Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 653749, 16 pages
http://dx.doi.org/10.1155/2013/653749

Research Article

A Novel Self-Adaptive Harmony Search Algorithm

Kaiping Luo

School of Economics & Management, Beihang University, Beijing 100191, China

Correspondence should be addressed to Kaiping Luo; lkp09257@buaa.edu.cn

Received 19 July 2013; Revised 4 October 2013; Accepted 8 October 2013

Academic Editor: Julián López-Gómez

The harmony search algorithm is a music-inspired optimization technology and has been successfully applied to diverse scientific and engineering problems. However, like other metaheuristic algorithms, it still faces two difficulties: parameter setting and finding the optimal balance between diversity and intensity in searching. This paper proposes a novel, self-adaptive search mechanism for optimization problems with continuous variables. This new variant can automatically configure the evolutionary parameters in accordance with problem characteristics, such as the scale and the boundaries, and dynamically select evolutionary strategies in accordance with its search performance. The new variant simplifies the parameter setting and efficiently solves all types of optimization problems with continuous variables. Statistical test results show that this variant is considerably robust and outperforms the original harmony search (HS), improved harmony search (IHS), and other self-adaptive variants for large-scale optimization problems and constrained problems.

1. Introduction

Harmony search (HS) is a relatively new metaheuristic optimization algorithm [1] that imitates the improvisation process of Jazz musicians as shown in Figure 1. Each musician is analogous to each decision variable; the audience's aesthetics are analogous to the objective function; the pitch range of a musical instrument corresponds to the value range of the decision variables; the musicians' improvisations correspond to local and global search schemes; musical harmony at certain time corresponds to a solution vector at certain iteration. In general, a musician's improvisation follows three rules [2]: (1) playing any one pitch from his/her memory, (2) playing an adjacent pitch to one pitch from his/her memory, and (3) playing a random pitch from the possible sound range. The HS algorithm mimics these rules and converts them into corresponding evolutionary strategies: (1) choosing any one value from the HS memory (defined as memory considerations), (2) choosing a value adjacent to one from the HS memory (defined as pitch adjustments), and (3) choosing a random value from the possible value range (defined as randomisation). The evolutionary rules in the HS algorithm are effectively directed using the following parameters: (1) harmony memory size (HMS, i.e., the number of solutions in harmony memory), (2) harmony memory considering rate (HMCR, distinctly HMCR $\in [0, 1]$), (3) pitch adjusting rate (PAR, distinctly PAR $\in [0, 1]$), (4) fine-tuning bandwidth (Bw, i.e., the adjustment range of a variable), and (5) the number of improvisations (NI, i.e., the maximum number of iterations).

The values of these parameters, particularly three key parameters, HMCR, PAR, and Bw, affect the performances of HS. However, parameter setting is problem dependent and difficult, like other evolutionary computational methods, such as genetic algorithms (GA), simulated annealing (SA), particle swarm optimization (PSO), and ant colony optimization (ACO). To alleviate the burden of manually finding the best parameter values, this work proposes a natural and novel approach to parameter setting based on the improvisation practice of musicians.

For any algorithm that involves the operation of searching randomly, the most difficult is to find an appropriate balance between diversity and intensity in the process of searching. when emphasising diversity, it is easy to conduct a slower convergence and even divergence; when emphasising intensity, it is easy to conduct premature convergence and become trapped in a local optimal solution. The HS algorithm is no exception. Thus, this paper also proposes a novel self-adaptive search mechanism.

The rest of this paper is organized as follows. Section 2 presents a concise review of related previous work. Section 3

describes the novel self-adaptive harmony search (NSHS) algorithm. Section 4 introduces a test suite composed of twelve benchmark functions. Section 5 analyzes the robustness of the proposed variant. Section 6 confirms the superiority of the novel variant by the comparison test. The last section makes a conclusion.

2. Review of Related Previous Work

2.1. Original Harmony Search Algorithm. The original HS algorithm developed by [1] is carried out as follows.

Step 1 (initialise the problem and algorithm parameters). First, the optimization problem is specified as follows:

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t. } x_i \in [lb_i, ub_i] \quad (i = 1, 2, \ldots, n), \tag{1}$$

where $f(\mathbf{x})$ is the objective function, $\mathbf{x}$ is a vector consisting of n decision variables, and lb_i and ub_i are the lower and upper bounds of variable x_i, respectively. The original HS algorithm basically considers the objective function only. If the problem has equality and/or inequality constraints and a solution vector violates any of them, either the resulting infeasible solution will be abandoned or its objective function value will have a certain penalty score added.

Second, the parameters HMS, HMCR, PAR, Bw, and NI must be valued, and the initial harmony memory (HM) must be generated. Consider

$$\mathrm{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_n^2 & f(x^2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_n^{\mathrm{HMS}} & f(x^{\mathrm{HMS}}) \end{bmatrix}. \tag{2}$$

Step 2 (improve a new harmony vector by improvisation). A new solution $x^{\mathrm{new}} = (x_1^{\mathrm{new}}, x_2^{\mathrm{new}}, \ldots, x_n^{\mathrm{new}})$, termed the harmony vector, is generated based on memory considerations, pitch adjustments, and randomisation. Consider

$$x_i^{\mathrm{new}} \longleftarrow \begin{cases} x_i \in \left\{x_i^1, x_i^2, \ldots, x_i^{\mathrm{HMS}}\right\}; & \text{with probability} \\ & \mathrm{HMCR} * (1 - \mathrm{PAR}) \\ x_i \cdot (1 + r * \mathrm{Bw}); & \text{with probability} \\ \quad r \sim \mathrm{U}[-1, 1] & \mathrm{HMCR} * \mathrm{PAR} \\ x_i \in [lb_i, ub_i]; & \text{with probability} \\ & 1 - \mathrm{HMCR}. \end{cases} \tag{3}$$

Step 3 (assess the new harmony vector and update the harmony memory). In this step, the merit of the new harmony vector is assessed based on its objective function value. If the new vector is better than the worst harmony vector in the current HM, the new vector is included in the HM and the existing worst vector is excluded from the HM.

Step 4 (check the stopping criterion). Repeat Steps 2~3 until the termination criterion is satisfied; that is, the number of iterations reaches NI.

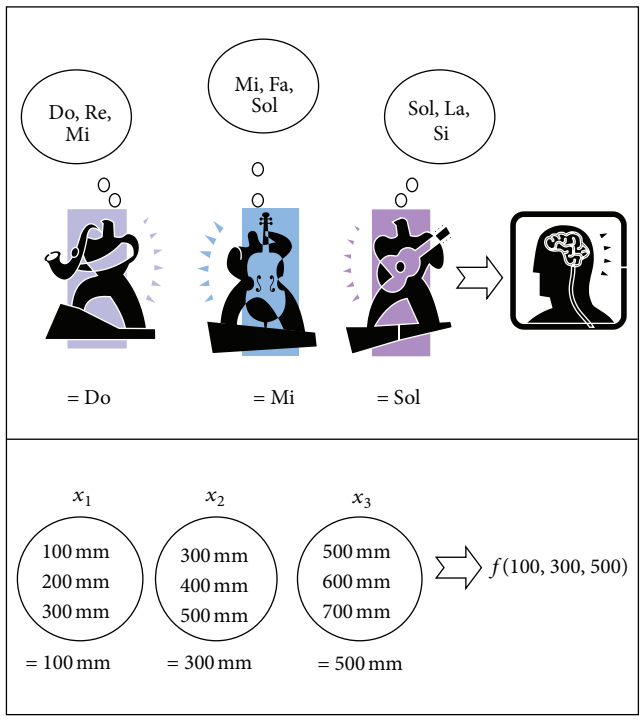


Figure 1: The Analogy between Improvisation and Optimization (from Geem [3]).

The structure of the HS algorithm is much easier. Its core evolutionary process includes only three operations: choosing, adjusting, and randomising. However, the metaheuristic algorithm handles intensification and diversification. Its intensification and diversification are represented by HMCR and PAR, respectively, and its superiority is confirmed by the existence of a large number of successful applications to diverse scientific and engineering problems, including timetabling [4], the 0-1 knapsack problem [5], power system design [6], transmission network planning [7], flow-shop scheduling [8], structural design [9], water network design [10], soil stability analysis [11], ecological conservation [12], and vehicle routing [13].

However, the major disadvantage of the HS algorithm is that its performance depends on the parameter setting and static parameter values are not conducive to balancing intensification and diversification.

2.2. Variants Based on Parameter Setting. To improve the performance of the original HS algorithm or alleviate the burden of manually finding the best parameter values, some variants based on parameter setting have been proposed.

2.2.1. Improved Harmony Search (IHS) Proposed by Mahdavi et al. [14]. In this variant, PAR and Bw are dynamically adjusted by the following formulae:

$$\mathrm{PAR}(k) = \mathrm{PAR}_{\min} + (\mathrm{PAR}_{\max} - \mathrm{PAR}_{\min}) * \frac{k}{\mathrm{NI}}, \tag{4}$$

$$\mathrm{Bw}(k) = \mathrm{Bw}_{\max} * \exp\left(\frac{k}{\mathrm{NI}} * \ln\left(\frac{\mathrm{Bw}_{\min}}{\mathrm{Bw}_{\max}}\right)\right), \tag{5}$$

where PAR_{min} and PAR_{max} indicate the minimum and maximum pitch adjusting rates, respectively. k and NI denote the kth-iteration and the maximum number of iterations, respectively. Bw_{min}, and Bw_{max} indicate the minimum and maximum bandwidths.

In fact, this variant does not alleviate the difficulty of parameter setting, because users have to provide suitable values of PAR_{min}, PAR_{max}, Bw_{min} and Bw_{max}. Furthermore, using (4), the pitch adjusting rate gradually increases from PAR_{min} to PAR_{max} through an iterative process, which is rather inconsistent with the observation that a larger iteration number usually generates a more perfect harmony and requires a smaller tuning probability.

