Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 413565, 29 pages
http://dx.doi.org/10.1155/2013/413565

Research Article

An Improved Harmony Search Based on Teaching-Learning Strategy for Unconstrained Optimization Problems

Shouheng Tuo,[1] Longquan Yong,[1] and Tao Zhou[2]

[1] *School of Mathematics and Computer Science, Shaanxi University of Technology, Hanzhong, Shaanxi 723001, China*
[2] *School of Science, Ningxia Medical University, Yinchuan, Ningxia 750004, China*

Correspondence should be addressed to Shouheng Tuo; tuo_sh@126.com

Received 22 August 2012; Accepted 12 November 2012

Academic Editor: Baozhen Yao

Harmony search (HS) algorithm is an emerging population-based metaheuristic algorithm, which is inspired by the music improvisation process. The HS method has been developed rapidly and applied widely during the past decade. In this paper, an improved global harmony search algorithm, named harmony search based on teaching-learning (HSTL), is presented for high dimension complex optimization problems. In HSTL algorithm, four strategies (harmony memory consideration, teaching-learning strategy, local pitch adjusting, and random mutation) are employed to maintain the proper balance between convergence and population diversity, and dynamic strategy is adopted to change the parameters. The proposed HSTL algorithm is investigated and compared with three other state-of-the-art HS optimization algorithms. Furthermore, to demonstrate the robustness and convergence, the success rate and convergence analysis is also studied. The experimental results of 31 complex benchmark functions demonstrate that the HSTL method has strong convergence and robustness and has better balance capacity of space exploration and local exploitation on high dimension complex optimization problems.

1. Introduction

With the development of scientific technology, many real-life optimization problems are becoming more and more complex and difficult. So how to resolve the complex problems in an exact manner within a reasonable time cost is very important. The traditional optimization algorithms are difficult to solve the nonlinear and nondifferential problems. In recent years, most popular swarm intelligence optimization algorithms, such as genetic algorithm (GA), particle swarm optimization (PSO), and differential evolution (DE) algorithm, have been successfully applied to large-scale complicated problems of scientific and engineering computing.

Inspired by the process of the musicians' improvisation of the harmony, the harmony search (HS) algorithm is proposed by Geem et al. [1, 2]. Similar to the GA and PSO, the HS algorithm is a meta-heuristic random optimization algorithm. In recent years, HS has been applied more broadly in the fields of engineering optimization. Such as pipe network design [3], structural optimization [4], clustering of text document [5], combined heat and power economic dispatch problem [6], and scheduling of multiple dam system [7]. The aforementioned applications show that HS algorithm has significant in solving complex engineering application problems. It has strong ability of exploration and has a cheap running cost. However, the classical harmony search algorithm is not efficient enough for solving large-scale problems, which has a slow convergence speed and low-precision solution. So some improved HS algorithms were proposed. Mahdavi et al. proposed an improved HS algorithm (IHS) [8] that employed a novel method generating new solution vectors which enhanced accuracy and convergence speed. Recently, Omran and Mahdavi tried to improve the performance of HS by incorporating some techniques from swarm intelligence. Tuo and Yong presented an improved harmony search algorithm with chaos (HSCH) [9]. The new variant named GHS (Global Best Harmony Search) [10] reportedly outperformed the HS and IHS algorithm over the benchmark problems.

HS algorithm has strong ability on exploring the regions of the solution space in a reasonable time. However, it has lower exploitation ability in later period. Therefore, some improved HS algorithm is proposed to enhance the

local search ability and solution precision, such as local-best harmony search algorithm with dynamic subpopulations (DLHS) [11], harmony search algorithm with differential mutation operator (HSDE) [12], and novel global harmony search algorithm for unconstrained problems (NGHS) [13, 14]. In addition, Gao et al. proposed modified harmony search methods for unimodal and multimodal optimization [15]. Pan et al. proposed self-adaptive harmony search algorithm for optimization (SGHS) [16]. Yadav et al. presented an intelligent tuned HS algorithm [17].

Now some existing state-of-the-art harmony search algorithms, such as NGHS, HSDE, and SGHS, have shown exceptional problem-solving ability, but they have some disadvantages over the multimodal and high dimensional function. To enhance the performance of high dimensional multimodal problem by HS method, this paper proposed an improved large-scale HS algorithm with harmony search teaching-learning (HSTL). In the HSTL algorithm, the HS algorithm is improved by embedding teaching-learning-based-optimization (TLBO) [17, 18] method which has a strong searching capacity for high dimensional problem.

The rest of this paper is organized as follows. In Section 2, the basic framework of the classical harmony search method is summarized in a comprehensive style, and the two excellent variants of HS (SGHS and NGHS) are briefly presented. A novel teaching-learning-based-optimization algorithm is provided, and the details of HSTL algorithm are presented in Section 3. In Section 4, 31 different characteristic benchmark problems, which consist of separable problems, nonseparable problems, shifted problems, shifted rotated problems, and hybrid composite problems, are considered, and the numerical results of HSTL method are demonstrated. Furthermore, to investigate the robustness and convergent performance of the HSTL algorithm, the comparison results of convergence speed and success rate are presented, and convergence analysis is preliminary put out. Finally, the research findings and contributions of the proposed HSTL algorithm are discussed in Section 5.

2. HS Algorithm and Other Variants

In this section, we introduce the classical harmony search (HS) algorithm and two excellent variants of HS algorithms: self-adaptive global-best harmony search (SGHS) algorithm and novel global harmony search (NGHS) algorithm.

2.1. Classical Harmony Search Algorithm (HS). The steps in the procedure of classical harmony search algorithm are as follows.

Step 1 (initialize the problem and algorithm parameters). In this step, the optimization problem is specified as follows:

$$\begin{aligned} &\min \quad f(x), \\ &\text{s.t.} \quad x = (x_1, x_2, \ldots, x_D), \\ &\qquad x_i \in \left[x_i^L, x_i^U\right], \quad i = 1, 2, \ldots, D, \end{aligned} \tag{1}$$

where $f(x)$ is an objective function, D is the number of decision variables.

The HS algorithm parameters are also specified in this step.

HMS: the harmony memory size or the number of solution vectors in the population;

HMCR: the harmony considering rate;

PAR: pitch adjusting rate;

BW: bandwidth;

maxFEs: the maximum number of improvisations.

Step 2 (initialize the harmony memory). The harmony memory (HM) consists of HMS harmony vectors. Each harmony vector is generated from a uniform distribution in the feasible space, as

$$\begin{aligned} &x_i^j = x_i^L + \left(x_i^U - x_i^L\right) \times \text{rand}(), \\ &i = 1, 2, \ldots, D; \quad j = 1, 2, \ldots, \text{HMS}, \end{aligned} \tag{2}$$

where rand() is a random between 0 and 1.

$$\text{HM} = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^{\text{HMS}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_D^1 \\ x_1^2 & x_2^2 & \cdots & x_D^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_D^{\text{HMS}} \end{bmatrix}. \tag{3}$$

Step 3 (improvise a new harmony). A new harmony vector $x^{\text{new}} = (x_1^{\text{new}}, x_2^{\text{new}}, \ldots, x_D^{\text{new}})$ is generated based on three rules:

(a) Memory consideration;

(b) Pitch adjustment;

(c) Random generation.

The improvisation procedure of new harmony vector works as Algorithm 1.

A new potential variation (or an offspring) is generated in Step 3, which is equivalent to mutation and crossover operator in standard evolution algorithms (EAs).

Step 4 (update harmony memory). Get the worst harmony memory x^{worst} from the HM. If x^{new} is better than x^{worst} then $x^{\text{worst}} := x^{\text{new}}$.

Step 5 (check stopping criterion). If the stopping criterion (maxFEs) is satisfied, computation is terminated. Otherwise, Steps 3 and 4 are repeated.

2.2. The SGHS Algorithm. In order to select the best parameters automatically, a self-adaptive global-best harmony search (SGHS) algorithm is proposed by Pan et al. [16]. The SGHS dynamically changes BW according to

$$\text{BW}(t) = \begin{cases} \text{BW}_{\max} - \dfrac{\text{BW}_{\max} - \text{BW}_{\min}}{T_{\max}} \times 2t, & \text{if } t < T_{\max}, \\ \text{BW}_{\min}, & \text{if } t \geq T_{\max}, \end{cases} \tag{4}$$

For i = 1 to D do
 If rand( ) ≤ HMCR then
  $x_i^{\text{new}} = x_i^j,\ j \in U\{1, 2, \ldots, \text{HMS}\}$
  If rand( ) ≤ PAR then
   $x_i^{\text{new}} = x_i^{\text{new}} \pm \text{rand}( ) \times \text{BW}(i)$
   $x_i^{\text{new}} = \min(\max(x_i^{\text{new}}, x_i^L), x_i^U)$
  End If
 Else
  $x_i^{\text{new}} = x_i^L + (x_i^U - x_i^L) \times \text{rand}( )$
 End If
End For

Algorithm 1: The improvisation procedure of new harmony vector by HS.

For i = 1 to D do
 $x_i^r = 2 \times x_i^{\text{best}} - x_i^{\text{worst}}$
 $x_i^r = \min(\max(x_i^r, x_i^L), x_i^U)$
 $x_i^{\text{new}} = x_i^{\text{worst}} + \text{rand}( ) \times (x_i^r - x_i^{\text{worst}})$.
 If rand( ) ≤ p_m //random mutation
  $x_i^{\text{new}} = x_i^L + (x_i^U - x_i^L) \times \text{rand}( )$
 End If
End For.

Algorithm 2: The NGHS modifies improvisation step.

where $\text{BW}_{\max}$ and $\text{BW}_{\min}$ are the maximum and minimum distance bandwidths.

HMCR (PAR) is dynamically selected from a suitable normally distribution with mean HMCRm (PARm) and standard deviation 0.01 (0.05). Initially, HMCRm (PARm) is set at 0.98 (0.9). After a specified number of generations, HMCRm (PARm) is recalculated by averaging all the recorded HMCR (PAR) values during the period of evolution.

2.3. The NGHS Algorithm. In the novel global harmony search (NGHS) algorithm [13, 14], three significant parameters, harmony memory considering rate (HMCR), bandwidth (BW), and pitch adjusting rate (PAR), are excluded from NGHS, and a random select rate (p_m) is included in the NGHS. In Step 3, NGHS works as Algorithm 2.

Where x^{best} and x^{worst}, respectively, are the best harmony and the worst harmony in HM, rand() is uniformly generated random number in [0, 1].

3. HSTL Algorithm

In this section, we proposed a novel harmony search method with teaching-learning (HSTL) strategy which was derived from teaching-learning-based optimization (TLBO) algorithm. Above all, the TLBO algorithm is introduced and analyzed, and then we focus on the details of HSTL algorithm and the strategies of dynamically adjusting the parameters.

3.1. The TLBO Algorithm. Teaching-learning-based Optimization (TLBO) algorithm [18–20] is a new nature-inspired algorithm; it mimics the teaching process of teacher and learning process among learners in a class. TLBO shows a better performance with less computational effort for large scale problems [19], that is, problems of a high dimensionality. In addition, TLBO needs very few parameters.

For each learner x^j $(j = 1, 2, \ldots, \text{NP})$
 Randomly select another learner x^k $(j \neq k)$
  If x^j is superior to x^k **then**
   $x^{j,\text{new}} = x^{j,\text{old}} + \text{rand}( ) \times (x^j - x^k)$
  Else
   $x^{j,\text{new}} = x^{j,\text{old}} + \text{rand}( ) \times (x^k - x^j)$
  End
End for
If $x^{j,\text{new}}$ is superior to $x^{j,\text{old}}$ **then**
 $x^j = x^{j,\text{new}}$
End If

Algorithm 3: The procedure of learner phase.

In the TLBO method, it is the task of teacher to try to increase mean knowledge of all learners of the class in the subject taught by him or her depending on his or her capability. Learners make efforts to increase their knowledge by interaction among themselves. A learner is considered as a solution or a vector, and different design variables of vector will be analogous to different subjects offered to learners and the learners' result is analogous to the "fitness" as in other population-based optimization techniques. The teacher is considered as the best solution obtained so far. The process of working of TLBO is divided into two phases, teacher phase and learner phase.

(1) *Teacher Phase.* Assume there are D number of subjects (i.e., design variables), "NP" number of learners (i.e., population size,), then x_i^{best} is the best learner (i.e., teacher) in subject i $(i = 1, 2, \ldots, D)$. The works of teaching are as follows:

$$x_i^{j,\text{new}} = x_i^{j,\text{old}} + \text{rand}\,() \times \left(x_i^{\text{best}} - T_F \times \text{Mean}_i\right),$$
$$\text{Mean}_i = \frac{1}{\text{NP}} \sum_{j=1}^{\text{NP}} x_i^j, \quad j = 1, 2, \ldots, \text{NP},\ i = 1, 2, \ldots, D, \tag{5}$$

where $x_i^{j,\text{old}}$ denotes the result of the jth $(j = 1, 2, \ldots, \text{NP})$ learner before learning the ith $(i = 1, 2, \ldots, D)$ subject, and $x_i^{j,\text{new}}$ is the result of the jth learner after learning the ith subject. T_F is the teaching factor which decides the value of Mean_i to be changed. The value of T_F is generated randomly with probability as $T_F = \text{round}[1 + \text{rand}()]$.

When the leaner x^j finished his or her learning from the teacher, update the x^j by $x^{j,\text{new}}$ if $x^{j,\text{new}}$ is better than $x^{j,\text{old}}$.

(2) *Learner Phase.* Another important approach to increase knowledge for a learner is to interact with other learners. Learning method is expressed in Algorithm 3.

Even since the TLBO algorithm proposed in 2011 by Rao et al. [18], it has been applied in the fields of engineering optimization, such as mechanical design optimization [18, 21],

heat exchangers [22], thermoelectric cooler [23], and unconstrained and constrained real parameter optimization problems [24].

In the TLBO method, teacher phase relying on the best solution found so far usually has the fast convergence speed and the best ability of exploitation; it is more suitable for improving accuracy of the global optimal solution. Learner phase relying on other learners usually has the slow convergence speed; however, it bears stronger exploration capability for solving multimodal problems. Therefore, we use the TLBO method to improve the performance and efficiency of HS algorithm in the HSTL method.

3.2. The HSTL Algorithm. In order to achieve the most satisfactory optimization performance by applying the HS algorithm to a given problem, we develop a novel harmony search algorithm combined with teaching-learning strategy, in which both new harmony generation strategies and associated control parameter values can be dynamically changed according to the process of evolution.

It is of a high importance between the convergence and the diversity in order to improve the search efficiency. In the classical HS algorithm, a new harmony is generated in Step 3. After the selecting operation in Step 4, the population variance may increase or decrease. With a high population variance, the diversity and exploration power will increase, in the same time the convergence and the exploitation power will decrease accordingly. Conversely, with a low population variance, the convergence and the exploitation power will increase [25]; the diversity and the exploration power will decrease. So it is significant how to keep balance between the convergence and the diversity. Classical HS algorithm loses its ability easily at later evolution process [26], because of improvising new harmony from HM with a high HMCR and local adjusting with PAR. And HM diversity decreases gradually from the early iteration to the last. However, in HS algorithm, a low HMCR employed will increase the probability (1-HMCR) of random selection in search space; the exploration power will enhance, but the local search ability and the exploitation accuracy cannot be improved by single pitch adjusting strategy.

To overcome the inherent weaknesses of HS, in this section, we propose HSTL method. An improved teaching-learning strategy is employed to improve the search ability of optimal solution in the HSTL method. The HSTL algorithm works as follows.

(1) Optimization target vector preparation: $x^{\text{new}} = x^r$, where x^r ($r = 1, 2, \ldots, \text{HMS}$) is a harmony vector selected randomly in HM. Next, four strategies are employed to improve the target vector.

(2) Improve the target vector x^{new} with the following 4 strategies.

(a) *Harmony Memory Consideration.* The ith design value of the target vector x_i^{new}, ($i = 1, 2, \ldots, D$) is chosen randomly from harmony memory (HM) with a probability of HMCR as

$$x_i^{\text{new}} = x_i^j, \quad j \in \text{U}\{1, 2, \ldots, \text{HMS}\}, \; i = 1, 2, \ldots, D. \tag{6}$$

The HMCR is the rate of choosing one value from the historical values stored in the HM, which varies between 0 and 1.

