, *188*, 1567–1579. [\[CrossRef\]](#)
12. Pan, Q.K.; Suganthan, P.N.; Tasgetiren, M.F.; Liang, J.J. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Appl. Math. Comput.* **2010**, *216*, 830–848. [\[CrossRef\]](#)
13. Zou, D.; Gao, L.; Li, S.; Wu, J.; Wang, X. A novel global harmony search algorithm for task assignment problem. *J. Syst. Softw.* **2010**, *83*, 1678–1688. [\[CrossRef\]](#)
14. Nowicki, E.; Smutnicki, C. A fast tabu search algorithm for the permutation flow-shop problem. *Eur. J. Oper. Res.* **1996**, *91*, 160–175. [\[CrossRef\]](#)
15. Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. Equation of state calculations by fast computing machines. *J. Chem. Phys.* **1953**, *21*, 1087–1092. [\[CrossRef\]](#)
16. Chiu, C.Y.; Shih, P.C.; Li, X. A dynamic adjusting novel global harmony search for continuous optimization problems. *Symmetry* **2018**, *10*, 337. [\[CrossRef\]](#)
17. Zou, D.; Gao, L.; Wu, J.; Li, S. Novel global harmony search algorithm for unconstrained problems. *Neurocomputing* **2010**, *73*, 3308–3318. [\[CrossRef\]](#)
18. Zou, D.; Gao, L.; Wu, J.; Li, S.; Li, Y. A novel global harmony search algorithm for reliability problems. *Comput. Ind. Eng.* **2010**, *58*, 307–316. [\[CrossRef\]](#)
19. Assad, A.; Deep, K. A two-phase harmony search algorithm for continuous optimization. *Comput. Intell.* **2017**, *33*, 1038–1075. [\[CrossRef\]](#)
20. Jamil, M.; Yang, X.S. A literature survey of benchmark functions for global optimisation problems. *Int. J. Math. Model. Numer. Optim.* **2013**, *4*, 150–194. [\[CrossRef\]](#)
21. Laguna, M.; Martí, R. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *J. Glob. Optim.* **2005**, *33*, 235–255. [\[CrossRef\]](#)
22. Tavakoli, S.; Valian, E.; Mohanna, S. Feedforward neural network training using intelligent global harmony search. *Evol. Syst.* **2012**, *3*, 125–131. [\[CrossRef\]](#)
23. Valian, E.; Tavakoli, S.; Mohanna, S. An intelligent global harmony search approach to continuous optimization problems. *Appl. Math. Comput.* **2014**, *232*, 670–684. [\[CrossRef\]](#)
24. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. [\[CrossRef\]](#)
25. García, S.; Molina, D.; Lozano, M.; Herrera, F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 Special Session on Real Parameter Optimization. *J. Heuristics* **2009**, *12*, 15–617. [\[CrossRef\]](#)
26. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Futur. Gener. Comput. Syst.* **2019**, *97*, 849–872. [\[CrossRef\]](#)
27. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
28. Kaveh, A.; Khayatizad, M. A new meta-heuristic method: Ray Optimization. *Comput. Struct.* **2012**, *112*–113, 283–294. [\[CrossRef\]](#)
29. Huang, F.; Wang, L.; He, Q. An effective co-evolutionary differential evolution for constrained optimization. *Appl. Math. Comput.* **2007**, *186*, 340–356. [\[CrossRef\]](#)
30. He, Q.; Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intell.* **2007**, *20*, 89–99. [\[CrossRef\]](#)
31. Coello, C.A.C.; Montes, E.M. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv. Eng. Inform.* **2002**, *16*, 193–203. [\[CrossRef\]](#)
32. Mezura-Montes, E.; Coello, C.A.C. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *Int. J. Gen. Syst.* **2008**, *37*, 443–473. [\[CrossRef\]](#)
33. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [\[CrossRef\]](#)
34. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. -Based Syst.* **2015**, *89*, 228–249. [\[CrossRef\]](#)

35. Heidari, A.A.; Abbaspour, A.R.; Jordehi, A.R. An efficient chaotic water cycle algorithm for optimization tasks. *Neural Comput. Appl.* **2017**, *28*, 57–85. [[CrossRef](#)]
36. Li, Y.; Li, X.; Liu, J.; Ruan, X. An Improved Bat Algorithm Based on Lévy Flights and Adjustment Factors. *Symmetry* **2019**, *7*, 925. [[CrossRef](#)]
37. Deb, K. Optimal design of a welded beam via genetic algorithms. *AIAA J.* **1991**, *29*, 2013–2015. [[CrossRef](#)]
38. Ragsdell, K.M.; Phillips, D.T. Optimal Design of a Class of Welded Structures Using Geometric Programming. *Am. Soc. Mech. Eng.* **1975**, 1021–1025. [[CrossRef](#)]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).