Later, Omran and Mahdavi [15] abandoned the Bw parameter by replacing direct adjustment with the use of a random variable for the best harmony vector as shown in (6), calling this new variant the global-best harmony search (GHS). Consider

$$x_i^{\text{new}} \longleftarrow x_k^{\text{best}} \quad (i = 1, 2, \ldots, n;\ k \in \{1, 2, \ldots, n\}) \quad \text{with probability HMCR} \cdot \text{PAR}. \tag{6}$$

As Wang and Huang [16] noted, the name of this variant is misused and the method can easily cause premature convergence. In fact, the most serious defect in the GHS variant is that if the domain of each variable is different, that is, $[lb_i, ub_i] \neq [lb_k, ub_k], (i, k = 1, 2, \ldots, n; i \neq k)$, then the infeasible solution will frequently be generated. Thus, Pan et al. [17] corrected this serious drawback using the following equation:

$$x_i^{\text{new}} \longleftarrow x_i^{\text{best}} \quad (i = 1, 2, \ldots, n) \quad \text{with probability HMCR} \cdot \text{PAR}. \tag{7}$$

2.2.2. The Self-Adaptive Harmony Search Proposed by Wang and Huang [16]. Wang and Huang recognised the shortcomings of IHS and GHS. They used the same modification to dynamically adapt PAR values during the search process, but the value of PAR is decreased linearly with time. Thus, their variant needs are still the values of PAR_{min} and PAR_{max}. The innovation of the variant is to adjust the value of each variable according to the maximal and minimal values in the harmony memory (HM) and without the Bw parameter again. Consider

$$x_i^{\text{new}} \longleftarrow \begin{cases} x_i + r \cdot \left(\max\limits_{h \in [1, \text{HMS}]} \text{HM}(h, i) - x_i \right); & \text{if } r \geq 0.5 \\ x_i - r \cdot \left(x_i - \min\limits_{h \in [1, \text{HMS}]} \text{HM}(h, i) \right); & \text{if } r < 0.5 \end{cases} \quad r \sim \text{U}[0, 1]. \tag{8}$$

Although this adjustment enables each variable to avoid violating its boundary constraint, the bandwidth is larger; therefore, the convergence is very slow and can even stop.

2.2.3. The Self-Adaptive Harmony Search Proposed by Pan et al. Pan et al. [18] proposed a local-best harmony search algorithm with dynamic subpopulations (DLHS). In the variant, the whole harmony memory is equally divided into m subgroups. Moreover, to keep population diversity, the small-sized sub-HMs are regrouped once the period R is reached. In the variant, HMCR and PAR are from a parameter set list (PSL) with length Len, which is randomly generated by the initial value with a uniform distribution in the range PAR $\in [0, 1]$ and HMCR $\in [0.9, 1]$ and then updated once the list is empty. In the variant, the value of the parameter Bw is also dynamically changed:

$$\text{Bw}(k) = \begin{cases} \text{Bw}_{\max} - \dfrac{\text{Bw}_{\max} - \text{Bw}_{\min}}{\text{NI}} * 2k; & \text{if } 2k < \text{NI}, \\ \text{Bw}_{\min}; & \text{otherwise.} \end{cases} \tag{9}$$

Although the variant takes the advantage of each local optimum, it consumes too much time as a cost. Thus, Pan et al. [17] proposed a self-adaptive global best harmony search algorithm (SGHS) based on the GHS algorithm. In the new variant, the value of HMCR (PAR) is normally distributed in the range [0.9, 1] ([0, 1]) with mean HMCRm (PARm) and standard deviation 0.01 (0.05). The initial values of HMCRm and PARm are given by the user. In a fixed learning period (LP), the values of HMCRm and PARm do not change; after LP, the values are recalculated by averaging all HMCR and PAR values recorded during the previous LP. In the variant, Bw is still determined by (9).

Obviously, the two variants require extra jobs to set the values of more parameters than the basic HS algorithm and other variants. Moreover, it is irrational that the value of PAR still varies in the interval [0, 1] regardless of the number of iterations because a successful search should gradually converge and the value of PAR should be decreased with time to prevent oscillation.

These defects inspired me to investigate how the key parameters of the HS algorithm do not depend on the problem's feature again.

2.3. Variants Based on Hybridisation with Other Metaheuristics. To improve the performance of the original HS, many researchers have hybridised the algorithm with other meta-heuristics, such as GA, PSO, and ACO [19]. This hybridisation can be categorised into two approaches [20]: (1) integrating some components of other meta-heuristic algorithms into the HS structure, and conversely, (2) integrating some HS components into other meta-heuristic algorithm structures. Perhaps hybrid methods are better than single meta-heuristics.

However, in my opinion, hybridisation will depart from the essential nature of the original heuristic and the resulting mimicry will also be nondescript. Thus, the correct way to investigate this should be by refining the essence of the imitated objects. This idea inspired me to improve the search mechanism of the original HS algorithm by capturing the deep essence of musical improvisation.

3. A Novel Self-Adaptive Harmony Search Algorithm

3.1. Design of the Control Parameters. The HS algorithm includes five parameters: harmony memory size (HMS), harmony memory considering rate (HMCR), pitch adjusting rate (PAR), fine-tuning bandwidth (Bw), and the number of improvisations (NI). Among these parameters, HMCR, PAR, and Bw are crucial for the convergence of HS.

It should be noted that each improvisation of a musician is always accompanied by fine-tuning until the improvisation is finished. That is, it is necessary to adjust each variable value before the HS algorithm is ended. Therefore, it is reasonable that the value of PAR ought to equal one and senseless to consider this parameter again. For this reason, the proposed self-adaptive variant does not use the PAR parameter.

The value of HMCR in the basic HS algorithm is fixed at its initialisation and does not change until the search ends. As the value of HMCR decreases, harmony memory is used less efficiently, the algorithm converges much more slowly, and more time is consumed. As the value of HMCR increases, harmony memory is used more efficiently, the method converges more rapidly, less time is consumed, and the HS is easily trapped in a local optimum. To enhance the power of the HS algorithm, HMCR is usually valued in the interval $[0.9, 0.99]$. However, the reason for this is not revealed [21]. In fact, musicians depend on their memory more for intricate than simple music. This fact inspired me to construct the HMCR parameter based on the dimension of the problem to be solved. That is, the dimension of the problem is analogous to the complexity of the music,

$$\mathrm{HMCR} = 1 - \frac{1}{n+1}, \tag{10}$$

where n denotes the dimension of the problem to be solved. Clearly, $0.5 < \mathrm{HMCR} < 1$; as the number of dimensions of the problem increases, the value of HMCR increases and harmony memory is used more frequently, and vice versa.

It should be noted that musicians fine-tune their playing throughout their improvisation, and the tuning range is wider at the beginning and narrower as the music is completed. This phenomenon inspired me to construct a dynamic fine-tuning bandwidth bw_i for each variable x_i, with bounds lb_i and ub_i and iteration number k:

$$bw_i(k) = \begin{cases} \dfrac{ub_i - lb_i}{100} * \left(1 - \dfrac{k}{\mathrm{NI}}\right) & \\ (k = 1, 2, \ldots, \mathrm{NI}); & \text{if } f\mathrm{std} > 0.0001 \\ 0.0001; & \text{otherwise,} \end{cases} \tag{11}$$

where $f\mathrm{std} = \mathrm{std}(f(\mathbf{x}^1), f(\mathbf{x}^2), \ldots, f(\mathbf{x}^{\mathrm{HMS}}))$ denotes the standard deviation for the objective function values in the current HM. The value of bw_i decreases gradually as the iteration number increases and increases as the range of each decision variable increases. The domain of each decision variable corresponds to the pitch range of each musical instrument; the gradual decrease in the adjustment range as the iteration number increases corresponds to the gradual decrease of fine-tuning bandwidth with increasing improvisation number. $(ub - lb)/100$ is analogous to the unit of fine-tuning bandwidth, and 0.0001 represents the minimal adjustment. $f\mathrm{std} \le 0.0001$ represents the slight difference between the merit of each harmony vector in HM. In fact, at the late stages of musical improvisation, the music is gradually mellowed and the fine-tuning range used is very narrow.

3.2. Design of the Self-Adaptive Mechanism. For every metaheuristic algorithm, accelerating the convergence and preventing the premature convergence appear contradictory. To resolve this contradiction, the evolutionary mechanism of the HS algorithm has to be properly revised. It should be again noted that fine-tuning occurs throughout the whole process of musician improvisation and the fine-tuning range is wider at the early stage and narrower at later stages. Therefore, this phenomenon can be mimicked by the following evolutionary operations:

$$x_i' \longleftarrow \begin{cases} x_i^j + r2 * bw_i; & \\ \quad (j \in \{1, 2, \ldots, \mathrm{HMS}\}) & \text{if } r < \mathrm{HMCR} \\ lb_i + r1 * (ub_i - lb_i) & \text{if } r \ge \mathrm{HMCR} \,\&\, \\ \quad +r2 * bw_i; & f\mathrm{std} > 0.0001 \\ \min\limits_j x_i^j + r1 * \left(\max\limits_j x_i^j - \min\limits_j x_i^j\right) & \text{if } r \ge \mathrm{HMCR} \,\&\, \\ \quad +r2 * bw_i; & f\mathrm{std} \le 0.0001, \end{cases} \tag{12}$$

where random numbers $r, r1 \sim \mathrm{U}[0,1]; r2 \sim \mathrm{U}[-1,1]$. The standard deviation for the objective function values fstd represents the whole merit of the new harmony vectors in the current HM. If the standard deviation is less than a given threshold 0.0001, that is, if $f\mathrm{std} \le 0.0001$, then the difference between the function values is very small. If the iteration number is sufficiently large, then the distribution of the harmony vectors is centralised; thus, a new random value should be generated within the narrow range $[\min_j x_i^j, \max_j x_i^j]$ for rapid convergence.

3.3. Evolutionary Process of the NSHS Algorithm. On the basis of the novel parameter-setting method and self-adaptive evolutionary mechanism, the computational procedure of the proposed variant can be given, as shown in Algorithm 1.

4. Benchmark Functions

This section lists the global minimization benchmark functions used to evaluate the performances of the proposed algorithm. The test suite comprises six well-known unimodal benchmark functions, six well-known multimodal benchmark functions, and six constrained optimization functions.