(b) *Teaching-Learning Strategy.* If the ith ($i = 1, 2, \ldots, D$) design variable of the target vector x_j^{new} has not been considered in HM, it will learn from the best harmony (i.e., teacher) with probability TLP in the teacher phase or from other harmony (i.e., learner) in the learner phase. The TLP is the probability of performing teaching-learning operator on design variables that have not been considered in (a). It works as follows.

Teacher Phase. In this phase, learner will learn from the best learner (i.e., teacher) in class. Learner modification is expressed as

$$\begin{aligned} x_i^{\text{new}} &= x_i^{\text{new}} + \text{rand}() \times \left[x_i^{\text{best}} - T_F \times M_i\right], \\ M_i &= \frac{\left(x_i^{\text{worst}} + x_i^{\text{new}}\right)}{2}, \quad i = 1, 2, \ldots, D, \end{aligned} \tag{7}$$

where x^{best} is the best harmony in HM, x^{worst} is the worst harmony in HM, and rand() is a uniformly distributed random number between 0 and 1.

The contribution in this section is that the mean value of population is replaced with $M_i = (x_i^{\text{worst}} + x_i^{\text{new}})/2$. There are two aspects to this apparent superiority. First, there are cheap to running cost because it do not need to compute the mean value of population in every iteration. Second, diversity of population will be enhanced more than the standard TLBO algorithm, that is because the new mean value M_i will be different for each individual, but the Mean_i in standard TLBO method is the same for every individual.

Learner Phase. In this phase, through comparing the advantages and disadvantages between the other two learners, the learner x^{new} will learn from their advantages, which draw on the idea of differential evolution algorithm. The process is as follows.

Randomly select two differential integer r_1 and r_2 from $1, 2, \ldots, \text{HMS}$.

If x^{r_1} is better than x^{r_2}

$$x_i^{\text{new}} = x_i^{\text{new}} + \text{rand}() \times (x_i^{r_1} - x_i^{r_2})$$

Else

$$x_i^{\text{new}} = x_i^{\text{new}} + \text{rand}() \times (x_i^{r_2} - x_i^{r_1})$$

End If

(c) *Local Pitch Adjusting Strategy.* To achieve better solutions in the search space, it will carry out the local pitch adjusting strategy with probability PAR when the ith design variable has not performed Harmony memory consideration and Teaching-Learning Strategy as

$$x_i^{\text{new}} = x_i^{\text{new}} \pm \text{rand}() \times \text{BW}(i), \tag{8}$$

$x^{\text{new}} = x^r$ ($r = 1, 2, \ldots$, HMS); // randomly select x^r as optimization target vector
For i = 1 to D do
  If rand( ) ≤ HMCR //(a) ***Harmony memory consideration***
    $x_i^{\text{new}} = x_i^j$ ($j = 1, 2, \ldots,$ HMS)
  Else If rand( ) ≤ TLP //(b) ***Teaching-Learning strategy***
    If rand( ) ≤ 0.5
      // ***Teaching***
      $x_i^{\text{new}} = x_i^{\text{new}} + \text{rand}( ) \times [x_i^{\text{best}} - 0.5T_F \times (x_i^{\text{worst}} + x_i^{\text{new}})]$
    Else
      // ***Learning***
      Randomly select r_1 and r_2 from {1, 2, ..., HMS}
      If x^{r_1} is better than x^{r_2}
        $x_i^{\text{new}} = x_i^{\text{new}} + \text{rand}( ) \times (x_i^{r_1} - x_i^{r_2})$
      Else
        $x_i^{\text{new}} = x_i^{\text{new}} + \text{rand}( ) \times (x_i^{r_2} - x_i^{r_1})$
      End
    End
    $x_i^{\text{new}} = \min(\max(x_i^{\text{new}}, x_i^L), x_l^U)$
  Else If rand(0, 1) ≤ PAR // (c) ***Local pitch adjusting strategy***
    $x_i^{\text{new}} = x_i^{\text{new}} \pm \text{rand}( ) \times \text{BW}(i)$
    $x_i^{\text{new}} = \min(\max(x_i^{\text{new}}, x_i^L), x_l^U)$
  Else If rand(0, 1) ≤ P_m // (d) ***Random mutation operator***
    $x_i^{\text{new}} = x_i^L + (x_i^U - x_i^L) \times \text{rand}( )$
  End If
End For

Algorithm 4: The improvisation process of new harmony in HSTL algorithm.

where rand() is a uniformly distributed random number between 0 and 1, and BW(i) is an arbitrary distance bandwidth.

(d) *Random Mutation Operator.* As with the HS algorithm, if design variable did not perform previous actions (harmony memory consideration, teaching-learning strategy, and local pitch adjusting strategy), the HSTL method will carry out random mutation operator in feasible space with probability P_m on ith design variable of x^{new} as follows:

$$x_i^{\text{new}} = x_i^L + \left(x_i^U - x_i^L\right) \times \text{rand}( ). \tag{9}$$

The improvisation of new target harmony in HSTL algorithm is shown in Algorithm 4.

The flow chart of HSTL algorithm is shown in Figure 1.

Parameters Dynamically Changed. To efficiently balance the exploration and exploitation power of the HSTL algorithm, HMCR, PAR, BW, and TLP parameters are dynamically adapted to a suitable range with the increase of generations. Equation (10) shows the dynamic change of HMCR, PAR, BW, and TLP, respectively.

Consider the following [8]:

$$\begin{aligned}
\text{HMCR} &= \text{HMCR}_{\min} + \left(\text{HMCR}_{\max} - \text{HMCR}_{\min}\right) \times \left(\frac{t}{T_{\max}}\right)^2, \\
\text{TLP} &= \text{TLP}_{\min} + \left(\text{TLP}_{\max} - \text{TLP}_{\min}\right) \times \left(\frac{t}{T_{\max}}\right)^k, \quad k = 5, \\
\text{PAR} &= \text{PAR}_{\max} - \frac{\left(\text{PAR}_{\max} - \text{PAR}_{\min}\right) \times t}{T_{\max}}, \\
\text{BW} &= \text{BW}_{\max} + \exp\left[\ln\left(\frac{\text{BW}_{\min}}{\text{BW}_{\max}}\right) \times \sqrt{\frac{t}{T_{\max}}}\right].
\end{aligned} \tag{10}$$

4. Numerical Experiments and Results

4.1. Test Functions. The test functions F_1–F_{23} [27–29] are shown in Table 1, and F_{24}–F_{31} are demonstrated. Table 2 in Functions F_{24}–F_{31} [30] are hybrid composition functions. The hybrid composition functions are built combining a nonseparable (NS) function with other functions. The considered functions are as folows.

(i) Nonseparable functions:

  (a) F_{15}: shifted Rosenbrock's function,
  (b) F_{17}: shifted Griewank's function,
  (c) NS-F_{21}: nonshifted Extended F_{10},
  (d) NS-F_{22}: nonshifted Bohachevsky.

(ii) Other component functions:

  (a) F_{13}: shifted sphere function,
  (b) F_{16}: shifted Rastrigin function,
  (c) F_{20}: Schwefel2.22 function.

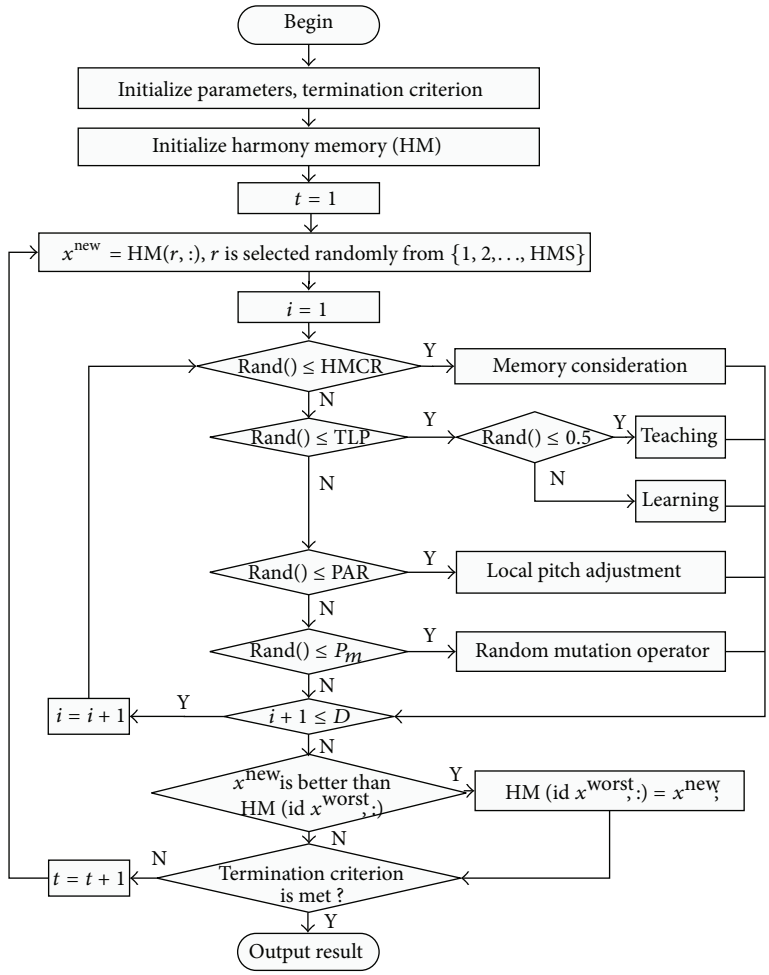

Figure 1: The flowchart of HSTL algorithm.

Hybrid function $F_{ns} \oplus F'(S)$ is composed of a nonseparable function F_{ns} and other function F'. The hybrid procedure is shown as follows. (1) S is divided into two parts ($part_1$ and $part_2$) as follows

If $m_{ns} \le 0.5$ then

$part_1$ is composed by the first $D \cdot m_{ns}$ even variables. (length ($part_1$) = $D \cdot m_{ns}$)

$part_2$ is composed by the remaining variables. (length ($part_2$) = D − length ($part_1$))

Else

$part_2$ is composed by the first $D \cdot (1 - m_{ns})$ odd variables. (length ($part_2$) = $D \cdot (1 - m_{ns})$)

$part_1$ is composed by the remaining variables. (length ($part_1$) = D − length ($part_2$))

End If

(2) Return F_{ns} ($part_1$) $\oplus$ F'($part_2$).

We have used 31 well-known unconstrained benchmark functions [27–30] as a testbed to evaluate the performance of the HSTL algorithm presented in this paper. These test functions are considered as particularly challenging for many existing meta-heuristic optimization algorithms except for F_{10}. The composition of these test functions contains some characteristics, such as separable design variables, nonseparable design variables, strong unimodality, strong multimodality, and hybrid. Many of them blend different characteristics together. Table 3 shows the characteristics of all test functions.

4.2. Experimental Setup. Simulation experiments are carried out to compare the optimization (minimization) capabilities of the presented method (HSTL) with respect to (a) classical HS [1, 2], (b) SGHS [16], and (c) NGHS [13, 14]. All the experiments were performed on Windows XP 32 system with Intel(R) Core(TM) i3-2120 CPU@3.30 GHz and 2 GB RAM, and all the program codes were written in MATLAB R2009a.

In the experiments, the parameters settings for the compared HS algorithms are shown in Table 4; the dimensions of the benchmark problems are set as 50, 100, and 200, respectively. To make the comparison fair, the populations for all the competitor algorithms (for all problems tested) were initialized using the same random seeds. The variants

Table 1: Benchmark test functions.

Function name	Benchmark functions expression	Search range	Optimum value
F_1: Ackley function	$F_1(x) = -20e^{(1/5)\sqrt{(1/D)\sum_{i=1}^{D} x_i^2}} - e^{(1/D)\sum_{i=1}^{D}\cos(2\pi x_i)} + 20 - e$	$[-15, 30]^D$	$x^* = (0, 0, \ldots, 0)$, $F_1(x^*) = 0$
F_2: Dixon and Price function	$F_2(x) = (x_1 + 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2$	$[-10, 10]^D$	$F_2(x^*) = 0$
F_3: Griewank function	$F_3 = (1/4000)\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}(\cos(x_i/\sqrt{i})) + 1$	$[-600, 600]^D$	$x^* = (0, 0, \ldots, 0)$, $F_3(x^*) = 0$
F_4: Levy function	$F_4(x) = \sin^2(\pi y_1) + \sum_{i=1}^{D-1}\left[(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1))\right] + (y_D - 1)^2(1 + 10\sin^2(2\pi y_D))$, where $y_i = 1 + ((x_i - 1)/4), i = 1, 2, \ldots, D$	$[-10, 10]^D$	$x^* = (1, 1, \ldots, 1)$, $F_4(x^*) = 0$
F_5: Michalewics function	$F_5(x) = -\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{2m}$; $m = 10$	$[-10, \pi]^D$	$n = 5$; $F_5(x^*) = -4.687658$
F_6: Powell function	$F_6(x) = \sum_{i=1}^{D/4}\left[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^2 + 10(x_{4i-3} - x_{4i})^4\right]$	$[-4, 5]^D$	$x^* = (3, -1, 0, 1, \ldots, 3, -1, 0, 1)$, $F_6(x^*) = 0$
F_7: Rastrigin function	$F_7(x) = 10D + \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i))$	$[-5.12, 5.12]^D$	$x^* = (0, 0, \ldots, 0)$, $F_7(x^*) = 0$
F_8: Rosenbrock function	$F_8(x) = \sum_{i=1}^{D-1}\left[100(x_i^2 - x_{i-1})^2 + (x_i - 1)^2\right]$	$[-10, 10]^D$	$x^* = (1, 1, \ldots, 1)$, $F_8(x^*) = 0$
F_9: Schwefel2.26 function	$F_9(x) = 418.9829D + \sum_{i=1}^{D} x_i^2 \sin(\sqrt{\lvert x_i\rvert})$	$[-512, 512]^D$	$x^* = (420.9687, 420.9687, \ldots, 420.9687)$, $F_9(x^*) = 0$
F_{10}: Sphere function	$F_{10}(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	$x^* = (0, 0, \ldots, 0)$, $F_{10}(x^*) = 0$
F_{11}: Trid function	$F_{11}(x) = \sum_{i=1}^{D}(x_i - 1)^2 - \sum_{i=2}^{D} x_i x_{i-1}$	$[-D^2, D^2]^D$	$D = 5$; $F_{11}(x^*) = -30$
F_{12}: Zakharov function	$F_{12}(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^4$	$[-5, 10]^D$	$x^* = (0, 0, \ldots, 0)$, $F_{12}(x^*) = 0$
F_{13}: Sphere shift function	$F_{13}(x) = \sum_{i=1}^{D} z_i^2 + f\_\text{bias}_{13}$, $f\_\text{bias}_{13} = -450$, $z = x - o$; $o = [o_1, o_2, ..., o_D]$	$[-100, 100]^D$	$x^* = o$, $F_{12}(x^*) = -450$
F_{14}: Schwefel shift function	$F_{14} = \max(\lvert z\rvert) + f\_\text{bias}_{14}$, $f\_\text{bias}_{14} = -450$, $z = x - o$; $o = [o_1, o_2, ..., o_D]$	$[-100, 100]^D$	$x^* = o$, $F_{14}(x^*) = -450$
F_{15}: Rosenbrock shift function	$F_{15}(x) = \sum_{i=1}^{D-1}\left[100(z_i^2 - z_{i-1})^2 + (z_i - 1)^2\right] + f\_\text{bias}_{15}$, $f\_\text{bias}_{15} = 390$, $z = x - o$; $o = [o_1, o_2, ..., o_D]$	$[-100, 100]^D$	$x^* = o$, $F_{15}(x^*) = 390$

Table 1: Continued.

