Appl Intell (2016) 44:725–753
DOI 10.1007/s10489-015-0721-7

An efficient modified harmony search algorithm with intersect mutation operator and cellular local search for continuous function optimization problems

Jin Yi[1] · Liang Gao[1] · Xinyu Li[1] · Jie Gao[1]

Published online: 14 November 2015

Abstract This paper proposes a modified harmony search (MHS) algorithm with an intersect mutation operator and cellular local search for continuous function optimization problems. Instead of focusing on the intelligent tuning of the parameters during the searching process, the MHS algorithm divides all harmonies in harmony memory into a better part and a worse part according to their fitness. The novel intersect mutation operation has been developed to generate new -harmony vectors. Furthermore, a cellular local search also has been developed in MHS, that helps to improve the optimization performance by exploring a huge search space in the early run phase to avoid premature, and exploiting a small region in the later run phase to refine the final solutions. To obtain better parameter settings for the proposed MHS algorithm, the impacts of the parameters are analyzed by an orthogonal test and a range analysis method. Finally, two sets of famous benchmark functions have been used to test and evaluate the performance of the proposed MHS algorithm. Functions in these benchmark sets have different characteristics so they can give a comprehensive evaluation on the performance of MHS. The experimental results show that the proposed algorithm not only performs better than those state-of-the-art HS variants but is also competitive with other famous meta-heuristic algorithms in terms of the solution accuracy and efficiency.

Keywords Harmony search · Continuous optimization · Intersect mutation operator · Cellular local search

1 Introduction

Along with the development of modern society, the real-life problems we face in the areas of science, engineering, economics and business are becoming more and more complex and difficult to solve. Searching for efficient methods to solve optimization problems has been one of the hottest topics in recent decades [1]. All the existing methods can be divided into two categories: exact methods and approximation methods. Exact methods can find the global optimum, but these methods are efficient only when the scale of the problem is small. However, in contrast, although the approximation methods cannot guarantee obtaining the global optimum, they can solve large-scale problems in a reasonable time. We all know that most real-life optimization problems are large scale problems. Therefore, the approximation methods are more suitable for these problems than exact methods. Thus, this paper focuses on approximation methods and proposes an effective novel method for unconstrained optimization problems.

The meta-heuristic algorithm that combines rules and randomness to imitate natural or social phenomena is one among the most important types of approximation methods. Research on meta-heuristic algorithms is the hottest topic in current studies on approximation methods. The phenomena of popular meta-heuristic algorithms include

✉ Xinyu Li
lixinyu@mail.hust.edu.cn

Jin Yi
yijin@hust.edu.cn

Liang Gao
gaoliang@mail.hust.edu.cn

Jie Gao
D201577194@hust.edu.cn

[1] The State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China

```
HS Algorithm
Begin
    Define fitness function $f(x)$, $x = (x_1, x_2, ..., x_N)^T$
    Define ($HMCR$), ($PAR$), ($HMS$)
    Define Maximum number of iterations ($NI$).
    $HM \leftarrow$ Generate Initial Population()
    min = minimum visible value
    max = maximum visible value
    while ($t \le NI$) do
        while ($i \le$ number of variables)
            if ($r_1 < HMCR$) then
                choose a value from $HM$ for $i$
                    $x^i_{new} = HM(a,i)$, $a \in \{1, 2, ..., HMS\}$
                if ($r_2 < PAR$) then
                    Adjust the value of i by :
                        $x^i_{new} = x^i_{new} \pm r_3 \cdot BW$
                end if
            else
                choose a random variable:
                    $x^i_{new} = LB_i + r_4 \cdot (UB_i - LB_i)$
        end while
        if ($fitness(x_{new}) \le worst(fitness(HM))$) then
            Update the $HM$
        end if
    end while
        Find the current best solution
end
```

Fig. 1 Pseudo code of the Harmony Search algorithm

biological evolutionary processes (e.g., the genetic algorithm(GA) [2] and differential evolution (DE) [3, 4]); animal behavior (e.g., particle swarm optimization (PSO) [5], artificial bee colony (ABC) [6], ant colony optimization (ACO) [7] and cuckoo search (CS) [8]); physical process (e.g., simulated annealing (SA) [9]); and chemical process (e.g., chemical reaction optimization (CRO) [10]). Over the last decades, many excellent meta-heuristic algorithms have been successfully applied to various real-life optimization problems and obtained better solutions than the classic methods.