NSHS Algorithm

Initialise the problem and algorithm parameters:

Define a standard fitness function $f(x), x_i \in [lb_i, ub_i]$ $(i = 1, 2, \ldots, n)$ by (1)

Define the parameters HMCR by (10), Bw by (11), HMS and NI

Generate randomly an initial **HM** and calculate $f$std

Search the best solution:

While $(k \leq \text{NI})$ do

Generate a new harmony vector by the following loop

While $(i \leq n)$ do

First, generate a new value for every variable by the following rules:

If $(r \leq \text{HMCR}, r \sim U[0, 1])$, then choose uniformly a value from the current memory for variable

$x_i' = x_i^j; j \in \{1, 2, \ldots, \text{HMS}\}$

Else, generate randomly

If $f$std > 0.0001, then generate randomly in the range of $[lb_i, ub_i]$

$x_i' = lb_i + r1 * (ub_i - lb_i); r1 \sim U[0, 1]$

Else, generate randomly in the range of $[\min_j x_i^j, \max_j x_i^j]$

$x_i' = \min_j x_i^j + r1 * (\max_j x_i^j - \min_j x_i^j); r1 \sim U[0, 1]$

End if

End if

Second, adjust the new value of every variable by the following rules:

If $f$std > 0.0001, then

$$x_i'' = x' + \frac{ub_i - lb_i}{100} * \left(1 - \frac{k}{\text{NI}}\right) * r2; r2 \sim U[-1, 1]$$

Else, generate randomly in the range of $[\min_j x_i^j, \max_j x_i^j]$

$x_i'' = x_i' + 0.0001 * r2; r2 \sim U[-1, 1]$

End if

End while

Assess the merit of the new harmony vector and update the current harmony memory

If $f(x^{\text{new}}) < f(x^{\text{worst}}) = \max_j f(x^j)$, then

Accept the new harmony vector and replace the worst vector in the **HM** with it.

Recalculate the standard deviation of the fitness function values $f$std.

End if

End while

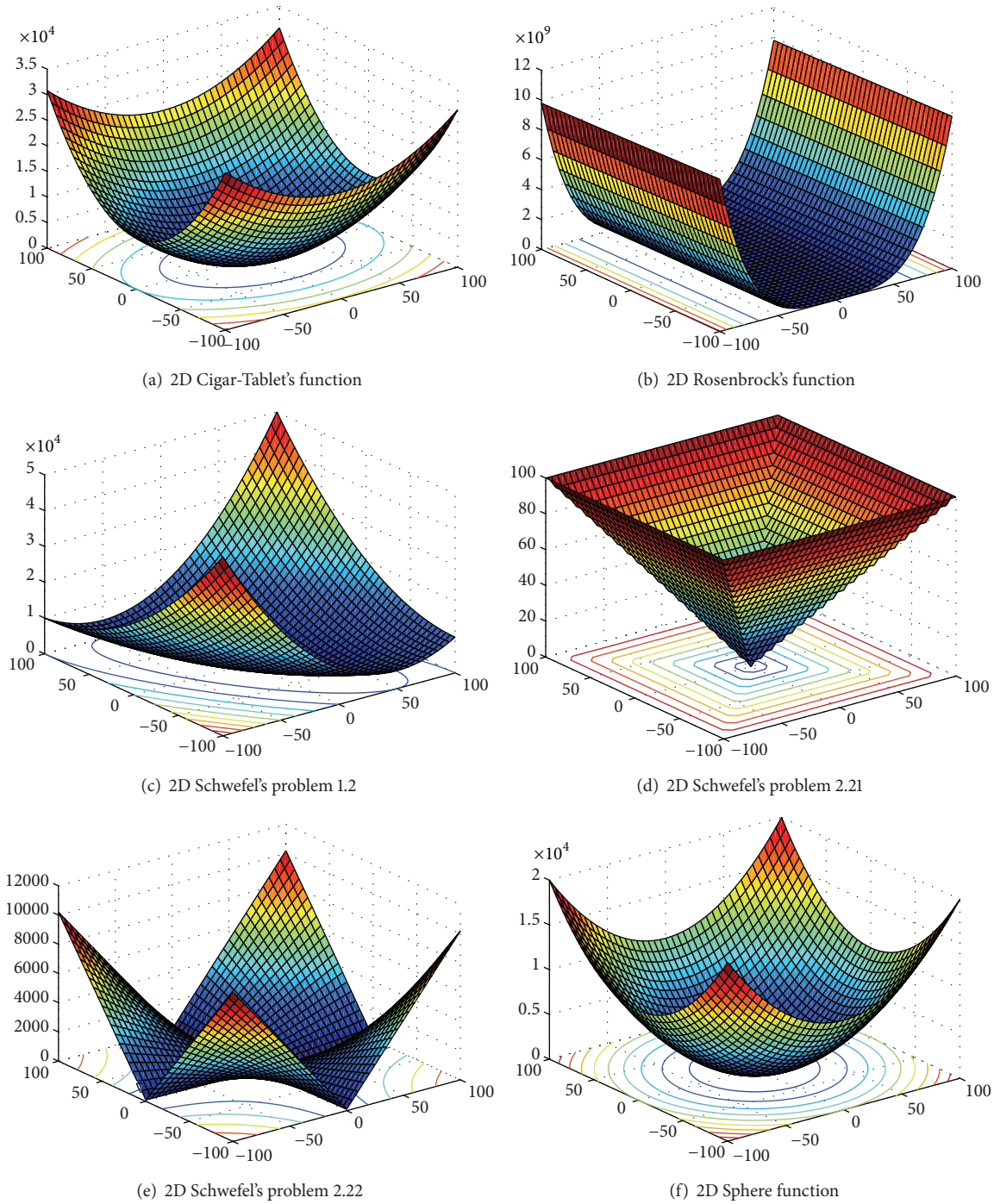
Determine the optimal solution in the **HM** by the minimum fitness.

Algorithm 1: Pseudo code of the NSHS algorithm.

Table 1: Six shifted rotated unimodal benchmark functions.

Name	Expression
Shifted rotated Cigar-Tablet's function	$f_1 = z_1^2 + \ln 8 \cdot z_n^2 + \ln 4 \cdot \sum_{i=2}^{n-1} z_i^2 + \text{bias}$
Shifted rotated Rosenbrock's function	$f_2 = \sum_{i=1}^{n-1} \left(100 \cdot \left(z_{i+1} - z_i^2\right)^2 + \left(z_i - 1\right)^2\right) + \text{bias}$
Shifted rotated Schwefel's problem 1.2 with noise in fitness	$f_3 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j\right)^2 (1 + 0.4 * \lvert N(0, 1)\rvert) + \text{bias}$
Shifted rotated Schwefel's problem 2.21 with noise in fitness	$f_4 = \max_i \{\lvert z_i\rvert\} (1 + 0.4 * \lvert N(0, 1)\rvert) + \text{bias}$
Shifted rotated Schwefel's problem 2.22 with noise in fitness	$f_5 = \left(\sum_{i=1}^{n} \lvert z_i\rvert + \prod_{i=1}^{n} \lvert z_i\rvert\right) (1 + 0.4 * \lvert N(0, 1)\rvert) + \text{bias}$
Shifted rotated sphere function with noise in fitness	$f_6 = \sum_{i=1}^{n} z_i^2 (1 + 0.4 * \lvert N(0, 1)\rvert) + \text{bias}$

*$z = (x - o)\mathbf{M}$. $x = \{x_1, x_2, \ldots, x_n\}$ is a solution in the search range of $[-100, 100]$. $o = \{o_1, o_2, \ldots, o_n\}$ is a random vector in the range of $[-100, 100]$. $\mathbf{M}$ is an identity orthogonal matrix. bias $\in \mathbb{R}$ is also a random number. In this study, these biases are randomly given. They are equal to -140, 390, -450, -310, -180, and -450, respectively. For the second benchmark function, global optimum $x^* = o + 1$ and $f^* = \text{bias}$; for other functions, global optimum $x^* = o$ and $f^* = \text{bias}$.



(a) 2D Cigar-Tablet's function

(b) 2D Rosenbrock's function

(c) 2D Schwefel's problem 1.2

(d) 2D Schwefel's problem 2.21

(e) 2D Schwefel's problem 2.22

(f) 2D Sphere function

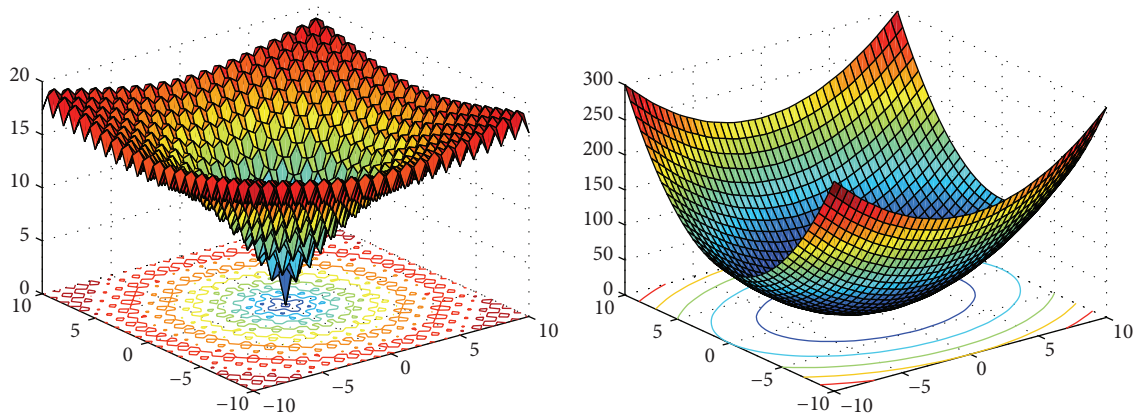
Figure 2: Six well-known unimodal benchmark function landscapes.

To evaluate the robustness of a stochastic search algorithm, the twelve unconstrained benchmark functions are randomly shifted and orthogonally rotated, and four benchmark functions are with normal noise in fitness.

4.1. Unimodal Benchmark Functions. Table 1 summarizes the shifted rotated unimodal benchmark functions. Figure 2 depicts their 3D shapes if the dimension is equal to two.

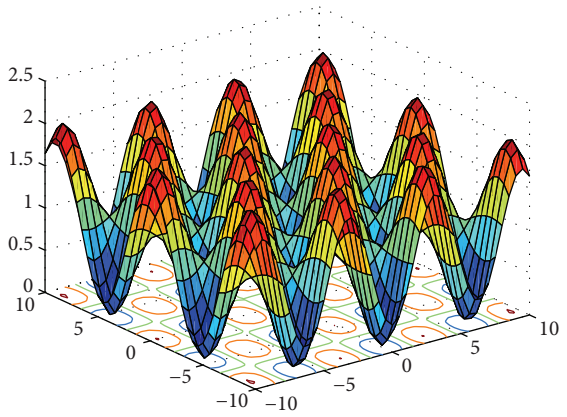
4.2. Multimodal Benchmark Functions. Table 2 summarizes the shifted rotated multimodal benchmark functions. Figure 3 depicts their 3D shapes if the dimension is equal to two.