Function name	Benchmark functions expression	Search range	Optimum value
F_{16}: Griewank shift function	$F_{16} = (1/4000)\sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D}(\cos(z_i/\sqrt{i})) + 1 + f\_\text{bias}_{16}$, $f\_\text{bias}_{16} = -180$, $z = (x - o) \times M$, $o = [o_1, o_2, ..., o_D]$	$[-600, 600]^D$	$x^* = o$, $F_{16}(x^*) = -180$
F_{17}: Rastrigin shift function	$F_{17}(x) = 10D + \sum_{i=1}^{D}\left(z_i^2 - 10\cos(2\pi z_i)\right) + f\_\text{bias}_{17}$, $f\_\text{bias}_{17} = -330$, $z = x - o$; $o = [o_1, o_2, ..., o_D]$	$[-5.12, 5.12]^D$	$x^* = o$, $F_{17}(x^*) = -330$
F_{18}: Ackley shift function	$F_{18}(x) = -20e^{(1/5)\sqrt{(1/D)\sum_{i=1}^{D} z_i^2}} - e^{(1/D)\sum_{i=1}^{D}\cos(2\pi z_i)} + 20 - e$, $f\_\text{bias}_{18} = -140$, $z = x - o$; $o = [o_1, o_2, ..., o_D]$	$[-32, 32]^D$	$x^* = o$, $F_{18}(x^*) = -140$
F_{19}: Fast Fractal "Double Dip" function	$F_{19}(x) = \sum_{i=1}^{D} \text{fractal1}D(x_i + \text{twist}(x_{(i \bmod D)+1}))$ $\text{twist}(y) = 4(y^4 - 2y^3 + y^2)$ $\text{fractal}\,1D(x) \approx \sum_{k=1}^{3}\sum_{1}^{2^{k-1}}\sum_{1}^{\text{ran2}(o)} \text{doubledip}(x, \text{ran}\,1(o), \frac{1}{2^{k-1}(2 - \text{ran}\,1(o))})$ $\text{doubledip}(x, c, s) = \begin{cases} c(-6144(x-c)^6 + 3088(x-c)^4 - 392(x-c)^2 + 1)s \\ 0, \ \text{otherwise} \end{cases}$ ran 1 (o): double, pseudo randomly chosen, with seed o, with equal probability from the interval $[0, 1]$ ran 2(o): integer, pseudo randomly chosen, with seed o, with equal probability from the set $\{0, 1, 2\}$ fractal $1D(x)$ is an approximation to a recursive algorithm, it does not take account of wrapping at the boundaries, or local reseeding of the random generators.	$[-1, 1]^D$	Unknown
F_{20}: Schwefel2.22 function	$F_{20}(x) = \sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	$[-10, 10]^D$	$x^* = (0, 0, ..., 0)$, $F_{20}(x^*) = 0$
F_{21}: Extended_f10 shift function	$F_{21}(x) = \left(\sum_{i=1}^{D-1} f_{10}(z_i, z_{i+1})\right) + f_{10}(z_D, z_1), \quad z = x - o$ $f_{10} = (x^2 + y^2)^{0.25} \cdot (\sin^2(50 \cdot (x^2 + y^2)^{0.1}) + 1)$	$[-100, 100]^D$	$x^* = o$, $F_{21}(x^*) = 0$
F_{22}: Bohachevsky function	$F_{22}(x) = \sum_{i=1}^{D}(z_i^2 + 2z_{i+1}^2 - 0.3\cos(3\pi z_i) - 0.4\cos(4\pi z_{i+1}) + 0.7), \quad z = x - o$	$[-15, 15]^D$	$x^* = o$, $F_{22}(x^*) = 0$
F_{23}: Schaffer shift function	$F_{23}(x) = \sum_{i=1}^{D-1}(z_i^2 + z_{i+1}^2)^{0.25}(\sin^2(50 \cdot (z_i^2 + z_{i+1}^2)^{0.1}) + 1), \quad z = x - o$	$[-100, 100]^D$	$x^* = o$, $F_{23}(x^*) = 0$

Table 2: The hybrid functions (F_{24}–F_{31}).

Function	F_{ns}	F'	m_{ns}	Range	Fitness optimum
F_{24}	NS-F_{21}	F_{13}	0.25	$[-100, 100]^D$	−450
F_{25}	NS-F_{22}	F_{15}	0.25	$[-100, 100]^D$	390
F_{26}	NS-F_{23}	F_{16}	0.25	$[-5, 5]^D$	−330
F_{27}	NS-F_{22}	NS-F_{20}	0.25	$[-10, 10]^D$	0
F_{28}	NS-F_{21}	F_{13}	0.5	$[-100, 100]^D$	−450
F_{29}	NS-F_{21}	F_{15}	0.75	$[-100, 100]^D$	390
F_{30}	NS-F_{21}	F_{16}	0.75	$[-5, 5]^D$	−330
F_{31}	NS-F_{22}	NS-F_{20}	0.75	$[-10, 10]^D$	0

Table 3: The characteristics of all test functions.

Function	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
Unimodal (U)/multimodal (M)	M	M	M	M	M	M	M	M	M	U	U	U	U	U	M	M
Shifted (Y/N)	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y
Separable (Y/N)	Y	N	N	Y	Y	N	Y	N	Y	Y	Y	Y	Y	N	N	Y
Hybrid (Y/N)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Function	F_{17}	F_{18}	F_{19}	F_{20}	F_{21}	F_{22}	F_{23}	F_{24}	F_{25}	F_{26}	F_{27}	F_{28}	F_{29}	F_{30}	F_{31}	
Unimodal (U)/multimodal (M)	M	M	M	U	U	U	U	U	M	M	M	U	M	M	N	
Shifted (Y/N)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	
Separable (Y/N)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
Hybrid (Y/N)	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	

Table 4: Parameter settings for the compared HS algorithms (HS, SGHS, NGHS, HSTL).

Algorithm	HMS	HMCR	PAR	BW	Others
HS	35	0.98	0.33	BW = (UB − LB)/1000	
SGHS	35	$HMCR_m = 0.9$	$PAR_{max} = 0.99$, $PAR_{min} = 0.01$	$BW_{max} = (UB - LB)/10$ $BW_{min} = 0.0005$	LP = 100
NGHS	35	/	/	/	$P_m = 0.01$
HSTL	35	$HMCR_{max} = 0.99$	$PAR_{max} = 0.8$	$BW_{max} = (UB - LB)/20$	$TLP_{max} = 0.5$, $TLP_{min} = 0.15$
		$HMCR_{min} = 0.75$	$PAR_{min} = 0.2$	$BW_{min} = (UB - LB)/3500$	$P_m = 0.15$, $k = 5$

of HS algorithm were set at the same termination criteria: the number of improvisations (function evaluation times: FEs) FEs = 500D, respectively.

The best and worst fitness values of each benchmark problem are recorded for 30 independent runs; the mean fitness, standard deviation (STD), and mean runtime of each function are calculated for 30 independent runs, and the Mean fitness, Standard Deviation (STD) and mean runtime of each function are calculated for 30 independent runs.

4.3. The Results Comparison between HSTL, HS, SGHS, and NGHS Algorithm. From Table 5, it can be found that the mean optimal fitness values of HSTL are always less than those of other 3 HS algorithms for all test functions except for F_{12}, F_{14}, F_{16}, and F_{29} when dimension are equal to 50 and maxFEs = 25000, and all these mean optimal fitness values are slightly larger than the NGHS algorithm for F_{12}, F_{14}, F_{16}, and F_{29}. According to criterions (best, mean, worst, and STD), the over performance of HSTL method outperforms the other three HS algorithms for 31 benchmark problems except F_{12}, F_{14}, and F_{29}. Taken together, the beast, mean, and STD obtained by the HSTL method are better than those of other three methods for most test functions. In addition, the runtime is less than SGHS for all problems.

In Tables 6 and 7, the results are shown for 100D and 200D problems, respectively. According to the results (best, mean, worst, and STD) of HSTL algorithm outperforms the other three harmony search algorithm. And it can be seen from Tables 6 and 7 that, compared with 50D problems, the HSTL method has more obvious advantages than other three HS algorithms (HS, SGHS, and NSGH).

Table 5: The optimization results of four harmony search algorithms for F_1–F_{31} ($D = 50$, FEs = $500D$).

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_1	HS	$2.566680E+00$	$3.302045E+00$	$3.884725E+00$	$3.248466E-01$	**$5.916126E-01$**
	SGHS	$1.813356E+00$	$2.557675E+00$	$3.129460E+00$	$3.608468E-01$	$1.539694E+00$
	NGHS	$3.447238E-02$	$5.592492E-02$	$8.840852E-02$	$1.464912E-02$	$6.187550E-01$
	HSTL	**$9.224783E-04$**	**$1.775338E-03$**	**$2.516676E-03$**	**$3.736810E-04$**	$1.150359E+00$
F_2	HS	$2.528891E+01$	$1.303365E+02$	$2.064702E+02$	$4.764531E+01$	**$3.728292E-01$**
	SGHS	$2.291379E+01$	$5.454353E+01$	$1.405853E+02$	$2.704011E+01$	$1.335320E+00$
	NGHS	$7.771989E-01$	$5.744259E+00$	$1.159369E+01$	$3.130084E+00$	$4.210442E-01$
	HSTL	**$6.666789E-01$**	**$7.870821E-01$**	**$1.185710E+00$**	**$1.526289E-01$**	$8.835301E-01$
F_3	HS	$1.498964E+00$	$1.663430E+00$	$1.988965E+00$	$1.398770E-01$	**$6.367849E-01$**
	SGHS	$1.190177E+00$	$1.475248E+00$	$2.130067E+00$	$2.263591E-01$	$1.593317E+00$
	NGHS	$4.418931E-02$	$7.146802E-02$	$1.559262E-01$	$2.679798E-02$	$6.544169E-01$
	HSTL	**$2.933239E-05$**	**$8.315104E-04$**	**$1.506785E-02$**	**$3.351017E-03$**	$1.209138E+00$
F_4	HS	$2.734048E-01$	$3.341458E-01$	$4.153356E-01$	$3.844929E-02$	**$1.368443E+00$**
	SGHS	$1.417380E-01$	$1.714592E+00$	$3.739141E+00$	$1.116863E+00$	$2.390772E+00$
	NGHS	$2.066964E-04$	$3.193237E-04$	$5.845768E-04$	$9.202494E-05$	$1.393585E+00$
	HSTL	**$4.120272E-06$**	**$6.258354E-06$**	**$9.738134E-06$**	**$1.414984E-06$**	$2.027401E+00$
F_5	HS	$-4.170235E+01$	$-4.036729E+01$	$-3.828157E+01$	$9.770373E-01$	**$8.758810E-01$**
	SGHS	$-4.660214E+01$	$-4.506838E+01$	$-4.284438E+01$	$9.381859E-01$	$1.822759E+00$
	NGHS	$-4.761973E+01$	$-4.653808E+01$	$-4.546623E+01$	$5.188843E-01$	$9.208168E-01$
	HSTL	**$-4.848365E+01$**	**$-4.783102E+01$**	**$-4.713305E+01$**	**$3.899320E-01$**	$1.454742E+00$
F_6	HS	$4.064420E+00$	$1.023338E+01$	$1.807872E+01$	$4.174841E+00$	**$1.270949E+00$**
	SGHS	$3.787277E+00$	$2.168953E+01$	$5.037661E+01$	$1.367783E+01$	$2.307965E+00$
	NGHS	$4.732459E-02$	$9.566685E-02$	$1.331749E-01$	**$2.187037E-02$**	$1.313182E+00$
	HSTL	**$3.072186E-02$**	**$7.276520E-02$**	**$1.260607E-01$**	$2.847471E-02$	$1.955321E+00$
F_7	HS	$2.373168E+01$	$3.076467E+01$	$3.740304E+01$	$3.698242E+00$	**$4.206309E-01$**
	SGHS	$4.212332E+01$	$5.047204E+01$	$5.912495E+01$	$5.715449E+00$	$1.387887E+00$
	NGHS	$4.155143E+00$	$8.088647E+00$	$1.126315E+01$	$1.970674E+00$	$4.668058E-01$
	HSTL	**$3.164383E-05$**	**$3.450611E-03$**	**$3.265254E-02$**	**$8.899514E-03$**	$9.399841E-01$
F_8	HS	$5.951376E+01$	$4.482384E+02$	$8.417699E+02$	$2.411457E+02$	**$3.847198E-01$**
	SGHS	$2.024308E+02$	$4.029735E+02$	$1.163300E+03$	$2.088718E+02$	$1.346647E+00$
	NGHS	**$2.246655E+01$**	$1.232892E+02$	$2.151188E+02$	$5.247493E+01$	$4.322725E-01$
	HSTL	$4.179333E+01$	**$7.883687E+01$**	**$1.109876E+02$**	**$3.016135E+01$**	$9.006008E-01$
F_9	HS	$1.891514E+02$	$2.441957E+02$	$3.147589E+02$	**$3.822653E+01$**	**$4.938682E-01$**
	SGHS	$4.914947E+02$	$1.155863E+03$	$1.616206E+03$	$3.125553E+02$	$1.506911E+00$
	NGHS	$1.177227E+00$	$1.939242E+02$	$5.928832E+02$	$1.437114E+02$	$5.429397E-01$
	HSTL	**$2.831025E-03$**	**$4.800827E+01$**	**$2.559129E+02$**	$8.324164E+01$	$1.055470E+00$

Table 5: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{10}	HS	$4.982674E+01$	$7.634414E+01$	$1.088877E+02$	$1.384184E+01$	**$5.155853E-01$**
	SGHS	$1.941824E+01$	$5.712271E+01$	$1.173837E+02$	$2.463712E+01$	$1.483779E+00$
	NGHS	$2.933452E-02$	$5.495792E-02$	$1.250118E-01$	$2.413133E-02$	$5.503947E-01$
	HSTL	**$5.196982E-05$**	**$1.156867E-04$**	**$1.822736E-04$**	**$3.949670E-05$**	$1.105805E+00$
F_{11}	HS	**$-8.495269E+03$**	$2.112918E+05$	$6.120867E+05$	$1.507484E+05$	**$5.529535E-01$**
	SGHS	$1.495650E+05$	$2.926573E+05$	$4.921806E+05$	$8.867408E+04$	$1.522435E+00$
	NGHS	$7.811770E+02$	$6.724066E+04$	$1.795737E+05$	$4.051646E+04$	$5.825563E-01$
	HSTL	$3.586693E+03$	**$3.982094E+04$**	**$1.129715E+05$**	**$3.077074E+04$**	$1.154489E+00$
F_{12}	HS	$1.808763E+02$	$2.922832E+02$	$3.779931E+02$	$5.595106E+01$	**$5.278572E-01$**
	SGHS	$1.966065E+02$	$2.939376E+02$	$4.245442E+02$	$6.167682E+01$	$1.413886E+00$
	NGHS	**$2.263637E+01$**	**$5.379553E+01$**	**$9.021302E+01$**	**$1.550666E+01$**	$5.732848E-01$
	HSTL	$1.598764E+02$	$2.741391E+02$	$3.460420E+02$	$5.654623E+01$	$1.075394E+00$
F_{13}	HS	$-3.982491E+02$	$-3.703682E+02$	$-3.359549E+02$	$1.639155E+01$	**$1.281255E+00$**
	SGHS	$-4.342564E+02$	$-4.020580E+02$	$-3.757443E+02$	$1.465006E+01$	$2.355018E+00$
	NGHS	$-4.499802E+02$	$-4.499468E+02$	$-4.498976E+02$	$2.171095E-02$	$1.325362E+00$
	HSTL	**$-4.499992E+02$**	**$-4.499987E+02$**	**$-4.499983E+02$**	**$2.372395E-04$**	$2.063549E+00$
F_{14}	HS	$-4.330907E+02$	$-4.272501E+02$	$-4.198265E+02$	$3.339807E+00$	**$1.293516E+00$**
	SGHS	$-4.290403E+02$	$-4.161920E+02$	$-4.061673E+02$	$6.869724E+00$	$2.335600E+00$
	NGHS	**$-4.475734E+02$**	**$-4.467607E+02$**	**$-4.456620E+02$**	**$5.729226E-01$**	$1.334976E+00$
	HSTL	$-4.345355E+02$	$-4.303615E+02$	$-4.257539E+02$	$2.813241E+00$	$2.110056E+00$
F_{15}	HS	$4.518000E+04$	$1.178572E+05$	$2.758509E+05$	$5.318182E+04$	**$1.548986E+00$**
	SGHS	$1.546908E+04$	$1.439934E+05$	$1.140938E+06$	$2.428257E+05$	$2.663552E+00$
	NGHS	$5.520145E+02$	$1.096629E+03$	**$5.304441E+03$**	$1.408912E+03$	$1.595152E+00$
	HSTL	**$4.364245E+02$**	**$8.663389E+02$**	$6.714816E+03$	**$1.380399E+03$**	$2.384685E+00$
F_{16}	HS	$-3.073606E+02$	$-2.979182E+02$	$-2.930230E+02$	$3.428569E+00$	**$1.388468E+00$**
	SGHS	$-2.904807E+02$	$-2.795281E+02$	$-2.653651E+02$	$7.917957E+00$	$2.457957E+00$
	NGHS	$-3.247746E+02$	**$-3.221057E+02$**	**$-3.177697E+02$**	**$2.003994E+00$**	$1.425334E+00$
	HSTL	**$-3.257734E+02$**	**$-3.194510E+02$**	$-3.131576E+02$	$3.257068E+00$	$2.154368E+00$
F_{17}	HS	$-1.784876E+02$	$-1.782673E+02$	$-1.779930E+02$	$1.484480E-01$	**$1.920537E+00$**
	SGHS	$-1.788319E+02$	$-1.785108E+02$	$-1.778547E+02$	$2.823212E-01$	$3.026930E+00$
	NGHS	$-1.799673E+02$	$-1.799278E+02$	$-1.798487E+02$	$2.657279E-02$	$1.936843E+00$
	HSTL	**$-1.799993E+02$**	**$-1.799977E+02$**	**$-1.799891E+02$**	**$2.941948E-03$**	$2.661228E+00$
F_{18}	HS	$-1.373170E+02$	$-1.367418E+02$	$-1.362105E+02$	$3.320310E-01$	**$1.538989E+00$**
	SGHS	$-1.380207E+02$	$-1.373137E+02$	$-1.363707E+02$	$3.842756E-01$	$2.631186E+00$
	NGHS	$-1.399544E+02$	$-1.399264E+02$	$-1.399026E+02$	$1.478740E-02$	$1.556766E+00$
	HSTL	**$-1.399944E+02$**	**$-1.399932E+02$**	**$-1.399923E+02$**	**$4.802592E-04$**	$2.308382E+00$

Table 5: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{19}	HS	$-7.783258E+02$	$-7.574243E+02$	$-7.440193E+02$	$9.549819E+00$	**3.364700E + 00**
	SGHS	$-7.977925E+02$	$-7.830542E+02$	$-7.648281E+02$	$8.315008E+00$	$4.449306E+00$
	NGHS	$-8.126267E+02$	$-7.983609E+02$	$-7.807750E+02$	$7.476016E+00$	$3.453860E+00$
	HSTL	**−8.133893E + 02**	**−8.054826E + 02**	**−7.933937E + 02**	**5.255994E + 00**	$4.110440E+00$
F_{20}	HS	$3.235346E+01$	$4.619696E+01$	$6.603604E+01$	$9.209638E+00$	**6.108355E − 01**
	SGHS	$1.840610E+01$	$2.931223E+01$	$4.219485E+01$	$6.705792E+00$	$1.656616E+00$
	NGHS	$1.023646E+00$	$1.326115E+00$	$1.638720E+00$	$1.801950E-01$	$6.740073E-01$
	HSTL	**1.666479E − 02**	**2.377405E − 02**	**3.561417E − 02**	**3.989928E − 03**	$1.340302E+00$
F_{21}	HS	$1.039417E+02$	$1.371442E+02$	$1.606956E+02$	$1.457780E+01$	$1.692515E+01$
	SGHS	$6.337241E+01$	$8.847938E+01$	$1.032481E+02$	$9.606610E+00$	$1.883071E+01$
	NGHS	$2.804776E+01$	$3.706996E+01$	$4.978956E+01$	$6.433861E+00$	**1.691865E + 01**
	HSTL	**4.874162E + 00**	**5.839427E + 00**	**7.688011E + 00**	**8.479931E − 01**	$1.766882E+01$
F_{22}	HS	$2.466530E+01$	$3.127048E+01$	$3.564485E+01$	$3.212730E+00$	**1.153501E + 01**
	SGHS	$1.342097E+01$	$2.030578E+01$	$2.459678E+01$	$3.326658E+00$	$1.349578E+01$
	NGHS	$4.017501E-02$	$7.928514E-02$	**1.477305E − 01**	**3.097912E − 02**	$1.160670E+01$
	HSTL	**8.611143E − 04**	**2.201951E − 02**	$4.140857E-01$	$9.228328E-02$	$1.214935E+01$
F_{23}	HS	$1.167773E+02$	$1.315135E+02$	$1.534918E+02$	$1.135387E+01$	**1.264720E + 01**
	SGHS	$7.270431E+01$	$8.324425E+01$	$1.008921E+02$	$6.879689E+00$	$1.442205E+01$
	NGHS	$2.477052E+01$	$3.758657E+01$	$5.407804E+01$	$7.976231E+00$	$1.312881E+01$
	HSTL	**4.038784E + 00**	**6.052038E + 00**	**1.261065E + 01**	**1.761817E + 00**	$1.385192E+01$
F_{24}	HS	$-4.217519E+02$	$-4.046735E+02$	$-3.782667E+02$	$1.257151E+01$	**1.359617E + 01**
	SGHS	$-4.431356E+02$	$-4.312111E+02$	$-3.858681E+02$	$1.369453E+01$	$1.555383E+01$
	NGHS	**−4.499996E + 02**	$-4.499993E+02$	$-4.499982E+02$	$3.724625E-04$	$1.364959E+01$
	HSTL	**−4.499996E + 02**	**−4.499994E + 02**	**−4.499991E + 02**	**1.730785E − 04**	$1.440449E+01$
F_{25}	HS	$2.665579E+04$	$6.003134E+04$	$1.020993E+05$	$2.295715E+04$	**1.384424E + 01**
	SGHS	$6.783432E+03$	$3.423766E+04$	$1.447178E+05$	$3.728901E+04$	$1.574535E+01$
	NGHS	$4.838503E+02$	$5.491368E+02$	**6.308092E + 02**	**4.002266E + 01**	$1.387828E+01$
	HSTL	**4.283814E + 02**	**5.425790E + 02**	$1.213942E+03$	$1.982288E+02$	$1.433272E+01$
F_{26}	HS	$-3.056734E+02$	$-3.005925E+02$	$-2.916669E+02$	$3.782453E+00$	**1.339199E + 01**
	SGHS	$-2.924154E+02$	$-2.838442E+02$	$-2.713927E+02$	$6.793276E+00$	$1.527401E+01$
	NGHS	$-3.267093E+02$	$-3.228070E+02$	$-3.183256E+02$	**2.244713E + 00**	$1.341534E+01$
	HSTL	**−3.289412E + 02**	**−3.236531E + 02**	**−3.188403E + 02**	$2.872375E+00$	$1.421595E+01$
F_{27}	HS	$4.849375E+00$	$6.498510E+00$	$8.495851E+00$	$1.017495E+00$	**1.262153E + 01**
	SGHS	$1.619215E+00$	$4.090404E+00$	$6.379225E+00$	$1.193628E+00$	$1.455011E+01$
	NGHS	$6.034321E-02$	$8.531001E-02$	$1.167611E-01$	$1.717796E-02$	$1.266599E+01$
	HSTL	**8.329468E − 03**	**9.918090E − 03**	**1.162367E − 02**	**1.094430E − 03**	$1.344353E+01$

Table 5: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{28}	HS	$-4.495086E+02$	$-4.478733E+02$	$-4.443561E+02$	$1.460489E+00$	**1.617566E + 01**
	SGHS	$-4.499731E+02$	$-4.498848E+02$	$-4.497073E+02$	$8.308225E-02$	$1.816665E+01$
	NGHS	**−4.500000E + 02**	**−4.500000E + 02**	**−4.500000E + 02**	**5.050670E − 14**	$1.633242E+01$
	HSTL	**−4.500000E + 02**	**−4.500000E + 02**	**−4.500000E + 02**	$3.894074E-06$	$1.700242E+01$
F_{29}	HS	$7.984189E+02$	$5.231460E+03$	$2.285396E+04$	$5.423033E+03$	**1.646778E + 01**
	SGHS	$6.857379E+02$	$2.714991E+03$	$1.007518E+04$	$2.839634E+03$	$1.841738E+01$
	NGHS	$4.290961E+02$	**5.562601E + 02**	**1.377085E + 03**	**2.419459E + 02**	$1.656795E+01$
	HSTL	**4.073724E + 02**	$1.159148E+03$	$9.324027E+03$	$2.181373E+03$	$1.728557E+01$
F_{30}	HS	$-3.077579E+02$	$-3.023452E+02$	$-2.973287E+02$	$3.007979E+00$	**1.629385E + 01**
	SGHS	$-3.057815E+02$	$-2.971863E+02$	$-2.874010E+02$	$4.692703E+00$	$1.819898E+01$
	NGHS	$-3.260430E+02$	$-3.234723E+02$	$-3.198728E+02$	$1.579631E+00$	$1.644573E+01$
	HSTL	**−3.295820E + 02**	**−3.280941E + 02**	**−3.260903E + 02**	**1.159512E + 00**	$1.714851E+01$
F_{31}	HS	$1.196768E+01$	$1.679724E+01$	$2.010432E+01$	$2.439712E+00$	**1.279175E + 01**
	SGHS	$4.187297E+00$	$9.601911E+00$	$1.575427E+01$	$3.206880E+00$	$1.473686E+01$
	NGHS	$2.868378E-02$	$4.841433E-02$	$9.156237E-02$	$1.566071E-02$	$1.283577E+01$
	HSTL	**2.103222E − 03**	**2.808338E − 03**	**3.640376E − 03**	**4.063424E − 04**	$1.358129E+01$

Table 6: The optimization results of four harmony search algorithms for F_1–F_{31} ($D = 100$, FEs = $500D$).

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_1	HS	3.472069E + 00	3.820833E + 00	4.047963E + 00	1.596325E − 01	1.769219E + 00
	SGHS	3.132910E + 00	3.913786E + 00	5.083229E + 00	5.509827E − 01	4.010046E + 00
	NGHS	1.444919E + 00	1.767520E + 00	2.084939E + 00	1.790358E − 01	**1.750870E + 00**
	HSTL	**2.287774E − 03**	**2.786403E − 03**	**3.224305E − 03**	**2.716016E − 04**	3.008579E + 00
F_2	HS	3.167536E + 02	6.351839E + 02	1.255875E + 03	2.510796E + 02	**1.193827E + 00**
	SGHS	3.589095E + 02	1.471320E + 03	3.982050E + 03	1.068029E + 03	3.400621E + 00
	NGHS	3.974602E + 01	6.225108E + 01	8.429758E + 01	1.057944E + 01	1.243040E + 00
	HSTL	**6.668194E − 01**	**1.286474E + 00**	**7.749294E + 00**	**1.726882E + 00**	2.346918E + 00
F_3	HS	2.603003E + 00	3.327735E + 00	3.759840E + 00	2.519202E − 01	1.920741E + 00
	SGHS	2.174329E + 00	4.132405E + 00	6.281051E + 00	1.173173E + 00	4.125897E + 00
	NGHS	1.188965E + 00	1.242508E + 00	1.340110E + 00	3.865128E − 02	**1.920139E + 00**
	HSTL	**1.376879E − 04**	**2.346120E − 04**	**5.029240E − 04**	**7.730570E − 05**	3.209074E + 00
F_4	HS	8.948965E − 01	1.142211E + 00	1.519557E + 00	1.440158E − 01	5.019303E + 00
	SGHS	5.690498E + 00	1.420662E + 01	2.458487E + 01	5.224712E + 00	7.419859E + 00
	NGHS	9.494633E − 02	1.460202E − 01	2.463073E − 01	3.355741E − 02	**5.010614E + 00**
	HSTL	**1.635147E − 05**	**1.380600E − 04**	**2.370741E − 03**	**5.255262E − 04**	6.668101E + 00
F_5	HS	−6.672187E + 01	−6.350148E + 01	−6.043354E + 01	2.052348E + 00	2.969232E + 00
	SGHS	−8.968012E + 01	−8.648817E + 01	−8.264598E + 01	1.813756E + 00	5.086223E + 00
	NGHS	−8.313566E + 01	−8.029515E + 01	−7.690579E + 01	1.438337E + 00	**2.968101E + 00**
	HSTL	**−9.585843E + 01**	**−9.464151E + 01**	**−9.352011E + 01**	**5.668992E − 01**	4.289409E + 00
F_6	HS	2.275081E + 01	3.404998E + 01	4.701336E + 01	7.463806E + 00	4.601501E + 00
	SGHS	6.202169E + 01	1.021203E + 02	1.576931E + 02	2.604590E + 01	7.029177E + 00
	NGHS	2.559237E + 00	4.248504E + 00	7.194444E + 00	1.263185E + 00	**4.571996E + 00**
	HSTL	**1.159057E − 01**	**2.164533E − 01**	**3.013113E − 01**	**4.862855E − 02**	6.357763E + 00
F_7	HS	8.761768E + 01	9.968145E + 01	1.095680E + 02	5.125492E + 00	**1.397587E + 00**
	SGHS	1.204523E + 02	1.515871E + 02	1.806614E + 02	1.516934E + 01	3.643591E + 00
	NGHS	5.302534E + 01	7.188578E + 01	8.426157E + 01	7.828465E + 00	1.433040E + 00
	HSTL	**1.577470E − 04**	**9.978969E − 02**	**9.954242E − 01**	**3.062646E − 01**	2.611144E + 00
F_8	HS	5.880175E + 02	1.300531E + 03	2.082232E + 03	3.831144E + 02	**1.223890E + 00**
	SGHS	8.999245E + 02	1.790814E + 03	2.911281E + 03	5.768125E + 02	3.494810E + 00
	NGHS	3.079063E + 02	4.138027E + 02	5.638547E + 02	7.193641E + 01	1.255683E + 00
	HSTL	**9.318870E + 01**	**1.469698E + 02**	**2.571664E + 02**	**4.541752E + 01**	2.443950E + 00
F_9	HS	7.122610E + 02	8.396009E + 02	9.466086E + 02	**6.566372E + 01**	**1.529895E + 00**
	SGHS	3.806886E + 03	4.736018E + 03	6.005643E + 03	6.494470E + 02	3.877912E + 00
	NGHS	6.828367E + 02	1.783404E + 03	2.842955E + 03	4.795603E + 02	1.569362E + 00
	HSTL	**7.968585E + 00**	**1.720210E + 02**	**3.973356E + 02**	1.098869E + 02	2.854428E + 00