4.3. Constrained Benchmark Functions. The six constrained optimization functions have also been described in the literature [2]. Thus, we only give their corresponding penalty

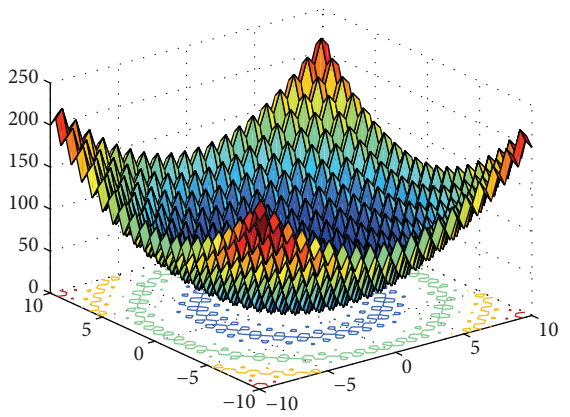


(a) 2D Ackley's function

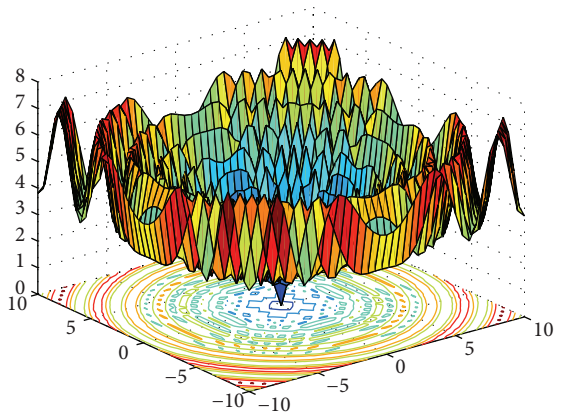
(b) 2D Bohachevsky's function



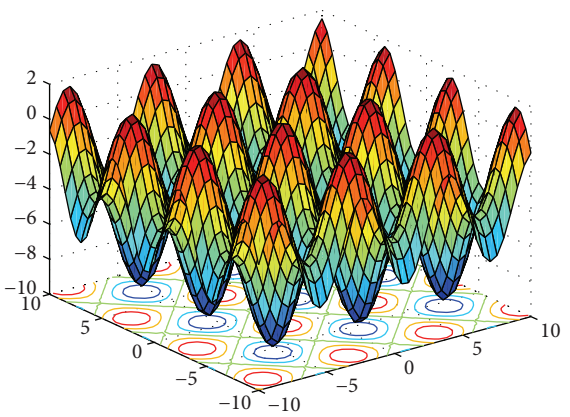
(c) 2D Griewank's function



(d) 2D Rastrigin's function



(e) 2D Schaffer's function



(f) 2D Schwefel's problem 2.13

FIGURE 3: Six well-known multimodal benchmark function landscapes.

functions defined in this study. It should be noted that these penalty functions are composed of many penalty terms in order to precisely control every violation:

$$f_{13} = \min\left\{(x_1 - 2)^2 + (x_2 - 1)^2\right\}, \quad x_1, x_2 \in [-10, 10]. \tag{13}$$

If $\max(\text{abs}(x) - 10) > 0$, then $f_{13} + \max(\text{abs}(x) - 10) \mapsto f_{13}$.

If $x_1 - 2x_2 + 1 = g_1(x) \neq 0$, then $f_{13} + 10\,\text{abs}(g_1) \mapsto f_{13}$.

If $-x_1^2/4 - x_2^2 + 1 = g_2(x) < 0$, then $f_{13} - 10g_2 \mapsto f_{13}$.

For f_{13}, the known best solution $x^* = (0.8229, 0.9114)$ and $f^* = 1.3935$. Consider

$$f_{14} = \min\left\{\left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2\right\}, \quad x_1, x_2 \in [0, 6]. \tag{14}$$

Table 2: Six well-known multimodal benchmark functions.

Name	Expression
Shifted rotated Ackley's function	$f_7 = -20\exp\left(-0.2\cdot\sqrt{\frac{1}{n}\sum_{i=1}^{n}z_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi z_i\right)\right) + 20 + e + \text{bias}$
Shifted rotated Bohachevsky's function	$f_8 = \sum_{i=1}^{n-1}\left(z_i^2 + 2z_{i+1}^2 - 0.3\cos\left(3\pi z_i\right) - 0.4\cos\left(4\pi z_{i+1}\right) + 0.7\right) + \text{bias}$
Shifted rotated Griewank's function	$f_9 = \frac{1}{4000}\sum_{i=1}^{n}z_i^2 - \prod_{i=1}^{n}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + \text{bias}$
Shifted rotated Rastrigin's function	$f_{10} = \sum_{i=1}^{n}\left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right) + \text{bias}$
Shifted rotated Schaffer's function	$f_{11} = \sum_{i=1}^{n-1}\left(z_i^2 + 2z_{i+1}^2\right)^{0.25}\left(1 + \sin^2\left(50\left(z_i^2 + 2z_{i+1}^2\right)^{0.1}\right)\right) + \text{bias}$
Shifted rotated Schwefel's problem 2.13	$f_{12} = \sum_{i=1}^{n}\left(\sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right) - \sum_{j=1}^{n}\left(a_{ij}\sin z_j + b_{ij}\cos z_j\right)\right) + \text{bias}$

*$z = (x - o)\mathbf{M}$. $x = \{x_1, x_2, \ldots, x_n\}$ is a solution in the search range of $[-10, 10]$. $o = \{o_1, o_2, \ldots, o_n\}$ is a random vector in the range of $[-10, 10]$. $\mathbf{M}$ is an identity orthogonal matrix. bias $\in \mathbb{R}$ is also a random number. In this study, these biases are randomly given. They are equal to −140, −310, −180, −330, 390, and −460, respectively. For the twelfth benchmark function, global optimum $x^* = \alpha$ and $f^* =$ bias; for other functions, global optimum $x^* = o$ and $f^* =$ bias.

Table 3: The descriptive statistical result of the relative errors of the NSHS algorithm in the case of HMS = 5, 10, and 50, respectively.

Function	HMS = 5		HMS = 10		HMS = 20		HMS = 50	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
f_1	$7.54E-07$	$4.24E-07$	$4.65E-07$	$2.53E-07$	$1.00E-06$	$5.24E-07$	$5.37E-06$	$1.69E-06$
f_2	$2.96E-01$	$5.36E-01$	$1.56E-01$	$1.45E-01$	$1.22E-01$	$1.36E-01$	$1.09E-01$	$1.32E-01$
f_3	$2.74E-06$	$1.72E-06$	$4.92E-06$	$3.28E-06$	$1.48E-03$	$8.05E-03$	$1.98E-03$	$8.55E-03$
f_4	$3.76E-05$	$1.76E-05$	$4.71E-05$	$1.32E-05$	$7.23E-05$	$1.64E-05$	$1.34E-04$	$2.53E-05$
f_5	$1.32E-04$	$5.49E-05$	$1.41E-04$	$3.03E-05$	$2.09E-04$	$4.82E-05$	$4.17E-04$	$8.23E-05$
f_6	$4.22E-07$	$2.46E-07$	$1.11E-07$	$6.73E-08$	$3.27E-07$	$2.02E-07$	$1.81E-06$	$6.08E-07$
f_7	$3.38E-06$	$1.46E-06$	$6.21E-06$	$2.01E-06$	$1.07E-05$	$3.03E-06$	$2.07E-05$	$4.22E-06$
f_8	$2.25E-09$	$9.12E-10$	$2.95E-09$	$1.26E-09$	$2.90E-08$	$1.80E-08$	$7.47E-07$	$2.82E-07$
f_9	$8.00E-05$	$8.44E-05$	$1.07E-04$	$7.34E-05$	$9.62E-05$	$6.45E-05$	$3.74E-05$	$3.95E-05$
f_{10}	$2.12E-03$	$2.69E-03$	$1.10E-03$	$1.69E-03$	$5.03E-04$	$1.39E-03$	$3.59E-04$	$9.48E-04$
f_{11}	$3.73E-04$	$4.75E-05$	$3.81E-04$	$4.29E-05$	$4.76E-04$	$3.89E-05$	$6.59E-04$	$5.76E-05$
f_{12}	$3.59E-07$	$2.67E-07$	$1.26E-07$	$9.95E-08$	$2.16E-07$	$1.71E-07$	$9.81E-07$	$4.43E-07$
f_{13}	$2.08E-04$	$1.61E-04$	$2.77E-04$	$2.07E-04$	$3.24E-04$	$1.79E-04$	$5.51E-04$	$3.38E-04$
f_{14}	$3.52E-06$	$4.29E-06$	$3.36E-06$	$2.67E-06$	$7.43E-06$	$4.40E-06$	$9.78E-06$	$5.48E-06$
f_{15}	$2.47E-05$	$6.67E-05$	$4.60E-05$	$1.91E-04$	$7.40E-05$	$1.95E-04$	$1.86E-03$	$1.56E-03$
f_{16}	$1.17E-04$	$7.96E-05$	$1.26E-04$	$8.46E-05$	$1.16E-04$	$6.32E-05$	$1.75E-04$	$7.27E-05$
f_{17}	$2.89E-02$	$2.92E-02$	$2.09E-02$	$2.72E-02$	$2.63E-02$	$2.48E-02$	$2.96E-02$	$3.21E-02$
f_{18}	$1.44E-02$	$9.70E-03$	$7.29E-03$	$3.81E-03$	$9.57E-03$	$3.65E-03$	$1.50E-02$	$6.03E-03$
Total	$1.90E-02$	$1.42E-01$	$1.04E-02$	$4.95E-02$	$8.96E-03$	$4.28E-02$	$8.85E-03$	$4.05E-02$

*Each group has 30 samples and the experiment has totally $4 * 18 * 30 = 2160$ samples.

If $\max(x - 6) > 0$, then $f_{14} + \max(x - 6) \mapsto f_{14}$.

If $\max(-x) > 0$, then $f_{14} + \max(-x) \mapsto f_{14}$.

If $4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 = g_1(x) < 0$, then $f_{14} - 100g_1 \mapsto f_{14}$.

If $x_1^2 + (x_2 - 2.5)^2 - 4.84 = g_2(x) < 0$, then $f_{14} - g_2 \mapsto f_{14}$.