Table 6: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{10}	HS	2.129878E + 02	2.699577E + 02	3.362863E + 02	3.021370E + 01	**1.482740E + 00**
	SGHS	1.475702E + 02	3.215163E + 02	5.256382E + 02	1.308856E + 02	3.719739E + 00
	NGHS	1.599879E + 01	2.545197E + 01	3.908629E + 01	5.800758E + 00	1.488095E + 00
	HSTL	**2.404241E − 04**	**3.881772E − 04**	**5.946775E − 04**	**9.559413E − 05**	2.861329E + 00
F_{11}	HS	3.939318E + 06	7.748172E + 06	1.544812E + 07	2.861829E + 06	**1.567541E + 00**
	SGHS	1.156049E + 07	2.205330E + 07	3.666560E + 07	7.051009E + 06	3.787745E + 00
	NGHS	3.799257E + 06	6.908533E + 06	9.606336E + 06	1.677867E + 06	1.569956E + 00
	HSTL	**3.921902E + 05**	**1.520828E + 06**	**3.598060E + 06**	**9.375356E + 05**	2.978233E + 00
F_{12}	HS	7.150539E + 02	9.583770E + 02	1.225170E + 03	1.259891E + 02	**1.530910E + 00**
	SGHS	7.715070E + 02	9.708888E + 02	1.242580E + 03	1.201418E + 02	3.590301E + 00
	NGHS	**2.108479E + 02**	**2.650550E + 02**	**3.204091E + 02**	**3.392485E + 01**	1.553830E + 00
	HSTL	6.766070E + 02	8.735240E + 02	1.063928E + 03	1.027158E + 02	2.874153E + 00
F_{13}	HS	−2.375559E + 02	−1.738990E + 02	−1.079270E + 02	3.473333E + 01	**3.058755E + 00**
	SGHS	−3.131437E + 02	−2.097408E + 02	−8.276031E + 01	7.569102E + 01	5.492338E + 00
	NGHS	−4.332349E + 02	−4.262997E + 02	−4.192761E + 02	3.856018E + 00	3.064137E + 00
	HSTL	**−4.499970E + 02**	**−4.499959E + 02**	**−4.499951E + 02**	**5.433935E − 04**	4.894014E + 00
F_{14}	HS	−4.232215E + 02	−4.193171E + 02	−4.108272E + 02	2.989275E + 00	**3.133809E + 00**
	SGHS	−3.977762E + 02	−3.880149E + 02	−3.766792E + 02	5.980155E + 00	5.474201E + 00
	NGHS	**−4.402313E + 02**	**−4.377563E + 02**	**−4.360122E + 02**	**1.032261E + 00**	3.189262E + 00
	HSTL	−4.245886E + 02	−4.181521E + 02	−4.133119E + 02	3.501804E + 00	4.858567E + 00
F_{15}	HS	3.297507E + 05	5.050010E + 05	7.154538E + 05	1.134203E + 05	**3.715503E + 00**
	SGHS	2.141247E + 05	2.473414E + 06	9.215806E + 06	2.312600E + 06	6.170230E + 00
	NGHS	6.173097E + 03	1.336579E + 04	2.527368E + 04	6.553278E + 03	3.722490E + 00
	HSTL	**6.679117E + 02**	**2.123307E + 03**	**6.663690E + 03**	**1.898390E + 03**	5.447209E + 00
F_{16}	HS	−2.417986E + 02	−2.316766E + 02	−2.243841E + 02	4.720329E + 00	3.407872E + 00
	SGHS	−1.978875E + 02	−1.657988E + 02	−1.304433E + 02	1.954773E + 01	5.825132E + 00
	NGHS	−2.700495E + 02	−2.575340E + 02	−2.441387E + 02	7.491290E + 00	**3.388501E + 00**
	HSTL	**−3.134902E + 02**	**−3.059890E + 02**	**−2.968384E + 02**	**4.108868E + 00**	5.092343E + 00
F_{17}	HS	−1.769331E + 02	−1.764810E + 02	−1.760164E + 02	2.480479E − 01	5.401888E + 00
	SGHS	−1.779920E + 02	−1.770352E + 02	−1.750318E + 02	8.297157E − 01	7.875262E + 00
	NGHS	−1.788120E + 02	−1.787740E + 02	−1.787241E + 02	2.570359E − 02	**5.350236E + 00**
	HSTL	**−1.799984E + 02**	**−1.799963E + 02**	**−1.799876E + 02**	**3.066164E − 03**	7.034552E + 00
F_{18}	HS	−1.365537E + 02	−1.361589E + 02	−1.357947E + 02	2.159838E − 01	3.742110E + 00
	SGHS	−1.369490E + 02	−1.362044E + 02	−1.344587E + 02	6.516864E − 01	6.201829E + 00
	NGHS	−1.385604E + 02	−1.381231E + 02	−1.377560E + 02	2.656460E − 01	**3.702051E + 00**
	HSTL	**−1.399928E + 02**	**−1.399894E + 02**	**−1.399455E + 02**	**1.034768E − 02**	5.419278E + 00

Table 6: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{19}	HS	$-1.397521E+03$	$-1.360864E+03$	$-1.328214E+03$	$2.038863E+01$	**$1.091921E+01$**
	SGHS	$-1.452010E+03$	$-1.426324E+03$	$-1.397075E+03$	$1.631505E+01$	$1.335878E+01$
	NGHS	$-1.451444E+03$	$-1.428229E+03$	$-1.393381E+03$	$1.340884E+01$	$1.124146E+01$
	HSTL	**$-1.513615E+03$**	**$-1.499926E+03$**	**$-1.485147E+03$**	**$8.517545E+00$**	$1.261544E+01$
F_{20}	HS	$1.193796E+02$	$1.193796E+02$	$1.193796E+02$	$1.193796E+02$	$1.193796E+02$
	SGHS	$9.824839E+01$	$9.824839E+01$	$9.824839E+01$	$9.824839E+01$	$9.824839E+01$
	NGHS	$3.473853E+01$	$3.473853E+01$	$3.473853E+01$	$3.473853E+01$	$3.473853E+01$
	HSTL	**$6.151721E-02$**	**$6.151721E-02$**	**$6.151721E-02$**	**$6.151721E-02$**	**$6.151721E-02$**
F_{21}	HS	$2.731607E+02$	$3.018028E+02$	$3.309987E+02$	$1.465698E+01$	$5.400367E+01$
	SGHS	$2.061790E+02$	$2.668907E+02$	$3.219105E+02$	$3.133660E+01$	$5.782494E+01$
	NGHS	$1.520321E+02$	$1.848391E+02$	$2.324962E+02$	$1.945112E+01$	**$5.352526E+01$**
	HSTL	**$1.216583E+01$**	**$1.522370E+01$**	**$1.868875E+01$**	**$2.164395E+00$**	$5.544992E+01$
F_{22}	HS	$7.035058E+01$	$7.966205E+01$	$8.675826E+01$	$4.501205E+00$	$3.134446E+01$
	SGHS	$4.845036E+01$	$7.002079E+01$	$1.031254E+02$	$1.472650E+01$	$3.551839E+01$
	NGHS	$1.890592E+01$	$2.264589E+01$	$2.708944E+01$	$2.757052E+00$	**$3.129377E+01$**
	HSTL	**$3.312806E-03$**	**$4.788330E-01$**	**$2.104085E+00$**	**$7.191299E-01$**	$3.285056E+01$
F_{23}	HS	$2.474923E+02$	$2.926433E+02$	$3.154565E+02$	$1.633700E+01$	$3.787547E+01$
	SGHS	$2.077105E+02$	$2.618490E+02$	$3.486951E+02$	$3.410635E+01$	$4.192231E+01$
	NGHS	$1.600018E+02$	$1.868942E+02$	$2.193507E+02$	$1.802714E+01$	**$3.742503E+01$**
	HSTL	**$1.226995E+01$**	**$1.528464E+01$**	**$2.013004E+01$**	**$2.366879E+00$**	$3.842275E+01$
F_{24}	HS	$-3.539791E+02$	$-3.147048E+02$	$-2.647908E+02$	$1.949672E+01$	**$3.449387E+01$**
	SGHS	$-4.095286E+02$	$-3.591410E+02$	$-2.333805E+02$	$5.324971E+01$	$3.871345E+01$
	NGHS	$-4.493291E+02$	$-4.489227E+02$	$-4.485923E+02$	$2.120875E-01$	$3.486910E+01$
	HSTL	**$-4.499990E+02$**	**$-4.499985E+02$**	**$-4.499980E+02$**	**$2.414762E-04$**	$3.688151E+01$
F_{25}	HS	$1.051141E+05$	$1.874186E+05$	$2.937699E+05$	$4.432322E+04$	**$3.544980E+01$**
	SGHS	$5.004828E+04$	$6.949195E+05$	$3.958335E+06$	$9.272106E+05$	$3.953879E+01$
	NGHS	$1.111344E+03$	$1.579429E+03$	$4.609411E+03$	$7.698066E+02$	$3.568522E+01$
	HSTL	**$4.768735E+02$**	**$8.024700E+02$**	**$2.168501E+03$**	**$4.225175E+02$**	$3.736183E+01$
F_{26}	HS	$-2.598286E+02$	$-2.502187E+02$	$-2.401834E+02$	$5.848484E+00$	$3.371420E+01$
	SGHS	$-2.231800E+02$	$-1.912732E+02$	$-1.659675E+02$	$1.303039E+01$	$3.722651E+01$
	NGHS	$-2.870567E+02$	$-2.734730E+02$	$-2.629992E+02$	$5.510929E+00$	**$3.296890E+01$**
	HSTL	**$-3.207702E+02$**	**$-3.136777E+02$**	**$-3.092977E+02$**	**$2.615418E+00$**	$3.492102E+01$
F_{27}	HS	$1.761478E+01$	$1.978397E+01$	$2.313474E+01$	$1.461489E+00$	**$3.122492E+01$**
	SGHS	$1.215437E+01$	$2.186805E+01$	$3.511913E+01$	$5.628417E+00$	$3.560859E+01$
	NGHS	$2.810761E+00$	$3.983730E+00$	$4.655769E+00$	$4.301760E-01$	$3.140541E+01$
	HSTL	**$2.089110E-02$**	**$2.831916E-02$**	**$5.808363E-02$**	**$8.512059E-03$**	$3.310728E+01$

Table 6: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{28}	HS	$-4.469079E+02$	$-4.443488E+02$	$-4.405620E+02$	$1.967662E+00$	**$4.318177E+01$**
	SGHS	$-4.494424E+02$	$-4.483361E+02$	$-4.463611E+02$	$1.058420E+00$	$4.732766E+01$
	NGHS	**$-4.500000E+02$**	**$-4.500000E+02$**	**$-4.500000E+02$**	**$3.611281E-10$**	$4.326785E+01$
	HSTL	**$-4.500000E+02$**	**$-4.500000E+02$**	**$-4.500000E+02$**	$9.511311E-06$	$4.509948E+01$
F_{29}	HS	$6.093107E+03$	$1.397065E+04$	$2.184820E+04$	$1.114053E+04$	$4.387041E+01$
	SGHS	$8.168591E+03$	$9.147798E+03$	$1.012700E+04$	$1.384807E+03$	$4.815433E+01$
	NGHS	$6.891292E+02$	$7.565447E+02$	$8.239602E+02$	$9.533993E+01$	**$4.375258E+01$**
	HSTL	**$4.954684E+02$**	**$5.024741E+02$**	**$5.094799E+02$**	**$9.907604E+00$**	$4.518065E+01$
F_{30}	HS	$-2.708213E+02$	$-2.658142E+02$	$-2.553643E+02$	$4.447667E+00$	**$4.291836E+01$**
	SGHS	$-2.498452E+02$	$-2.356727E+02$	$-2.214623E+02$	$6.894224E+00$	$4.714809E+01$
	NGHS	$-2.967689E+02$	$-2.895710E+02$	$-2.794822E+02$	$4.509198E+00$	$4.315103E+01$
	HSTL	**$-3.281517E+02$**	**$-3.254159E+02$**	**$-3.216770E+02$**	**$1.727024E+00$**	$4.476932E+01$
F_{31}	HS	$3.744918E+01$	$4.335719E+01$	$4.893459E+01$	$3.361402E+00$	$3.044300E+01$
	SGHS	$2.842099E+01$	$4.402938E+01$	$6.465355E+01$	$8.002441E+00$	$3.374892E+01$
	NGHS	$5.436447E+00$	$8.517679E+00$	$1.110289E+01$	$1.334154E+00$	**$2.980321E+01$**
	HSTL	**$7.377615E-03$**	**$8.118152E-02$**	**$4.421143E-01$**	**$1.112750E-01$**	$3.135758E+01$

Table 7: The optimization results of four harmony search algorithms for F_1–F_{31} ($D = 200$, FEs = $500D$).

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_1	HS	$4.603726E+00$	$4.929354E+00$	$5.266226E+00$	$1.613707E-01$	**$5.715312E+00$**
	SGHS	$4.421737E+00$	$5.637606E+00$	$7.262746E+00$	$8.839935E-01$	$1.138709E+01$
	NGHS	$5.145771E+00$	$5.521262E+00$	$5.794585E+00$	$1.715834E-01$	$5.721544E+00$
	HSTL	**$3.575306E-03$**	**$4.026710E-03$**	**$4.538061E-03$**	**$2.396758E-04$**	$8.701444E+00$
F_2	HS	$4.078042E+03$	$6.153632E+03$	$8.488061E+03$	$1.312734E+03$	$4.147045E+00$
	SGHS	$7.148166E+03$	$3.459359E+04$	$1.047654E+05$	$2.687689E+04$	$9.907156E+00$
	NGHS	$2.696362E+03$	$4.466500E+03$	$5.439083E+03$	$6.579299E+02$	**$4.042910E+00$**
	HSTL	**$6.688204E-01$**	**$9.705961E+00$**	**$5.070641E+01$**	**$1.456647E+01$**	$7.049820E+00$
F_3	HS	$9.161714E+00$	$1.244639E+01$	$1.459380E+01$	$1.320806E+00$	$6.456437E+00$
	SGHS	$8.285825E+00$	$1.792049E+01$	$5.102385E+01$	$9.342384E+00$	$1.205154E+01$
	NGHS	$1.653665E+01$	$1.945936E+01$	$2.288375E+01$	$1.916412E+00$	**$6.373141E+00$**
	HSTL	**$3.399743E-04$**	**$1.932473E-03$**	**$2.008061E-02$**	**$4.825732E-03$**	$9.435844E+00$
F_4	HS	$3.848564E+00$	$4.708132E+00$	$5.294488E+00$	$3.531660E-01$	$1.828390E+01$
	SGHS	$4.320375E+01$	$6.698858E+01$	$9.905690E+01$	$1.803405E+01$	$2.448338E+01$
	NGHS	$7.186447E+00$	$8.321538E+00$	$9.657602E+00$	$6.923441E-01$	**$1.812159E+01$**
	HSTL	**$3.729585E-02$**	**$1.541456E-01$**	**$2.844556E-01$**	**$7.373455E-02$**	$2.201131E+01$
F_5	HS	$-9.228155E+01$	$-8.912511E+01$	$-8.545813E+01$	$2.196104E+00$	$1.075750E+01$
	SGHS	$-1.691553E+02$	$-1.632416E+02$	$-1.554168E+02$	$4.230945E+00$	$1.607256E+01$
	NGHS	$-1.266981E+02$	$-1.212701E+02$	$-1.165799E+02$	$2.487909E+00$	**$1.057139E+01$**
	HSTL	**$-1.879057E+02$**	**$-1.859165E+02$**	**$-1.841057E+02$**	**$9.032725E-01$**	$1.379854E+01$
F_6	HS	$1.059685E+02$	$1.605423E+02$	$2.171595E+02$	$3.043869E+01$	$1.913746E+01$
	SGHS	$2.877414E+02$	$5.597265E+02$	$1.076033E+03$	$1.825775E+02$	$2.544270E+01$
	NGHS	$1.476215E+02$	$1.815020E+02$	$2.722819E+02$	$2.829572E+01$	**$1.891687E+01$**
	HSTL	**$4.134323E-01$**	**$8.303796E-01$**	**$1.245398E+00$**	**$1.952532E-01$**	$2.286965E+01$
F_7	HS	$3.049237E+02$	$3.272066E+02$	$3.630817E+02$	$1.403316E+01$	$5.018255E+00$
	SGHS	$3.796419E+02$	$4.822636E+02$	$5.650928E+02$	$4.455035E+01$	$1.075236E+01$
	NGHS	$4.522705E+02$	$5.209768E+02$	$5.738802E+02$	$3.656850E+01$	**$4.953818E+00$**
	HSTL	**$1.658281E-03$**	**$1.295575E+00$**	**$3.005119E+00$**	**$1.058554E+00$**	$7.836046E+00$
F_8	HS	$3.896266E+03$	$5.289774E+03$	$7.844040E+03$	$1.015647E+03$	$4.297905E+00$
	SGHS	$6.818663E+03$	$1.570204E+04$	$2.975647E+04$	$8.010501E+03$	$1.001014E+01$
	NGHS	$3.307225E+03$	$4.050635E+03$	$5.045072E+03$	$4.439356E+02$	**$4.207027E+00$**
	HSTL	**$1.929455E+02$**	**$2.304209E+02$**	**$3.286657E+02$**	**$4.180170E+01$**	$7.172235E+00$
F_9	HS	$3.233837E+03$	$3.714921E+03$	$4.247033E+03$	**$2.604085E+02$**	$5.199977E+00$
	SGHS	$1.175294E+04$	$1.460607E+04$	$1.810298E+04$	$1.831608E+03$	$1.119387E+01$
	NGHS	$1.228090E+04$	$1.400627E+04$	$1.550115E+04$	$8.852628E+02$	**$5.119688E+00$**
	HSTL	**$1.550538E+03$**	**$1.951142E+03$**	**$2.716798E+03$**	$3.492572E+02$	$8.289662E+00$