For f_{14}, the known best solution $x^* = (2.2468, 2.3819)$ and $f^* = 13.5909$. Consider

$$f_{15} = \min\left\{5.357847x_3^2 + 0.8356891x_1x_2 + 37.293239x_1 - 40792.141\right\}, \quad (15)$$

$$x_1 \in [78, 102], \; x_2 \in [33, 45], \; x_3, x_4, x_5 \in [27, 45].$$

If $x_1 < 78$, then $\text{abs}(f_{15}) - x_1 + 78 \mapsto f_{15}$.
If $x_1 > 102$, then $\text{abs}(f_{15}) + x_1 - 102 \mapsto f_{15}$.
If $x_2 < 33$, then $\text{abs}(f_{15}) - x_2 + 33 \mapsto f_{15}$.
If $x_2 > 45$, then $\text{abs}(f_{15}) + x_2 - 45 \mapsto f_{15}$.
If $27 - \max(x_3, x_4, x_5) > 0$, then $\text{abs}(f_{15}) - \max(x_3, x_4, x_5) + 27 \mapsto f_{15}$.
If $\max(x_3, x_4, x_5) - 45 > 0$, then $\text{abs}(f_{15}) + \max(x_3, x_4, x_5) - 45 \mapsto f_{15}$.
If $85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.002205x_3x_5 = g_1(x) < 0$, then $\text{abs}(f_{15}) - g_1 \mapsto f_{15}$.
If $6.66593 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.002205x_3x_5 = g_2(x) < 0$, then $\text{abs}(f_{15}) - 10g_2 \mapsto f_{15}$.
If $-9.48751 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 = g_3(x) < 0$, then $\text{abs}(f_{15}) - g_3 \mapsto f_{15}$.
If $29.48751 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 = g_4(x) < 0$, then $\text{abs}(f_{15}) - g_4 \mapsto f_{15}$.
If $-10.699039 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 = g_5(x) < 0$ then $\text{abs}(f_{15}) - 10000g_5 \mapsto f_{15}$.
If $15.699039 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 = g_6(x) < 0$, then $\text{abs}(f_{15}) - g_6 \mapsto f_{15}$.
For f_{15}, the known best solution $x^* = (78, 33, 29.995, 45, 36.7764)$ and $f^* = -30665.5$. Consider

$$f_{15} = \min\left\{(x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7\right\}, \quad x_i \in [-10, 10]. \tag{16}$$

If $\max(\text{abs}(x) - 10) > 0$, then $f_{16} + \max(\text{abs}(x) - 10) \mapsto f_{16}$.
If $127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 = g_1(x) < 0$, then $f_{16} - 100g_1 \mapsto f_{16}$.
If $282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 = g_2(x) < 0$, then $f_{16} - g_2 \mapsto f_{16}$.
If $196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 = g_3(x) < 0$, then $f_{16} - g_3 \mapsto f_{16}$.
If $-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 = g_4(x) < 0$, then $f_{16} - 100g_4 \mapsto f_{16}$.
For f_{16}, the known best solution $x^* = (2.3305, 1.9513, -0.4775, 4.3657, -0.6245, 1.0381, 1.5942)$ and $f^* = 680.6301$. Consider

$$f_{17} = \min\left\{x_1 + x_2 + x_3\right\}, \quad x_1 \in [100, 10000], \quad x_2, x_3 \in [1000, 10000], \quad x_4, x_5, x_6, x_7, x_8 \in [10, 1000]. \tag{17}$$

If $x_1 < 100$, then $f_{17} + 100(100 - x_1) \mapsto f_{17}$.
If $x_1 > 10000$, then $f_{17} + x_1 - 10000 \mapsto f_{17}$.
If $1000 - \max(x_2, x_3) > 0$, then $f_{17} + 100(1000 - \max(x_2, x_3)) \mapsto f_{17}$.
If $\max(x_2, x_3) - 10000 > 0$, then $f_{17} + \max(x_2, x_3) - 10000 \mapsto f_{17}$.
If $10 - \max(x_4, x_5, x_6, x_7, x_8) > 0$, then $f_{17} + 10 - \max(x_4, x_5, x_6, x_7, x_8) \mapsto f_{17}$.
If $\max(x_4, x_5, x_6, x_7, x_8) - 1000 > 0$, then $f_{17} + \max(x_4, x_5, x_6, x_7, x_8) - 1000 \mapsto f_{17}$.
If $1 - 0.0025(x_4 + x_6) = g_1(x) < 0$, then $f_{17} - 3500g_1 \mapsto f_{17}$.

Table 4: The report of SPSS on the univariate (relative error) analysis of variance with two fixed factors (problem/function and HMS).

(a) Tests of between-subjects effects (dependent variable: relative error)

Source	Type III sum of squares	df	Mean square	F	Sig.
Corrected Model	3.958[a]	71	.056	11.537	.000
	.301	1	.301	62.211	.000
HMS	.038	3	.013	2.632	**.048**
Problem	3.293	17	.194	40.092	**.000**
HMS * Problem	.627	51	.012	2.542	**.000**
Error	10.089	2088	.005		
Total	14.347	2160			
Corrected Total	14.046	2159			

[a] R squared = .282 (adjusted R squared = .257).

(b) Hochberg's multiple comparisons (dependent variable: relative error)

(I) HMS	(J) HMS	Mean difference (I – J)	Std.	Sig.
5	10	.0086	.0042	**.225**
	20	.0100	.0042	**.101**
	50	.0102	.0042	**.095**
10	5	–.0086	.0042	**.225**
	20	.0014	.0042	**1.000**
	50	.0015	.0042	**1.000**
20	5	–.0100	.0042	**.101**
	10	–.0014	.0042	**1.000**
	50	.0001	.0042	**1.000**
50	5	–.0102	.0042	**.095**
	10	–.0015	.0042	**1.000**
	20	–.0001	.0042	**1.000**

Based on observed means.

If $1 - 0.0025(x_5 + x_7 - x_4) = g_2(x) < 0$, then $f_{17} - 6000g_2 \mapsto f_{17}$.
If $1 - 0.01(x_8 - x_5) = g_3(x) < 0$, then $f_{17} - 4000g_3 \mapsto f_{17}$.
If $x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 = g_4(x) < 0$, then $f_{17} - g_4 \mapsto f_{17}$.
If $x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 = g_5(x) < 0$, then $f_{17} - 0.1g_5 \mapsto f_{17}$.
If $x_3x_8 - x_3x_5 + 2500x_5 - 1250000 = g_6(x) < 0$, then $f_{17} - g_6 \mapsto f_{17}$.
For f_{17}, the known best solution $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$ and $f^* = 7049.3309$. Consider

$$f_{18} = \min\left\{x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2\right.$$

Table 5: The descriptive statistical result of the relative errors of the NSHS algorithm using three types of the initial search ranges.

Function	The whole domain		The left part		The right part	
	Mean	Std.	Mean	Std.	Mean	Std.
f_1	$4.10E-07$	$2.00E-07$	$3.97E-07$	$2.35E-07$	$4.67E-07$	$4.34E-07$
f_2	$7.96E-01$	$2.34E+00$	$6.64E-01$	$2.30E+00$	$1.20E-01$	$1.90E-01$
f_3	$4.88E-06$	$2.72E-06$	$4.87E-06$	$3.20E-06$	$4.73E-06$	$2.64E-06$
f_4	$4.62E-05$	$1.16E-05$	$4.75E-05$	$1.20E-05$	$4.77E-05$	$1.40E-05$
f_5	$1.46E-04$	$3.73E-05$	$1.39E-04$	$4.14E-05$	$1.29E-04$	$3.29E-05$
f_6	$1.12E-07$	$7.03E-08$	$1.48E-07$	$1.68E-07$	$1.21E-07$	$6.96E-08$
f_7	$6.70E-06$	$1.45E-06$	$7.06E-06$	$1.59E-06$	$6.47E-06$	$1.82E-06$
f_8	$4.08E-09$	$4.51E-09$	$4.22E-09$	$6.34E-09$	$9.48E-09$	$1.96E-08$
f_9	$1.18E-04$	$8.47E-05$	$1.23E-04$	$6.71E-05$	$1.16E-04$	$7.40E-05$
f_{10}	$1.49E-03$	$2.44E-03$	$1.44E-03$	$2.08E-03$	$5.38E-04$	$1.14E-03$
f_{11}	$3.87E-04$	$5.90E-05$	$3.98E-04$	$5.71E-05$	$3.90E-04$	$4.95E-05$
f_{12}	$1.84E-07$	$1.64E-07$	$1.49E-07$	$1.39E-07$	$1.90E-07$	$2.35E-07$
f_{13}	$2.86E-04$	$2.09E-04$	$2.99E-04$	$1.81E-04$	$2.82E-04$	$1.88E-04$
f_{14}	$6.86E-06$	$5.36E-06$	$5.06E-06$	$3.60E-06$	$5.53E-06$	$5.02E-06$
f_{15}	$1.43E-05$	$9.29E-06$	$1.14E-05$	$3.25E-06$	$1.24E-05$	$3.99E-06$
f_{16}	$1.01E-04$	$5.61E-05$	$9.12E-05$	$3.53E-05$	$1.24E-04$	$8.51E-05$
f_{17}	$2.86E-02$	$2.43E-02$	$1.73E-02$	$2.47E-02$	$3.06E-02$	$1.37E-02$
f_{18}	$8.42E-03$	$4.12E-03$	$9.03E-03$	$3.92E-03$	$8.45E-03$	$3.72E-03$
Total	$4.64E-02$	$5.72E-01$	$3.85E-02$	$5.54E-01$	$8.91E-03$	$5.23E-02$

*Each group has 30 samples and the experiment has totally $3 * 18 * 30 = 1620$ samples.

$$+(x_{10}-7)^2+45\Big\},$$
$$x_i \in [-10,10] \quad (i=1,2,\ldots,10). \tag{18}$$

If $\max(\text{abs}(x)-10)>0$, then $f_{18}+\max(\text{abs}(x)-10) \mapsto f_{18}$.

If $105-4x_1-5x_2+3x_7-9x_8 = g_1(x)<0$, then $f_{18}-10g_1 \mapsto f_{18}$.

If $-10x_1+8x_2+17x_7-2x_8 = g_2(x)<0$, then $f_{18}-10g_2 \mapsto f_{18}$.

If $8x_1-2x_2-5x_9+2x_{10}+12 = g_3(x)<0$, then $f_{18}-10g_3 \mapsto f_{18}$.