Table 7: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{10}	HS	1.035974E + 03	1.224840E + 03	1.408721E + 03	9.946126E + 01	4.635011E + 00
	SGHS	9.842122E + 02	1.758492E + 03	3.754911E + 03	7.377144E + 02	1.040168E + 01
	NGHS	1.861439E + 03	2.058453E + 03	2.504462E + 03	1.837061E + 02	**4.467012E + 00**
	HSTL	**1.314733E − 03**	**1.760664E − 03**	**2.393748E − 03**	**2.837635E − 04**	7.936856E + 00
F_{11}	HS	3.073770E + 08	5.529318E + 08	7.578658E + 08	1.318618E + 08	5.023786E + 00
	SGHS	8.707673E + 08	1.161818E + 09	1.767454E + 09	2.629050E + 08	1.075328E + 01
	NGHS	7.093037E + 08	9.105403E + 08	1.190901E + 09	1.208984E + 08	**4.824641E + 00**
	HSTL	**2.965758E + 07**	**8.970849E + 07**	**1.931661E + 08**	**3.740898E + 07**	8.290714E + 00
F_{12}	HS	2.178914E + 03	2.482882E + 03	2.815688E + 03	1.988376E + 02	4.845730E + 00
	SGHS	2.308245E + 03	2.584773E + 03	2.953531E + 03	2.051620E + 02	1.019302E + 01
	NGHS	**9.284727E + 02**	**1.072448E + 03**	**1.250569E + 03**	**9.258004E + 01**	**4.699592E + 00**
	HSTL	1.938323E + 03	2.308801E + 03	2.714633E + 03	1.905069E + 02	8.051031E + 00
F_{13}	HS	4.977364E + 02	8.339352E + 02	1.058941E + 03	1.248060E + 02	8.006508E + 00
	SGHS	2.137557E + 02	1.083748E + 03	2.437995E + 03	5.816088E + 02	1.404649E + 01
	NGHS	1.392416E + 03	1.708667E + 03	1.994529E + 03	1.697142E + 02	**7.789651E + 00**
	HSTL	**−3.915048E + 02**	**−3.394840E + 02**	**−2.719662E + 02**	**2.748671E + 01**	1.184497E + 01
F_{14}	HS	−4.084859E + 02	−4.048983E + 02	−3.994353E + 02	2.724485E + 00	8.170102E + 00
	SGHS	−3.756669E + 02	−3.637827E + 02	−3.512079E + 02	6.398155E + 00	1.406004E + 01
	NGHS	**−4.248494E + 02**	**−4.217489E + 02**	**−4.171032E + 02**	1.942003E + 00	**7.962233E + 00**
	HSTL	−4.078666E + 02	−4.041830E + 02	−4.006726E + 02	**1.886381E + 00**	1.207676E + 01
F_{15}	HS	2.261680E + 06	4.109099E + 06	5.954376E + 06	9.167839E + 05	9.762151E + 00
	SGHS	6.929055E + 06	5.330388E + 07	2.498381E + 08	5.898978E + 07	1.596695E + 01
	NGHS	5.823986E + 06	7.816998E + 06	1.131635E + 07	1.505165E + 06	**9.593909E + 00**
	HSTL	**2.775018E + 05**	**5.853541E + 05**	**9.776046E + 05**	**2.019841E + 05**	1.376301E + 01
F_{16}	HS	−2.948123E + 01	−5.791258E − 01	1.683795E + 01	1.195259E + 01	9.144801E + 00
	SGHS	7.067304E + 01	1.786652E + 02	2.626093E + 02	5.419678E + 01	1.508770E + 01
	NGHS	1.593170E + 02	1.986273E + 02	2.485464E + 02	2.374692E + 01	**8.921733E + 00**
	HSTL	**−2.547415E + 02**	**−2.353550E + 02**	**−2.151499E + 02**	**9.688531E + 00**	1.285946E + 01
F_{17}	HS	−1.685846E + 02	−1.671598E + 02	−1.654225E + 02	8.709843E − 01	1.678058E + 01
	SGHS	−1.751726E + 02	−1.626560E + 02	−1.461760E + 02	7.197573E + 00	2.278318E + 01
	NGHS	−1.634951E + 02	−1.604326E + 02	−1.574177E + 02	1.704898E + 00	**1.650759E + 01**
	HSTL	**−1.783495E + 02**	**−1.781095E + 02**	**−1.778417E + 02**	**1.456349E − 01**	2.036008E + 01
F_{18}	HS	−1.354532E + 02	−1.350248E + 02	−1.346804E + 02	1.775738E − 01	9.748667E + 00
	SGHS	−1.341637E + 02	−1.319425E + 02	−1.279421E + 02	1.937096E + 00	1.586162E + 01
	NGHS	−1.348232E + 02	−1.344521E + 02	−1.342897E + 02	1.505267E − 01	**9.520511E + 00**
	HSTL	**−1.385641E + 02**	**−1.382701E + 02**	**−1.380733E + 02**	**1.368319E − 01**	1.358972E + 01

Table 7: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{19}	HS	$-2.603380E+03$	$-2.531804E+03$	$-2.473751E+03$	$3.179450E+01$	$2.876496E+01$
	SGHS	$-2.721608E+03$	$-2.665352E+03$	$-2.578190E+03$	$3.671259E+01$	$3.458217E+01$
	NGHS	$-2.581654E+03$	$-2.492258E+03$	$-2.446278E+03$	$3.258052E+01$	**2.814525E + 01**
	HSTL	**−2.929585E + 03**	**−2.904859E + 03**	**−2.882272E + 03**	**1.460064E + 01**	$3.221904E+01$
F_{20}	HS	$9.189454E+02$	$5.077136E+27$	$1.015427E+29$	$2.270564E+28$	**5.038898E + 00**
	SGHS	$5.059004E+02$	$8.779452E+02$	$1.348477E+03$	$1.953273E+02$	$1.104361E+01$
	NGHS	$1.246908E+03$	$2.028307E+03$	$2.888645E+03$	$4.719852E+02$	$5.148672E+00$
	HSTL	**1.837030E − 01**	**2.280353E − 01**	**2.673117E − 01**	**2.164823E − 02**	$8.886787E+00$
F_{21}	HS	$6.377972E+02$	$6.978627E+02$	$7.582129E+02$	$3.386564E+01$	**1.646089E + 02**
	SGHS	$6.320206E+02$	$7.707349E+02$	$8.819430E+02$	$5.316649E+01$	$1.737214E+02$
	NGHS	$7.097658E+02$	$7.856465E+02$	$8.525136E+02$	$4.013482E+01$	$1.646784E+02$
	HSTL	**9.040644E + 01**	**9.986598E + 01**	**1.162748E + 02**	**6.666442E + 00**	$1.696227E+02$
F_{22}	HS	$2.143132E+02$	$2.280396E+02$	$2.618505E+02$	$1.143029E+01$	$7.711841E+01$
	SGHS	$1.576182E+02$	$2.361854E+02$	$4.033144E+02$	$5.873130E+01$	$8.653562E+01$
	NGHS	$2.229408E+02$	$2.658819E+02$	$2.931895E+02$	$1.367509E+01$	**7.686817E + 01**
	HSTL	**9.638521E + 00**	**1.807740E + 01**	**2.585962E + 01**	**3.506528E + 00**	$8.063761E+01$
F_{23}	HS	$6.234643E+02$	$6.947096E+02$	$7.419847E+02$	$3.350395E+01$	$1.019985E+02$
	SGHS	$6.526572E+02$	$7.693093E+02$	$8.338596E+02$	$4.956235E+01$	$1.112867E+02$
	NGHS	$6.898719E+02$	$7.661332E+02$	$8.566126E+02$	$4.173909E+01$	**1.018226E + 02**
	HSTL	**8.482421E + 01**	**9.938394E + 01**	**1.108417E + 02**	**8.042784E + 00**	$1.054352E+02$
F_{24}	HS	$4.553252E+01$	$2.076506E+02$	$4.542336E+02$	$1.063305E+02$	$8.318836E+01$
	SGHS	$-2.939057E+02$	$7.490397E+01$	$6.242859E+02$	$3.186816E+02$	$9.254373E+01$
	NGHS	$-2.373270E+02$	$-1.728535E+02$	$-9.543890E+01$	$4.073554E+01$	**8.306743E + 01**
	HSTL	**−4.499950E + 02**	**−4.499933E + 02**	**−4.499913E + 02**	**9.985585E − 04**	$8.722270E+01$
F_{25}	HS	$7.366636E+05$	$1.041736E+06$	$1.399154E+06$	$1.825535E+05$	$8.464179E+01$
	SGHS	$8.054915E+05$	$1.228918E+07$	$4.659116E+07$	$1.341661E+07$	$9.395835E+01$
	NGHS	$1.559065E+05$	$2.605101E+05$	$3.969066E+05$	$6.581579E+04$	**8.432677E + 01**
	HSTL	**1.448330E + 03**	**4.054597E + 03**	**9.585072E + 03**	**2.811645E + 03**	$8.853785E+01$
F_{26}	HS	$-8.620111E+01$	$-6.877671E+01$	$-4.175610E+01$	$1.198860E+01$	**8.370164E + 01**
	SGHS	$3.837488E+01$	$9.840901E+01$	$2.221608E+02$	$5.104657E+01$	$9.300928E+01$
	NGHS	$-1.214266E+01$	$7.757135E+00$	$3.280678E+01$	$1.408601E+01$	$8.382478E+01$
	HSTL	**−2.834421E + 02**	**−2.731218E + 02**	**−2.591557E + 02**	**6.993937E + 00**	$8.785385E+01$
F_{27}	HS	$6.017485E+01$	$6.526806E+01$	$7.041752E+01$	$2.646571E+00$	$8.031504E+01$
	SGHS	$8.524694E+01$	$1.156612E+02$	$1.717014E+02$	$2.490438E+01$	$8.958076E+01$
	NGHS	$5.648135E+01$	$6.439351E+01$	$7.123722E+01$	$4.693589E+00$	**7.798441E + 01**
	HSTL	**2.094078E + 00**	**2.833941E + 00**	**4.047095E + 00**	**5.585953E − 01**	$8.093007E+01$

Table 7: Continued.

Function	Algorithm	Best	Mean	Worst	STD	Runtime
F_{28}	HS	$-4.289935E+02$	$-4.211903E+02$	$-4.163227E+02$	$3.623053E+00$	**$1.249831E+02$**
	SGHS	$-4.488077E+02$	$-4.432399E+02$	$-4.317460E+02$	$4.446639E+00$	$1.353388E+02$
	NGHS	$-4.499999E+02$	$-4.499997E+02$	$-4.499995E+02$	$9.834457E-05$	$1.265239E+02$
	HSTL	**$-4.500000E+02$**	**$-4.499999E+02$**	**$-4.499999E+02$**	**$2.351848E-05$**	$1.325924E+02$
F_{29}	HS	$2.233712E+04$	$3.892963E+04$	$5.633712E+04$	$9.226538E+03$	**$1.300818E+02$**
	SGHS	$5.525834E+03$	$2.751029E+05$	$2.276536E+06$	$5.361823E+05$	$1.405805E+02$
	NGHS	$1.510576E+03$	$2.042038E+03$	$6.840093E+03$	**$1.160782E+03$**	$1.305521E+02$
	HSTL	**$4.888426E+02$**	**$1.232585E+03$**	**$5.838617E+03$**	$1.452099E+03$	$1.351203E+02$
F_{30}	HS	$-1.772284E+02$	$-1.610854E+02$	$-1.510294E+02$	$6.803769E+00$	**$1.289660E+02$**
	SGHS	$-9.536516E+01$	$-6.623147E+01$	$-3.230639E+01$	$1.639332E+01$	$1.391979E+02$
	NGHS	$-1.574155E+02$	$-1.463189E+02$	$-1.301902E+02$	$7.514495E+00$	$1.291356E+02$
	HSTL	**$-3.201694E+02$**	**$-3.156785E+02$**	**$-3.090753E+02$**	**$3.220219E+00$**	$1.337009E+02$
F_{31}	HS	$1.212613E+02$	$1.318822E+02$	$1.410318E+02$	$5.212928E+00$	**$8.330888E+01$**
	SGHS	$1.123687E+02$	$1.636391E+02$	$1.993816E+02$	$2.540267E+01$	$9.407166E+01$
	NGHS	$1.189049E+02$	$1.313797E+02$	$1.457549E+02$	$6.404969E+00$	$8.467554E+01$
	HSTL	**$4.403576E+00$**	**$7.020892E+00$**	**$1.210083E+01$**	**$1.807576E+00$**	$8.733987E+01$

The convergence of HSTL method is compared with three other HS algorithms: HS, SGHS, and NGHS on 50-D functions F_3, F_{10}, and F_{27}, where the HSTL algorithm demonstrates an evident superiority on efficiency and stability, which can be observed from the convergence graphs (Figures 2(a) and 4(a)) and boxplots (Figures 2(b) and 4(b)).

For 50-D problems, the convergence graphs and the boxplots are shown in Figures 2, 3, and 4 on 50-D for sphere unimodal function, griewank multimodal inseparable function, and hybrid function, Hybrid5, which is composed of Bohachevsky and Schwefel2.22 function, where Figures 2(b) and 4(b) plot the boxplots of best results in 30 independent runs. Figures 2(a) and 4(a) portray the convergence curves. It is evident from the convergence graphs that strongly uniform convergence can be maintained throughout the procedure of evolution. From the boxplots we can see that the HSTL algorithm has better convergence, stability, and robustness in most cases than HS, SGHS, and NGHS algorithms.

Figures 5 and 6 show the boxplots and convergence graphs of 3 different problems for dimensions equal to 100 and 200, respectively, where HSTL algorithm demonstrates obvious superiority on efficiency and stability.

4.4. Comparison of the Convergence Speed and Success Rate. In order to give a fair chance to 4 HS algorithm (HS, SGHS, NGHS, and HSTL) compared. We run each algorithm on each benchmark test function and stop as soon as the minimum error value acquired by the algorithm falls below the predefined threshold or a maximum number of FEs is exceeded. For all test functions, the algorithms carry out 30 independent runs.

The respective thresholds values of the error for 30 benchmark problems are in Table 11.

Table 8 displays the statistics results about the success rate, average runtime, and average FEs of F_1–F_{30}.

As Table 8 shows, for a high-dimensional problem (D = 100), the HS, SGHS, and NGHS have great difficulty in finding the global optima on all problems. The HSTL yields 100% success rate for F_1, F_2, F_4–F_{11}, F_{14}, F_{15}, F_{18}, F_{20}, F_{21}, F_{24}, F_{26}, F_{29}, F_{30}. The HSTL algorithm performs much better with success rates of 100% on most problems, and over 50% on F_3, F_{13}, F_{16}, F_{17}, F_{19}, F_{22}, and F_{28} where other algorithms fail in finding the global optima.

Table 9 illustrates the costing run of 4 HS algorithm for F_1–F_{31}. The runtime is equal to the average value of all functions mean runtime; the FEs is equal to the average value of all functions mean FEs; the Success Rate is equal to the mean value of the success rate of all functions. In Table 9, we intend to show how well the presented HSTL algorithm performs when compared to HS, SGHS and NGHS algorithm. From the statistics shown in Table 9 can be seen that HSTL uses the least runtime and FEs, and acquires the best success rate among these algorithms on over performances.

4.5. Convergence Analysis. To investigate the convergence of proposed HSTL algorithm, we record the population variance of each algorithm. As Figure 7 shows, for each type function, the fluctuation of population variance in HSTL is smaller than the fluctuation of population variance in SGHS and NGHS algorithm, and the population variance graphs fall steadily throughout the search process. As a consequence, the proposed HSTL algorithm has stronger robustness and convergence than other variants of HS.

4.6. Parameter HMS Study. In this section, the effect of HMS value on the performance of the HSTL method is investigated. The experimental results generated by using different HMS values (5, 10, 15, 20, 25, 30, 35, and 40) for dimensions equal to 50 are demonstrated in Table 10, respectively.

We can see from Table 10, in which there are some unimodal functions (i.e., F_{10}–F_{14}, F_{23}), a small value of HMS (i.e., 5 or 10) is superior to a large value. For other benchmark functions, there is no obvious indication that one setting value of HMS is superior to the other setting values. Therefore, we can think that, a small HMS is suitable for a simple problem. However, for some complex problems, we can set slightly more HMS that is not exceeding 50. It is reasonable and logical, for it is similar to the quickly memory of musician for a simple harmony improvisation and the depth memory of musician for an outstanding harmony improvisation.

5. Conclusion

In this paper, a novel harmony search combined teaching-learning (HSTL) algorithm is presented to improve the performance and efficiency of the harmony search algorithm. The proposed HSTL algorithm employs the idea of teaching and learning. Four strategies (harmony memory consideration, teaching-learning strategy, local pitch adjusting, and random mutation) are employed to maintain the proper balance between convergence and population diversity. With the process of evolution, the dynamic strategy is adopted to change the parameters HMCR, TLP, BW, and PAR. Numerical experiments show that the dynamic changes of parameters are especially effective in balance between the exploration power and the exploitation power. The population variance analysis indicated that the HSTL algorithm has strong convergence throughout evolution progresses. The sensitivity analysis of HMS parameter showed that it does not have significant influence on complex multimodal problems.

We have compared the performance of proposed HSTL algorithm with the classical HS algorithm and two excellent variant algorithms over a suite of 31 unconstrained numerical optimization functions and evidently concluded that HSTL algorithm is more effective and stable in obtaining high quality solutions and has less FEs, less runtime, and higher success rates under the same conditions.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant no. 60974082 and 81160183, Scientific Research Program funded by Shaanxi Provincial Education Department under Grant no. 12JK0863, Ministry of Education "Chunhui Plan" project (no. Z2011051), Natural

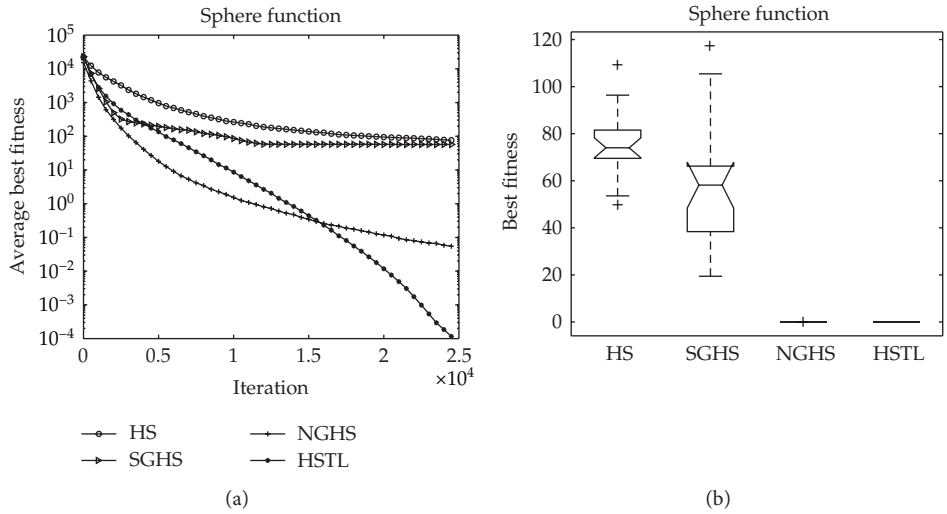

Figure 2: Convergence graph and boxplot for sphere function (F_{10}) on $D = 50$.

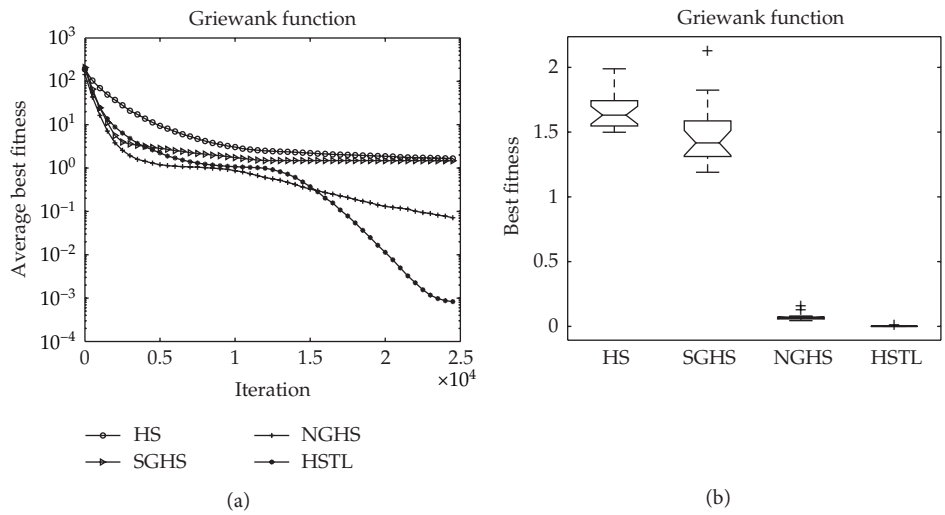

Figure 3: Convergence graph and boxplot for Griewank function (F_4) on $D = 50$.

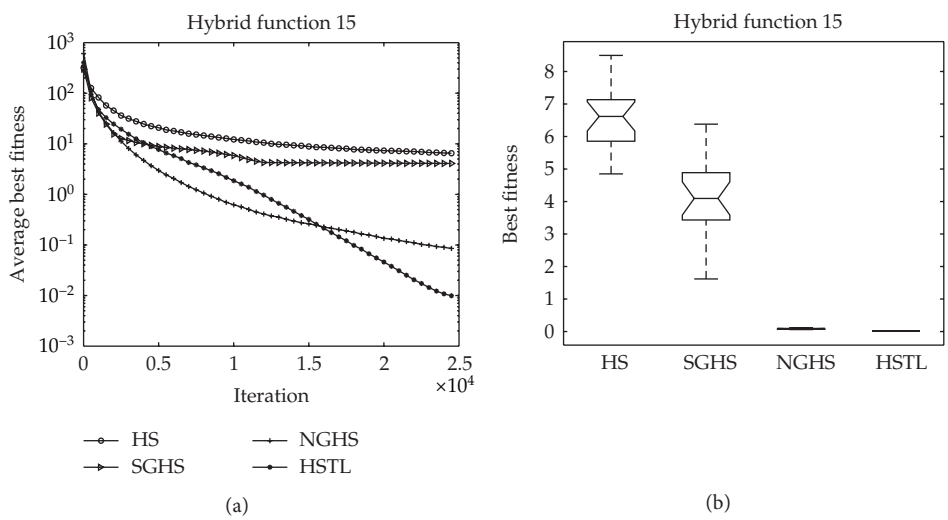

Figure 4: Convergence graph and boxplot for hybrid 15 function (F_{27}) on $D = 50$.

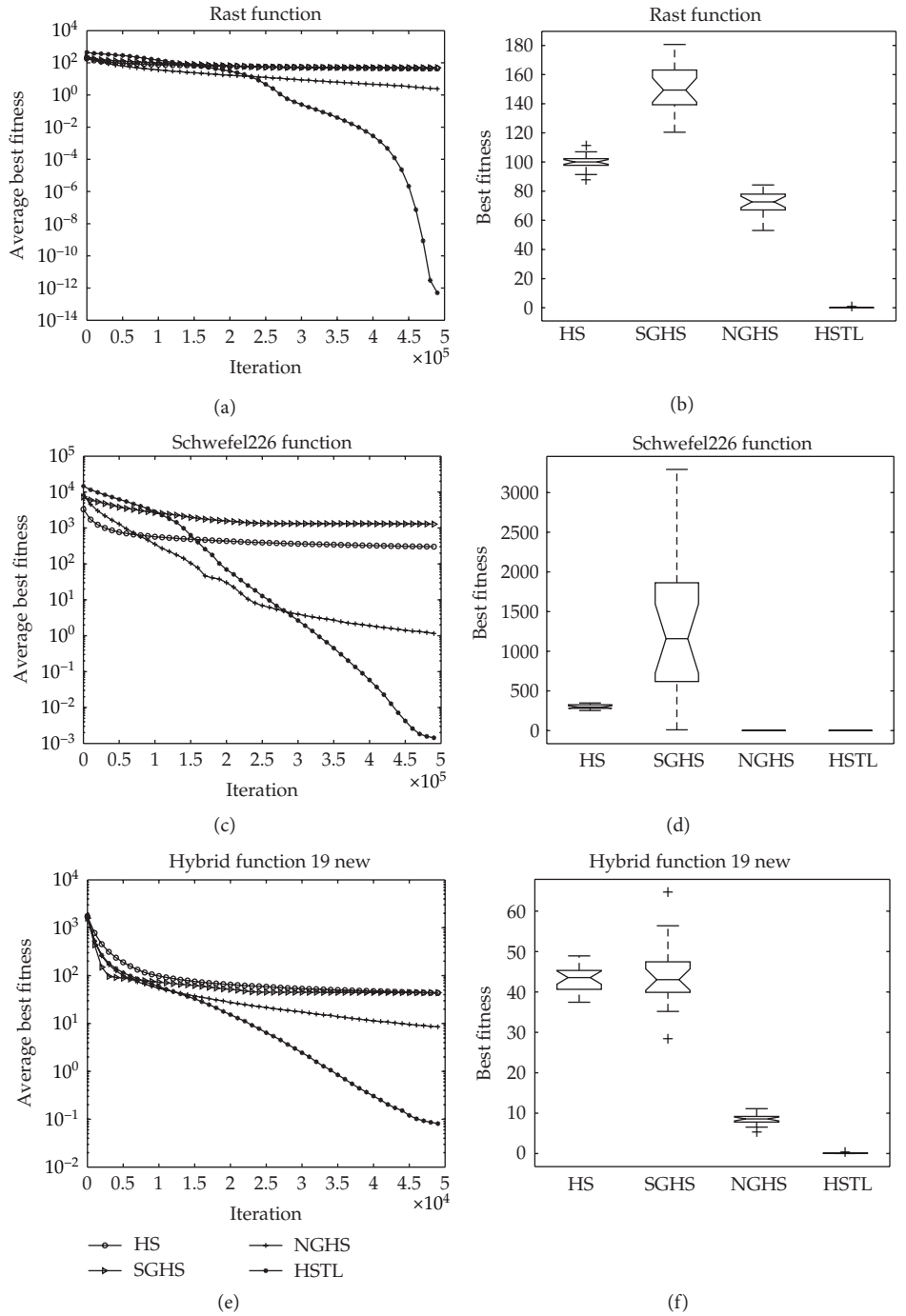

Figure 5: Convergence graphs and boxplots for functions Rastrigin, Schwefel, and hybrid 19 new on $D = 100$.

Table 8: The success rate, mean runtime, and FEs for function F_1–F_{19}, F_{21}–F_{31}.

Function	Algorithm	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
Success rate (SR)	HS	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	SGHS	0%	0%	0%	0%	60%	0%	0%	0%	0%	0%
	NGHS	0%	0%	0%	0%	0%	0%	0%	45%	15%	0%
	HSTL	**100%**	**100%**	**95%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
Mean runtime (s)	HS	19	14	21	50	31	48	15	14	17	16
	SGHS	39	34	41	73	40	72	36	35	39	37
	NGHS	**18**	13	**20**	49	30	48	**14**	11	16	16
	HSTL	20	**10**	21	**48**	**26**	**46**	18	**4**	**14**	19
Mean FEs	HS	500000	500000	500000	500000	500000	500000	500000	500000	500000	500000
	SGHS	500000	500000	500000	500000	397181	500000	500000	500000	500000	500000
	NGHS	500000	500000	500000	500000	500000	500000	500000	425302	497089	500000
	HSTL	**475167**	**291492**	**452830**	**450158**	**386120**	**443818**	**482502**	**105551**	**332693**	**483873**
Function	Algorithm	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{21}
Success rate (SR)	HS	70%	45%	0%	0%	0%	0%	0%	0%	0%	0%
	SGHS	0%	0%	0%	0%	65%	0%	0%	0%	0%	0%
	NGHS	75%	**100%**	0%	0%	90%	15%	0%	0%	0%	0%
	HSTL	**100%**	**100%**	**90%**	**100%**	**100%**	**85%**	**90%**	**100%**	**80%**	**100%**
Mean runtime (s)	HS	8	13	32	32	38	34	55	38	88	527
	SGHS	35	34	55	54	39	56	78	60	114	570
	NGHS	9	**3**	**31**	**31**	**8**	**32**	54	36	92	525
	HSTL	**5**	9	36	**31**	18	34	51	**32**	**81**	467
Mean Fes	HS	264713	471826	500000	500000	500000	500000	500000	500000	500000	500000
	SGHS	500000	500000	500000	500000	323631	500000	500000	500000	500000	500000
	NGHS	335021	**107386**	500000	500000	**114380**	485463	500000	500000	500000	500000
	HSTL	**122555**	240403	**494888**	**418223**	206975	**437342**	**488064**	**434498**	**415872**	**445225**
Function	Algorithm	F_{22}	F_{23}	F_{24}	F_{25}	F_{26}	F_{27}	F_{28}	F_{29}	F_{30}	F_{31}
Success rate (SR)	HS	0%	0%	0%	0%	**100%**	0%	0%	0%	0%	0%
	SGHS	0%	0%	0%	0%	95%	0%	**100%**	0%	0%	0%
	NGHS	0%	0%	0%	0%	**100%**	0%	**100%**	25%	0%	0%
	HSTL	**100%**	**50%**	0%	**100%**	**100%**	**100%**	**100%**	**60%**	**100%**	**100%**
Mean runtime (s)	HS	289	**373**	**359**	**364**	35	310	453	441	436	312
	SGHS	329	416	400	406	133	351	337	476	477	351
	NGHS	293	374	360	366	**22**	**310**	**13**	382	437	317
	HSTL	**273**	375	363	**271**	46	311	104	**363**	**364**	**273**
Mean Fes	HS	500000	500000	500000	500000	54677	500000	500000	500000	500000	500000
	SGHS	500000	500000	500000	500000	181162	500000	341216	500000	500000	500000
	NGHS	500000	500000	500000	500000	**33331**	500000	14644	451443	500000	500000
	HSTL	**459671**	**497514**	**500000**	**367442**	85659	**500000**	**132414**	**432651**	**415004**	**429027**

Table 9: The average runtime, average FEs, average success rate, and costing run of all functions.

Algorithm	Average runtime (ART)	Average FEs (AFE)	Average success rate (ASR)
HS	145.1399	477136	0.069355
SGHS	169.5528	475586.8	0.103226
NGHS	127.2487	434324.5	0.182258
HSTL	**120.8951**	**384181.3**	**0.919355**

Where ART = $\sum_{i=1}^{31}$ runtime$_i$, AFE = $\sum_{i=1}^{31}$ FEs$_i$, ASR = $\sum_{i=1}^{31}$ SR$_i$.