If $-3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4+120 = g_4(x)<0$, then $f_{18}-10g_4 \mapsto f_{18}$.

If $-5x_1^2-8x_2-(x_3-6)^2+2x_4+40 = g_5(x)<0$, then $f_{18}-10g_5 \mapsto f_{18}$.

If $-x_1^2-2(x_2-2)^2+2x_1x_2-14x_5+6x_6 = g_6(x)<0$, then $f_{18}-10g_6 \mapsto f_{18}$.

If $-0.5(x_1-8)^2+2(x_2-4)^2-3x_5^2+x_6+30 = g_7(x)<0$, then $f_{18}-10g_7 \mapsto f_{18}$.

If $3x_1-6x_2-12(x_9-8)^2+7x_{10} = g_8(x)<0$, then $f_{18}-g_8 \mapsto f_{18}$.

For f_{18}, the known best solution $x^* = (2.1720, 2.3637, 8.7739, 5.0960, 0.9907, 1.4306, 1.3216, 9.8287, 8.2801, 8.3760)$ and $f^* = 24.3062$.

5. Robustness of the Proposed NSHS Algorithm

The performance of a stochastic search algorithm often is affected by the initial populations. For the HS algorithm, the initial population distribution is determined by the value of HMS and the initial search ranges. Thus, this section investigates the robustness of the proposed variant from the two aspects.

5.1. Effect of HMS. Yang [21] declared that the value of HMS is insensitive to the performance of the original HS algorithm. To reveal the fact that the value of HMS does not also have a significant impact on the performance of the NSHS, a full-factorial experiment was conducted. In the experiment, the HMS is valued at four levels (5, 10, 20, and 50) as a fixed factor; above twelve unconstrained benchmark functions with 10-dimension and six constrained functions were employed; the simulation was repeated 30 times, and each simulation underwent 50,000 evaluations for the objective function. The relative error between the running result and the global or known optimum is tested as a dependent variable. The descriptive statistical result of these errors is shown in Table 3.

Clearly, there exists the tiny difference. However, we can claim that the difference is due to the function tested rather than the HMS by the test result of univariate analysis of variance, seeing the report of SPSS shown in Table 4.

Table 6: The report of SPSS on the univariate (relative error) analysis of variance with two fixed factors (problem/function and initial range).

(a) Tests of between-subjects effects (dependent variable: relative error)

Source	Sum of squares	df	Mean square	F	Sig.
Corrected Model	31.122[a]	53	.587	2.939	.000
	1.584	1	1.584	7.927	.005
Initial range	.422	2	.211	1.055	**.348**
Problem	23.416	17	1.377	6.895	**.000**
Initial range ∗ problem	7.285	34	.214	1.073	**.357**
Error	312.834	1566	.200		
Total	345.540	1620			
Corrected	343.956	1619			

[a] R squared = .090 (adjusted R squared = .060).

(b) Hochberg's multiple comparisons (dependent variable: relative error)

(I) Initial range	(J) Initial range	Mean difference (I − J)	Std.	Sig.
Left	Right	.0296	.0272	**.622**
	Whole	−.0079	.0272	**.988**
Right	Left	−.0296	.0272	**.622**
	Whole	−.0375	.0272	**.424**
Whole	Left	.0079	.0272	**.988**
	Right	.0375	.0272	**.424**

Based on observed means.

Therefore, the performance of the proposed NSHS algorithm is not affected by the value of HMS. To save storage space, it is encouraged to use a small HMS, as Pan et al. [17] explained that "since HM is emulating musician's short-term memory and the short-term memory of the human is known to be small, it is logical."

5.2. Effect of Initial Search Ranges. To identify whether the performance of the proposed variant depends on the initial HM, a full-factorial experiment was conducted using the parameters HMS = 10 and NI = 50,000. In the experiment, the initial HM was randomly generated at three types of the initial search ranges, that is, the whole domain, the left part, and the right part. The experiment was repeated 30 times and 1620 observations were collected. The descriptive statistical result of the relative errors of the proposed variant is shown in Table 5.

Obviously, there still exists the small difference. However, the test report of SPSS shown in Table 6 still indicates that the difference is due to the function tested rather than the initial search ranges.

By the results of analysis of variance as shown in Tables 4 and 6, the performance of the proposed NSHS algorithm does not depend on the initial HM. By the descriptive statistical results as shown in Tables 3 and 5, moreover, no matter how the initial HM is, the NSHS algorithm can stability search for the global optima of these benchmark functions with a tiny error, except for function f_2. Therefore, the novel variant is considerably robust.

6. Superiority of the Proposed NSHS Algorithm

To confirm the significant superiority of the proposed variant, a comparison test was conducted. The test compared the performance of the NSHS with the original HS algorithm [1] and the other variants mentioned in Section 2, including Mahdavi's IHS [14], Wang's self-adaptive variant represented by HSw for convenience [16], Pan's DLHS [18], and SGHS [17]. For fair comparison, the parameter settings are those proposed by their developers, as shown in Table 7.

For sufficiently persuasive comparative results, every algorithm was run independently 30 times to solve the aforementioned 18 benchmark functions on the same computer (CPU: AMD 4 × 3.5 Ghz; RAM: 4 GB), and, at the beginning of solving a problem, each algorithm used the same initial HM. The conclusion is made based on an analysis of variance rather than a pairwise t-test because Garc's study yet indicated that the latter method is suitable for a single-problem analysis and a multiple-problem analysis should employ a nonparametric test [22]. Next, the experimental results are elaborated from two aspects, respectively.

6.1. Unconstrained Benchmark Function. The performances of these algorithms regarding twelve unconstrained benchmark functions were investigated at different scales, that is, 10, 100, and 1000. Table 8, Table 10 and Table 12 respectively summarize the descriptive statistical results at different scales. For 10-dimension benchmark functions, the superiority of the NSHS is not significant except for functions f_3 and f_4, as shown in Table 8. The test result shown in Table 9 indicates that the performance of the NSHS is significant better than the IHS at the 95% confidence level and the average error of the NSHS is less than other algorithms except for the DLHS. For 100-dimension benchmark functions, the average error of the NSHS is much less than other algorithms except for the SGHS, as shown in Table 11. Based on the data in Tables 12 and 13, we can see that the superiority of the proposed variant is extremely significant for 1000-dimension benchmark functions.

6.2. Constrained Benchmark Function. The performances of these algorithms with respect to the aforementioned optimization problems with constraints were investigated. The descriptive statistical result and the report on multiple comparisons are, respectively, shown in Tables 14 and 15. Clearly, the performance of the proposed variant is vastly superior to the HS, the IHS, the HSw, and the SGHS. Although the performance of the NSHS is slightly better

Table 7: Algorithm parameters.

Algorithm	HMS	HMCR	PAR	Bw	NI	Others
HS	9	0.9	0.3	0.01	50,000	None
IHS	9	0.9	Min: 0.01; max: 0.99	Min: 0.0001; max: $1/(20(ub - lb))$	50,000	None
HSw	9	0.9	Min: 0.01; max: 0.99	Needless	50,000	None
DLHS	9	Initial: 0.91	Initail: 0.5	Min: 0.0001; max: $(ub - lb)/200$	50,000	$m = 3$; $R = 50$: len = 200
SGHS	9	Initial: 0.98	Initial: 0.9	Min: 0.0005; max: $(ub - lb)/10$	50,000	LP = 100
NSHS*	9	$1 - 1/(n + 1)$	Needless	No initial value, using (11)	50,000	None

*NSHS has only two key parameters HMCR and Bw, but they do not require the user's inputs.

Table 8: The descriptive statistical result of the relative errors of the six algorithms to solve twelve unconstrained benchmark functions with 10 dimensions.

Function	HS	IHS	HSw	DLHS	SGHS	NSHS
	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)
f_1	$1.02E-09$ ($7.27E-10$)	$4.77E-04$ ($3.90E-04$)	$1.10E-04$ ($1.39E-04$)	$1.96E-12$ ($1.20E-12$)	**$6.40E-13$** (**$4.98E-13$**)	$4.49E-07$ ($3.42E-07$)
f_2	$2.17E-01$ ($7.90E-01$)	$4.47E-01$ ($5.54E-01$)	$1.77E-01$ ($1.77E-01$)	$2.36E-01$ ($3.18E-01$)	$4.50E-01$ ($1.50E+00$)	$4.94E-01$ ($9.18E-01$)
f_3	$7.08E-01$ ($4.88E-01$)	$8.61E-01$ ($5.77E-01$)	$8.14E-01$ ($5.33E-01$)	$1.64E-02$ ($8.96E-02$)	$2.59E-01$ ($3.38E-01$)	**$5.11E-06$** (**$2.91E-06$**)
f_4	$1.49E-02$ ($7.17E-03$)	$1.37E-02$ ($9.75E-03$)	$6.97E-03$ ($2.31E-03$)	$1.12E-03$ ($3.48E-03$)	$1.94E-02$ ($2.14E-02$)	**$4.22E-05$** (**$1.22E-05$**)
f_5	$6.15E-03$ ($2.77E-03$)	$1.13E-02$ ($3.65E-03$)	$2.50E-03$ ($1.17E-03$)	**$2.05E-07$** (**$6.51E-08$**)	$1.87E-03$ ($1.14E-03$)	$1.19E-04$ ($4.72E-05$)
f_6	$2.13E-04$ ($2.84E-04$)	$8.78E-04$ ($6.27E-04$)	$1.61E-04$ ($1.69E-04$)	**$4.25E-13$** (**$2.80E-13$**)	$1.31E-05$ ($1.48E-05$)	$1.39E-07$ ($7.43E-08$)
f_7	$1.74E-06$ ($6.23E-07$)	$7.56E-07$ ($9.01E-08$)	$8.43E-05$ ($3.82E-05$)	**$8.71E-08$** (**$4.24E-08$**)	$5.24E-08$ ($1.25E-08$)	$6.00E-06$ ($1.43E-06$)
f_8	$9.05E-09$ ($5.01E-09$)	$9.92E-10$ ($3.60E-10$)	$1.89E-05$ ($2.00E-05$)	$1.78E-11$ ($1.47E-11$)	**$5.61E-12$** (**$4.12E-12$**)	$3.10E-09$ ($2.39E-09$)
f_9	$7.64E-05$ ($6.26E-05$)	$7.37E-05$ ($6.07E-05$)	$5.74E-05$ ($4.79E-05$)	**$5.06E-05$** (**$3.89E-05$**)	$2.01E-04$ ($1.08E-04$)	$1.13E-04$ ($9.19E-05$)
f_{10}	$5.38E-08$ ($4.63E-08$)	$4.37E-09$ ($1.28E-09$)	$1.15E-04$ ($9.04E-05$)	$6.86E-11$ ($5.53E-11$)	**$2.63E-11$** (**$1.73E-11$**)	$5.47E-04$ ($1.37E-03$)
f_{11}	$2.92E-03$ ($3.01E-03$)	$2.61E-03$ ($3.48E-03$)	$1.13E-03$ ($4.56E-04$)	$1.97E-04$ ($7.40E-04$)	**$3.94E-05$** (**$4.99E-06$**)	$3.99E-04$ ($5.66E-05$)
f_{12}	$1.04E-10$ ($6.02E-11$)	$2.21E-12$ ($2.58E-12$)	$4.51E-08$ ($3.87E-08$)	$4.54E-13$ ($1.06E-12$)	**$8.38E-14$** (**$6.21E-14$**)	$1.58E-07$ ($1.37E-07$)
Total	$7.91E-02$ ($3.30E-01$)	$1.11E-01$ ($3.43E-01$)	$8.36E-02$ ($2.77E-01$)	**$2.12E-02$** (**$1.14E-01$**)	$6.08E-02$ ($4.57E-01$)	**$4.12E-02$** (**$2.94E-01$**)

*Each group has 30 samples and the experiment has totally $12 * 6 * 30 = 2160$ samples. The bold represents the significant best.