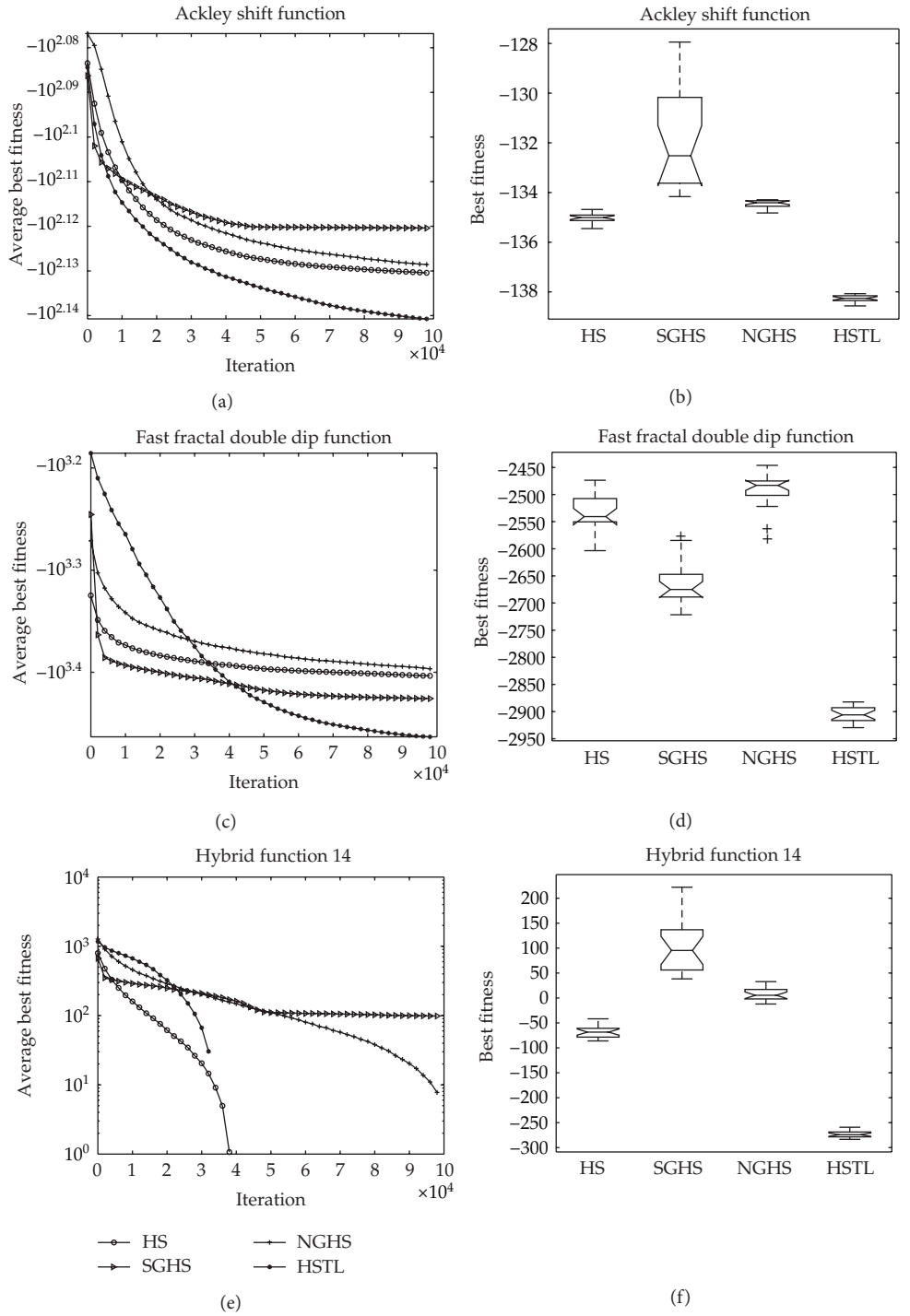

Figure 6: Convergence graphs and boxplots for shift functions Ackley, fast fractal double, and hybrid 14 on $D = 200$.

Table 10: The effect of HMS on the performance of the HSTL algorithm ($D = 50$, FEs = 25000).

HMS	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
5	**7.02E − 06**	1.28E + 00	5.30E − 03	**1.12E − 06**	−4.82E + 01	**3.23E − 03**	4.97E − 02	7.23E + 01	1.36E − 03	1.36E − 03
10	2.10E − 05	8.84E − 01	2.71E − 03	1.26E − 06	−4.82E + 01	4.83E − 03	**7.11E − 10**	4.39E + 01	1.14E − 03	**4.92E − 08**
15	6.98E − 05	7.11E − 01	4.94E − 04	1.29E − 06	−4.83E + 01	6.93E − 03	7.66E − 09	4.35E + 01	**1.06E − 03**	2.19E − 07
20	1.03E − 04	7.07E − 01	8.64E − 04	1.58E − 06	−4.85E + 01	9.06E − 03	4.44E − 08	**4.16E + 01**	1.16E − 03	3.70E − 07
25	1.40E − 04	7.23E − 01	2.23E − 03	1.47E − 06	−4.80E + 01	1.17E − 02	7.97E − 08	4.55E + 01	1.27E − 03	8.60E − 07
30	2.05E − 04	7.72E − 01	**1.82E − 06**	1.90E − 06	−4.81E + 01	1.46E − 02	3.07E − 07	4.39E + 01	1.14E − 03	1.78E − 06
35	2.85E − 04	**7.03E − 01**	2.21E − 06	1.69E − 06	−4.82E + 01	1.61E − 02	4.45E − 07	4.37E + 01	1.09E − 03	2.54E − 06
40	3.81E − 04	7.49E − 01	3.34E − 06	2.00E − 06	**−4.85E + 01**	1.97E − 02	6.64E − 07	4.46E + 01	1.22E − 03	3.88E − 06
HMS	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
05	**7.97E + 03**	**3.57E + 00**	**−4.50E + 02**	**−4.50E + 02**	6.45E + 02	−3.24E + 02	−1.80E + 02	−1.39E + 02	−8.04E + 02	9.41E + 00
10	2.13E + 04	2.72E + 01	−4.50E + 02	−4.50E + 02	5.21E + 02	−3.24E + 02	−1.80E + 02	**−1.40E + 02**	−8.03E + 02	5.08E + 00
15	1.80E + 04	3.40E + 01	−4.50E + 02	−4.50E + 02	6.78E + 02	−3.24E + 02	−1.80E + 02	−1.40E + 02	−8.08E + 02	**4.10E + 00**
20	1.82E + 04	3.93E + 01	−4.50E + 02	−4.49E + 02	5.96E + 02	−3.25E + 02	−1.80E + 02	−1.40E + 02	−8.10E + 02	4.20E + 00
25	9.80E + 03	5.43E + 01	−4.50E + 02	−4.48E + 02	**5.19E + 02**	−3.24E + 02	−1.80E + 02	−1.40E + 02	−8.10E + 02	4.27E + 00
30	2.18E + 04	6.84E + 01	−4.50E + 02	−4.47E + 02	5.33E + 02	−3.25E + 02	−1.80E + 02	−1.40E + 02	−8.11E + 02	4.33E + 00
35	1.46E + 04	9.77E + 01	−4.50E + 02	−4.46E + 02	8.04E + 02	**−3.26E + 02**	**−1.80E + 02**	−1.40E + 02	**−8.11E + 02**	4.24E + 00
40	1.24E + 04	1.18E + 02	−4.50E + 02	−4.44E + 02	8.18E + 02	−3.26E + 02	−1.80E + 02	−1.40E + 02	−8.10E + 02	4.29E + 00
HMS	F_{21}	F_{22}	F_{23}	F_{24}	F_{25}	F_{26}	F_{27}	F_{28}	F_{29}	F_{30}
05	2.02E − 01	9.64E + 00	**−4.50E + 02**	4.74E + 02	−3.26E + 02	5.05E − 03	−4.50E + 02	6.01E + 02	−3.29E + 02	1.50E − 03
10	6.50E − 02	5.27E + 00	−4.50E + 02	4.71E + 02	−3.26E + 02	4.96E − 03	−4.50E + 02	7.29E + 02	−3.28E + 02	1.41E − 03
15	7.34E − 02	4.40E + 00	−4.50E + 02	**4.70E + 02**	−3.27E + 02	**4.86E − 03**	−4.50E + 02	**4.19E + 02**	−3.29E + 02	2.21E − 02
20	5.28E − 02	4.27E + 00	−4.50E + 02	4.96E + 02	−3.26E + 02	5.08E − 03	−4.50E + 02	4.26E + 02	−3.29E + 02	**1.25E − 03**
25	3.54E − 04	4.06E + 00	−4.50E + 02	4.93E + 02	−3.26E + 02	5.37E − 03	−4.50E + 02	9.77E + 02	−3.29E + 02	1.53E − 03
30	**3.34E − 04**	**3.80E + 00**	−4.50E + 02	4.91E + 02	**−3.27E + 02**	5.22E − 03	−4.50E + 02	6.57E + 02	**−3.29E + 02**	1.39E − 03
35	4.41E − 04	4.06E + 00	−4.50E + 02	4.93E + 02	−3.27E + 02	5.16E − 03	−4.50E + 02	5.06E + 02	−3.29E + 02	1.61E − 03
40	3.65E − 04	4.06E + 00	−4.50E + 02	4.86E + 02	−3.27E + 02	5.40E − 03	**−4.50E + 02**	8.34E + 02	−3.29E + 02	1.45E − 03

Table 11

$F_1 \sim F_{10}$:	$1.0E-05$	$1.0E+00$	$1.0E-05$	$1.0E-05$	$5.0E+00$	$1.0E-02$	$1.0E-10$	$2D$	$1.0E+00$	$1.0E-10$
$F_{11} \sim F_{20}$:	$2DE+04$	D	$1.0E-04$	$1.0E-01$	$50D$	$1.0E+00$	$1.0E-04$	$1.0E-02$	$1.0E+00$	$1.0E+01$
$F_{21} \sim F_{30}$:	$1.0E-03$	$1.0E+01$	$1.0E-04$	$2D$	$2D$	$1.0E-02$	$1.0E-05$	D	$1.0E+00$	$1.0E-02$

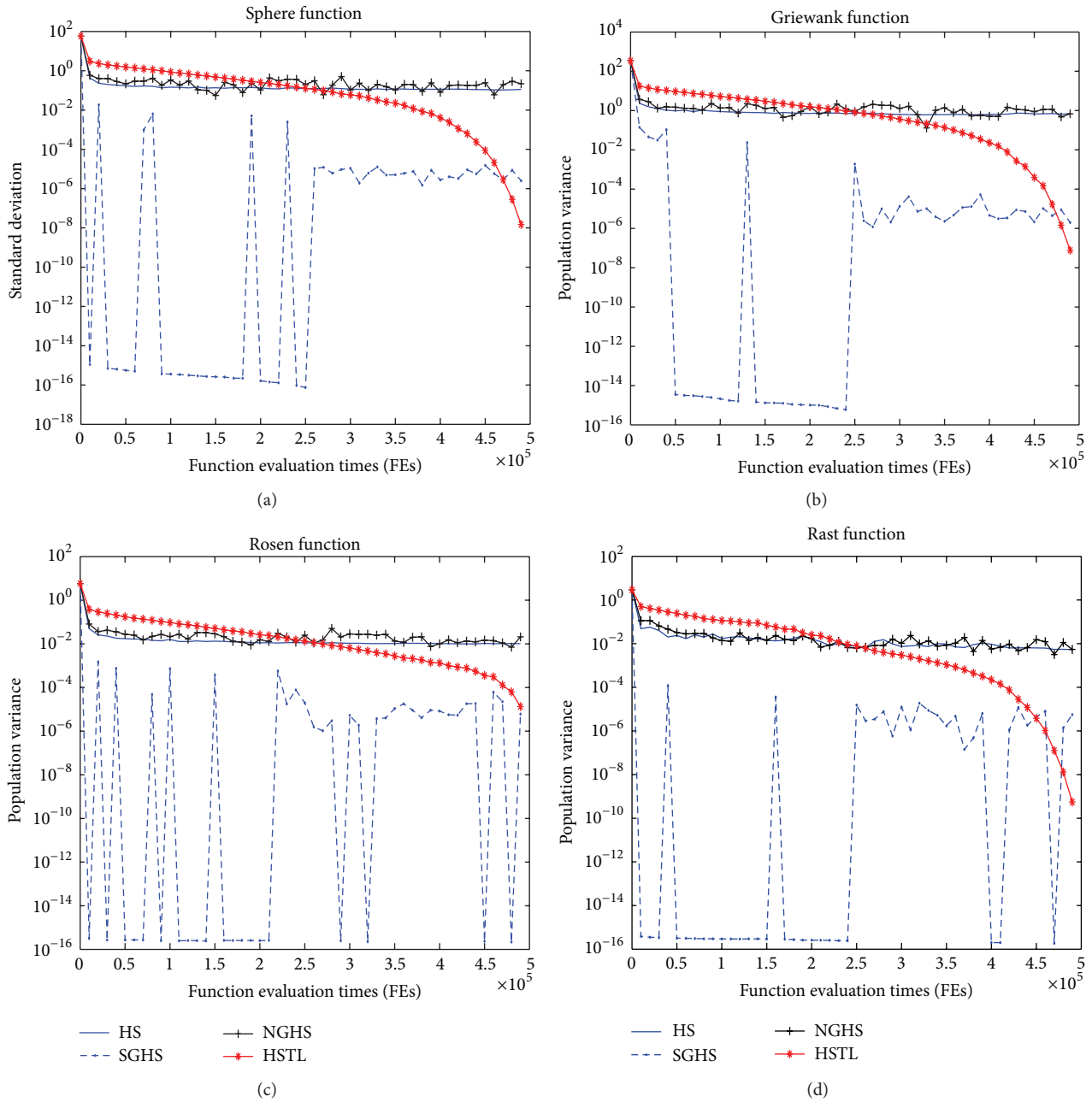

Figure 7: The population variance graphs.

Science Foundation of Ningxia Hui Autonomous Region (no. NZ12179), and Scientific Research Fund of Ningxia Education Department (no. NGY2011042).

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, J. H. Kim, and G. V. Loganathan, "Harmony search optimization: application to pipe network design," *International Journal of Modelling and Simulation*, vol. 22, no. 2, pp. 125–133, 2002.

[4] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm," *Computers and Structures*, vol. 82, no. 9-10, pp. 781–798, 2004.

[5] J. Kang and W. Zhang, "Combination of fuzzy C-means and harmony search algorithms for clustering of text document," *Journal of Computational Information Systems*, vol. 16, no. 7, pp. 5980–5986, 2011.

[6] A. Vasebi, M. Fesanghary, and S. M. T. Bathaee, "Combined heat and power economic dispatch by harmony search algorithm," *International Journal of Electrical Power and Energy Systems*, vol. 29, no. 10, pp. 713–719, 2007.

[7] Z. W. Geem, "Optimal scheduling of multiple dam system using harmony search algorithm," in *Lecture Notes in Computer Science*, vol. 4507, pp. 316–323, 2007.

[8] 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.

[9] S. Tuo and L. Yong, "An improved harmony search algorithm with chaos," *Journal of Computational Information Systems*, vol. 8, no. 10, pp. 4269–4276, 2012.

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

[11] 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.

[12] P. Chakraborty, G. G. Roy, S. Das, D. Jain, and A. Abraham, "An improved harmony search algorithm with differential mutation operator," *Fundamenta Informaticae*, vol. 95, no. 4, pp. 401–426, 2009.

[13] D. X. Zou, L. Q. Gao, J. Wu, S. Li, and Y. Li, "A novel global harmony search algorithm for reliability problems," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 307–316, 2010.

[14] D. Zou, L. Gao, J. Wu, and S. Li, "Novel global harmony search algorithm for unconstrained problems," *Neurocomputing*, vol. 73, no. 16–18, pp. 3308–3318, 2010.

[15] X. Z. Gao, X. Wang, and S. J. Ovaska, "Uni-modal and multi-modal optimization using modified Harmony Search methods," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 10, pp. 2985–2996, 2009.

[16] 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.

[17] P. Yadav, R. Kumar, S. K. Panda, and C. S. Chang, "An intelligent tuned harmony search algorithm for optimization," *Information Sciences*, vol. 196, pp. 47–72, 2012.

[18] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems," *CAD Computer Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[19] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems," *Information Sciences*, vol. 183, pp. 1–15, 2012.

[20] R. V. Rao and V. Patel, "An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 3, pp. 535–560, 2012.

[21] R. V. Rao and V. J. Savsani, *Mechanical Design Optimization Using Advanced Optimization Techniques*, Springer-Verlag, London, UK, 2012.

[22] R. V. Rao and V. Patel, "Multi-objective optimization of heat exchangers using a modified teaching-learning based optimization algorithm," *Applied Mathematical Modelling*, vol. 37, no. 3, pp. 1147–1162, 2013.

[23] V. R. Rao and V. Patel, "Multi-objective optimization of two stage thermoelectric cooler using a modified teaching-learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 430–445, 2013.

[24] R. V. Rao, V. J. Savsani, and J. Balic, "Teaching-learning-based optimization algorithm for unconstrained and constrained real parameter optimization problems," *Engineering Optimization*, vol. 44, no. 12, pp. 1447–1462, 2012.

[25] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 41, no. 1, pp. 89–106, 2011.

[26] H. Sarvari and K. Zamanifar, "Improvement of harmony search algorithm by using statistical analysis," *Artificial Intelligence Review*, vol. 37, no. 3, pp. 181–215, 2012.

[27] M. Fukushima, Test Functions for Unconstrained Global Optimization, http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm.

[28] K. Tang, X. Yao, P. N. Suganthan et al., Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization, http://www.ntu.edu.sg/home/EPNSugan/, 2008.

[29] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large scale global optimization," Tech. Rep., Nature Inspired Computation and Applications Laboratory, USTC, China & Nanyang Technological University, Nanyang Avenue, Singapore, 2009.

[30] F. Herrera, M. Lozano, and D. Molina, "Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other meta-heuristics for large scale continuous optimization problems," http://sci2s.ugr.es/eamhco/CFP.php.