Table 9: Dunnett's one-tailed (>) multiple comparisons between the NSHS and other five algorithms based on the samples of solving twelve unconstrained benchmark functions with 10 dimensions.

(I) Algorithm	(J) Algorithm	Mean difference (I – J)	Std.	Sig.
HS	NSHS	.03782732242345	.0199	.103
IHS	NSHS	.07017638607941*	.0199	**.001**
HSw	NSHS	.04233406887345	.0199	.064
DLHS	NSHS	−.02008968262494	.0199	.986
SGHS	NSHS	.01957666996786	.0199	.421

*The mean difference is significant at the 0.05 level. The test is based on observed means of the dependent variable (relative error).

Table 10: The descriptive statistical result of the relative errors of the six algorithms to solve twelve unconstrained benchmark functions with 100 dimensions.

Function	HS	IHS	HSw	DLHS	SGHS	NSHS
	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)
f_1	$2.30E+02$ ($2.34E+01$)	$2.38E+02$ ($3.29E+01$)	$2.06E+02$ ($2.57E+01$)	$1.30E+02$ ($3.60E+01$)	$5.23E-02$ ($8.03E-03$)	**$1.56E-03$** (**$3.58E-04$**)
f_2	$8.24E+06$ ($1.82E+06$)	$7.97E+06$ ($1.77E+06$)	$5.60E+06$ ($1.14E+06$)	$2.21E+06$ ($1.08E+06$)	$4.91E+00$ ($2.32E+00$)	**$2.44E+00$** (**$7.03E+00$**)
f_3	$8.94E+02$ ($1.30E+02$)	$9.46E+02$ ($1.04E+02$)	$8.68E+02$ ($1.29E+02$)	$5.79E+02$ ($7.58E+01$)	$7.99E+02$ ($1.66E+02$)	$8.04E+02$ ($1.83E+02$)
f_4	$3.27E-01$ ($2.30E-02$)	$3.22E-01$ ($2.08E-02$)	$2.43E-01$ ($1.91E-02$)	$2.69E-01$ ($2.65E-02$)	$4.22E-01$ ($4.47E-02$)	$4.24E-01$ ($3.39E-02$)
f_5	$2.09E+10$ ($7.98E+10$)	$1.50E+12$ ($8.23E+12$)	$6.56E+00$ ($7.28E-01$)	$3.84E+00$ ($6.54E-01$)	**$1.38E+00$** (**$2.12E-01$**)	$4.78E+00$ ($1.05E+00$)
f_6	$9.32E+01$ ($1.32E+01$)	$8.88E+01$ ($9.45E+00$)	$8.59E+01$ ($1.28E+01$)	$6.79E+01$ ($1.38E+01$)	$2.40E+01$ ($4.48E+00$)	**$1.19E+00$** (**$2.37E+00$**)
f_7	$4.93E-02$ ($1.90E-03$)	$4.88E-02$ ($2.15E-03$)	$4.79E-02$ ($1.87E-03$)	$4.21E-02$ ($4.51E-03$)	**$3.60E-04$** (**$2.36E-04$**)	$5.01E-04$ ($1.20E-03$)
f_8	$2.63E+00$ ($2.85E-01$)	$2.66E+00$ ($2.34E-01$)	$2.38E+00$ ($2.72E-01$)	$1.50E+00$ ($4.14E-01$)	$1.23E-02$ ($3.98E-03$)	**$1.10E-02$** (**$5.63E-03$**)
f_9	$5.86E-03$ ($8.71E-05$)	$5.87E-03$ ($5.18E-05$)	$5.74E-03$ ($1.10E-04$)	$4.46E-03$ ($9.65E-04$)	$3.07E-05$ ($5.04E-05$)	**$1.77E-05$** (**$2.40E-05$**)
f_{10}	$2.40E+00$ ($1.83E-01$)	$2.41E+00$ ($1.66E-01$)	$2.69E+00$ ($1.59E-01$)	$1.52E+00$ ($4.14E-01$)	$1.54E-01$ ($2.06E-02$)	**$3.35E-01$** (**$3.89E-02$**)
f_{11}	$4.62E-01$ ($3.03E-02$)	$4.26E-01$ ($2.40E-02$)	$4.59E-01$ ($1.79E-02$)	$3.89E-01$ ($3.40E-02$)	$1.24E-01$ ($1.64E-02$)	$5.57E-01$ ($1.08E-01$)
f_{12}	$1.32E-02$ ($1.59E-03$)	$1.26E-02$ ($1.73E-03$)	$3.86E-02$ ($3.18E-03$)	$7.20E-03$ ($2.15E-03$)	**$4.73E-05$** (**$2.23E-05$**)	$2.93E-04$ ($6.85E-05$)
Total	$1.74E+09$ ($2.34E+10$)	$1.25E+11$ ($2.37E+12$)	$4.67E+05$ ($1.58E+06$)	$1.85E+05$ ($6.86E+05$)	$6.92E+01$ ($2.25E+02$)	**$6.78E+01$** (**$2.28E+02$**)

*Each group has 30 samples and the experiment has totally $12 * 6 * 30 = 2160$ samples. The bold represents the significant best.

Table 11: Dunnett's one-tailed (>) multiple comparisons between the NSHS and other five algorithms based on the samples of solving twelve unconstrained benchmark functions with 100 dimensions.

(I) Algorithm	(J) Algorithm	Mean difference (I − J)	Std.	Sig.
HS	NSHS	**1743032079.82376**	72262361658.159	.826
IHS	NSHS	**125173553098.6880**	72262361658.159	.143
HSw	NSHS	**466806.998460054**	72262361658.159	.833
DLHS	NSHS	**184496.101783522**	72262361658.159	.833
SGHS	NSHS	1.39707526033672	72262361658.159	.833

*The mean difference is significant at the 0.05 level. The test is based on observed means of the dependent variable (relative error).

than the DLHS, the evolutionary mechanism of the former is simpler and the NSHS does not require too much key-parameters setting. Moreover, the performance of the NSHS is significantly better than the DLHS for functions f_{15} and f_{18}, seeing Table 14.

7. Conclusion

This paper presented a novel self-adaptive harmony search algorithm (NSHS) for optimization problems with continuous variables with the aim of improving the performance of the harmony search. To investigate the performance of the novel variant, many comparison experiments were conducted based on a test suite comprising six unimodal benchmark functions, six multimodal benchmark functions, and six constrained benchmark functions. Using nonparametric statistical tests, the NSHS's robustness and superiority were detected. Experimental results on the effect of HMS and initial search ranges indicated that the performance of the NSHS did not depend on the initial HM. The results, when compared with related previous work, manifested that the NSHS performed dramatically better than other

Table 12: The descriptive statistical result of the relative errors of the six algorithms to solve twelve unconstrained benchmark functions with 1000 dimensions.

Function	HS	IHS	HSw	DLHS	SGHS	NSHS
	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)
f_1	$2.65E+04$ ($5.35E+02$)	$2.63E+04$ ($5.15E+02$)	$2.85E+04$ ($6.46E+02$)	$2.63E+04$ ($1.76E+03$)	$1.77E+03$ ($1.33E+02$)	**$7.93E+00$** (**$5.36E-01$**)
f_2	$6.34E+09$ ($3.62E+08$)	$6.41E+09$ ($2.85E+08$)	$6.43E+09$ ($3.01E+08$)	$6.31E+09$ ($6.22E+08$)	$5.04E+07$ ($3.26E+06$)	**$6.91E+03$** (**$1.40E+03$**)
f_3	$1.16E+05$ ($1.38E+04$)	$1.17E+05$ ($1.34E+04$)	$1.10E+05$ ($9.12E+03$)	$8.30E+04$ ($6.59E+03$)	$7.01E+04$ ($1.02E+04$)	**$6.86E+04$** (**$7.45E+03$**)
f_4	$5.58E-01$ ($7.02E-03$)	$5.56E-01$ ($9.08E-03$)	$5.18E-01$ ($6.71E-03$)	$5.43E-01$ ($7.79E-03$)	$5.86E-01$ ($1.19E-02$)	$5.72E-01$ ($1.16E-02$)
f_5	$2.61E+00$ ($3.17E-02$)	$2.61E+00$ ($5.05E-02$)	$2.69E+00$ ($5.58E-02$)	$2.60E+00$ ($5.30E-02$)	$2.69E+00$ ($4.96E-02$)	$3.15E+00$ ($6.74E-02$)
f_6	$7.83E+03$ ($2.54E+02$)	$7.97E+03$ ($2.55E+02$)	$8.08E+03$ ($2.23E+02$)	$7.78E+03$ ($3.95E+02$)	$7.72E+03$ ($2.70E+02$)	$1.02E+04$ ($4.90E+02$)
f_7	$1.04E-01$ ($5.97E-04$)	$1.04E-01$ ($5.02E-04$)	$1.06E-01$ ($9.92E-04$)	$1.04E-01$ ($1.23E-03$)	$5.69E-02$ ($9.70E-04$)	**$4.47E-02$** (**$1.37E-03$**)
f_8	$2.61E+02$ ($5.43E+00$)	$2.60E+02$ ($6.21E+00$)	$2.79E+02$ ($7.31E+00$)	$2.60E+02$ ($1.43E+01$)	$1.95E+01$ ($1.32E+00$)	**$2.07E+00$** (**$5.15E-02$**)
f_9	$4.28E-02$ ($1.01E-03$)	$4.26E-02$ ($1.18E-03$)	$4.51E-02$ ($9.60E-04$)	$4.26E-02$ ($2.36E-03$)	$7.97E-03$ ($1.54E-04$)	**$5.12E-04$** (**$3.80E-05$**)
f_{10}	$1.10E+02$ ($1.95E+00$)	$1.10E+02$ ($2.46E+00$)	$1.17E+02$ ($3.02E+00$)	$1.12E+02$ ($5.36E+00$)	$3.26E+01$ ($1.09E+00$)	**$2.54E+01$** (**$8.27E-01$**)
f_{11}	$1.02E+01$ ($1.06E-01$)	$9.92E+00$ ($9.23E-02$)	$1.04E+01$ ($9.89E-02$)	$1.00E+01$ ($1.54E-01$)	**$6.68E+00$** (**$2.17E-01$**)	$1.19E+01$ ($1.03E-01$)
f_{12}	$1.70E+00$ ($3.57E-02$)	$1.68E+00$ ($3.93E-02$)	$2.27E+00$ ($5.14E-02$)	$1.73E+00$ ($1.12E-01$)	$3.21E-01$ ($2.46E-02$)	**$1.59E-01$** (**$7.18E-03$**)
Total	$5.29E+08$ ($1.76E+09$)	$5.34E+08$ ($1.77E+09$)	$5.36E+08$ ($1.78E+09$)	$5.26E+08$ ($1.76E+09$)	$4.21E+06$ ($1.40E+07$)	**$7.14E+03$** (**$1.90E+04$**)

*Each group has 30 samples and the experiment has totally $12 * 6 * 30 = 2160$ samples. For a solvability, the search ranges of function f5 are renewed in $[-1, 1]$ because 100^1000 reaches an infinity even 10^1000. The bold represents the significant best.

Table 13: Dunnett's one-tailed (>) multiple comparisons between the NSHS and other five algorithms based on the samples of solving twelve unconstrained benchmark functions with 1000 dimensions.

(I) Algorithm	(J) Algorithm	Mean difference (I − J)	Std.	Sig.
HS	NSHS	**528495067.3354575***	7295157.814	**.000**
IHS	NSHS	**533757007.9854518***	7295157.814	**.000**
HSw	NSHS	**536111617.63627493***	7295157.814	**.000**
DLHS	NSHS	**525985111.0408457***	7295157.814	**.000**
SGHS	NSHS	**4198894.56625653**	7295157.814	.611

*The mean difference is significant at the 0.05 level. The test is based on observed means of the dependent variable (relative error).

six algorithms for large-scale problems and constrained problems.

The originality of this work is mainly reflected in two aspects.

(i) Proposed a simpler approach to the setting of key parameters based on musical practices. The proposed variant has only two key parameters HMCR and Bw, but they do not require the user's inputs. Moreover, the parameters settings are in accordance with the features of the optimization problem to be solved, such as the scale and the boundaries. Moreover, this approach discards the PAR parameter and further simplifies the parameter setting. Musical improvisation is usually accompanied by fine-tuning; thus, it is senseless to consider this parameter and fine-tuning bandwidth at the same time.

(ii) Developed a novel self-adaptive evolutionary scheme based on the standard deviation of the fitness function values in the memory to accelerate algorithm convergence and assure the reaching of a global optimum. This scheme automatically narrows the search space based on a suitable threshold and, meanwhile, solves

Table 14: The descriptive statistical result of the relative errors of the six algorithms to solve six constrained benchmark functions.

Function	HS	IHS	HSw	DLHS	SGHS	NSHS
	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)	Mean (std.)
f_{13}	$3.22E-01$ ($5.19E-01$)	$7.81E-01$ ($1.04E+00$)	$7.68E-01$ ($8.69E-01$)	$-2.22E-05$ ($2.46E-06$)	$7.09E-02$ ($1.10E-01$)	**$2.44E-04$** (**$1.87E-04$**)
f_{14}	$1.91E-05$ ($2.45E-05$)	$1.71E-04$ ($1.72E-04$)	$4.59E-03$ ($2.51E-03$)	**$1.84E-06$** (**$2.72E-06$**)	$2.59E-03$ ($2.38E-03$)	$4.49E-06$ ($4.70E-06$)
f_{15}	$8.72E-03$ ($4.86E-03$)	$8.72E-03$ ($5.13E-03$)	$1.06E-02$ ($4.14E-03$)	$3.35E-03$ ($2.98E-03$)	$1.16E-02$ ($5.63E-03$)	**$1.16E-05$** (**$3.20E-06$**)
f_{16}	$8.44E-03$ ($9.40E-03$)	$2.84E-02$ ($2.04E-02$)	$1.80E-02$ ($1.23E-02$)	$1.34E-04$ ($1.29E-04$)	$6.31E-03$ ($6.17E-03$)	**$1.00E-04$** (**$6.43E-05$**)
f_{17}	$6.81E-02$ ($7.64E-02$)	$7.68E-02$ ($8.20E-02$)	$8.24E-02$ ($4.25E-02$)	$3.93E-02$ ($3.71E-02$)	$1.02E-01$ ($7.23E-02$)	**$2.48E-02$** (**$2.14E-02$**)
f_{18}	$2.81E-01$ ($1.81E-01$)	$5.93E-01$ ($4.37E-01$)	$3.76E-01$ ($2.15E-01$)	$7.49E-02$ ($5.50E-02$)	$2.55E-01$ ($1.82E-01$)	**$7.98E-03$** (**$3.76E-03$**)
Total	$1.16E-01$ ($2.63E-01$)	$2.53E-01$ ($5.62E-01$)	$2.12E-01$ ($4.62E-01$)	$1.95E-02$ ($3.91E-02$)	$7.47E-02$ ($1.29E-01$)	**$5.20E-03$** (**$1.22E-02$**)

*The experiment collected $6 * 6 * 30 = 1080$ observations, 23 out of whom are invalid. For function f_{15}, the DLHS got 2 infeasible solutions. For function f_{17}, the HS got 4 infeasible solutions and the IHS 5, the HSw 3, the DLHS 1, the SGHS 5, and the NSHS 3, respectively. The bold represents the significant best.

Table 15: Dunnett's one-tailed (>) multiple comparisons between the NSHS and other five algorithms based on the samples of solving six constrained benchmark functions.

(I) Algorithm	(J) Algorithm	Mean difference (I − J)	Std.	Sig.
HS	NSHS	**.11043867790257***	.028	**.000**
IHS	NSHS	**.24772610041475***	.028	**.000**
HSw	NSHS	**.20691985495254***	.028	**.000**
DLHS	NSHS	.01430413405475	.028	.640
SGHS	NSHS	**.06953695639486***	.028	**.028**

*The mean difference is significant at the 0.05 level. The test is based on observed means of the dependent variable (relative error).

the problem of the optimal balance between search diversity and intensity.

Acknowledgments

Many thanks to the anonymous reviewer for valuable comments that helped to improve this paper. This work was partly supported by the National Natural Science Foundation of China under Grant 71101003, 91224007, and 71332003 and the Fundamental Research Funds for the Central Universities under Grant YWF-13-D2-JC-12.

References

[1] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[2] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[3] Z. W. Geem, "State-of-the-art in the structure of harmony search algorithm," in *Studies in Computational Intelligence*, Z. W. Geem, Ed., pp. 1–10, Springer, Berlin, Germany, 2010.

[4] M. A. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Annals of Operations Research*, vol. 194, pp. 3–31, 2012.

[5] D. Zou, L. Gao, S. Li, and J. Wu, "Solving 0-1 knapsack problem by a novel global harmony search algorithm," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1556–1564, 2011.

[6] S. Sivasubramani and K. S. Swarup, "Multi-objective harmony search algorithm for optimal power flow problem," *International Journal of Electrical Power and Energy Systems*, vol. 33, no. 3, pp. 745–752, 2011.

[7] A. Verma, B. K. Panigrahi, and P. R. Bijwe, "Harmony search algorithm for transmission network expansion planning," *IET Generation, Transmission and Distribution*, vol. 4, no. 6, Article ID IGTDAW000004000006000663000001, pp. 663–673, 2010.

[8] L. Wang, Q.-K. Pan, and M. F. Tasgetiren, "Minimizing the total flow time in a flow shop with blocking by using hybridm harmony search algorithms," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7929–7936, 2010.

[9] M. P. Saka, "Optimum design of steel sway frames to BS5950 using harmony search algorithm," *Journal of Constructional Steel Research*, vol. 65, no. 1, pp. 36–43, 2009.

[10] Z. W. Geem, "Particle-swarm harmony search for water network design," *Engineering Optimization*, vol. 41, no. 4, pp. 297–311, 2009.

[11] Y. M. Cheng, L. Li, T. Lansivaara, S. C. Chi, and Y. J. Sun, "An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis," *Engineering Optimization*, vol. 40, no. 2, pp. 95–115, 2008.

[12] Z. W. Geem and J. C. Williams, "Harmony search and ecological optimization," *International Journal of Energy and Environment*, vol. 1, pp. 150–154, 2007.

[13] Z. W. Geem, K. S. Lee, and Y. Park, "Application of harmony search to vehicle routing," *American Journal of Applied Sciences*, vol. 2, pp. 1552–1557, 2005.

[14] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[15] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.

[16] C.-M. Wang and Y.-F. Huang, "Self-adaptive harmony search algorithm for optimization," *Expert Systems with Applications*, vol. 37, no. 4, pp. 2826–2837, 2010.

[17] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation*, vol. 216, no. 3, pp. 830–848, 2010.

[18] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic subpopulations," *Engineering Optimization*, vol. 42, no. 2, pp. 101–117, 2010.

[19] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267–283, 2009.

[20] O. M. Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 49–68, 2011.

[21] Y. Xin-She, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithms: Theory and Applications*, Z. W. Geem, Ed., pp. 1–13, Springer, 2009.

[22] S. García, D. Molina, M. Lozano, and F. Herrera, "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," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.