The harmony search (HS) algorithm is one of the recently developed meta-heuristic algorithms [11]. It imitates the music improvisation process in which musicians continuously adjust the pitch of their instruments to achieve better harmony. The search process of global optimization problems is similar to the music improvisation process, in that each decision variable continuously changes its value during the search process to converge to the global optimum. Mahdavi [12] concluded that the HS algorithm has the characteristics of few mathematical requirements, easy implementation, and fast convergence speed. Hence, HS has caught many researchers' attention and has been applied in many areas, such as water network design [13], structure optimization [14], medical physics [15], image segmentation [16], face-milling parameters optimization [17], scheduling [18], motion estimation [19], and others [20–22].

The HS algorithm has three important parameters includes harmony memory consideration rate *HMCR*, pitch adjustment rate *PAR* and bandwidth *bw*. HS algorithm is not efficient enough in performing local search in numerical optimization applications and is sensitive to the three parameter settings [23]. Thus, it is of great interest to improve the performance of the HS algorithm. Based on recent literature about the HS algorithm, the related work can be divided into two directions: to modify the structure of the HS algorithm; to apply more efficient parameter tuning strategies.

In the modification of HS structure, Oman and Mahdavi proposed a global-best harmony search (GHS) inspired by swarm intelligence [24]. In the GHS algorithm, new-harmony vectors can learn from the current best harmony vector in the harmony memory pool with probability *HMCR*PAR*. The experimental results showed that GHS could perform better than the classical HS. Wang et al. presented an effective differential harmony search (DHS) algorithm and applied it to solve economic load dispatch problems [25]. The main difference between DHS and HS is that DHS applies a mutation operation inspired by differential evolution algorithm to generate the new-harmony vectors. Abhik et al. proposed an opposition-based harmony search (OHS) algorithm [26]. In the OHS algorithm, the concept of opposition-based learning was employed in HM initialization and generation jumping, and the simulation results showed the effectiveness, robustness, and superiority of the OHS algorithm. In another work, Wang et al. presented an opposition-based learning HS with mutation (OLHS-M) for solving global continuous optimization problems [27]. Different from OHS, in OLHS-M, the opposition-based learning technique is incorporated to the process of improvisation to enlarge the search space. Zou et al. proposed a novel global harmony search (NGHS)

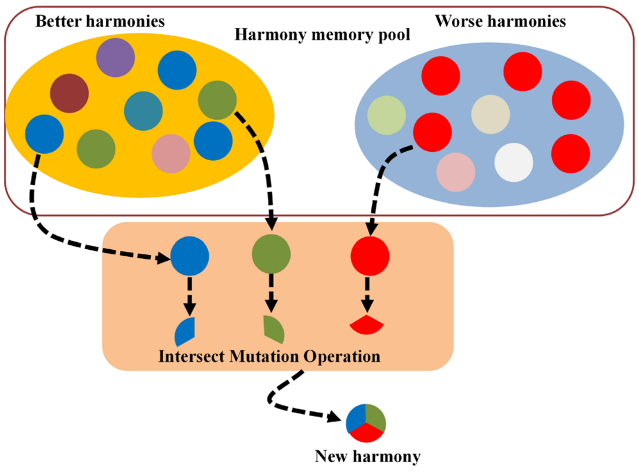


Fig. 2 The procedure for improvising new harmony by the intersect mutation operation

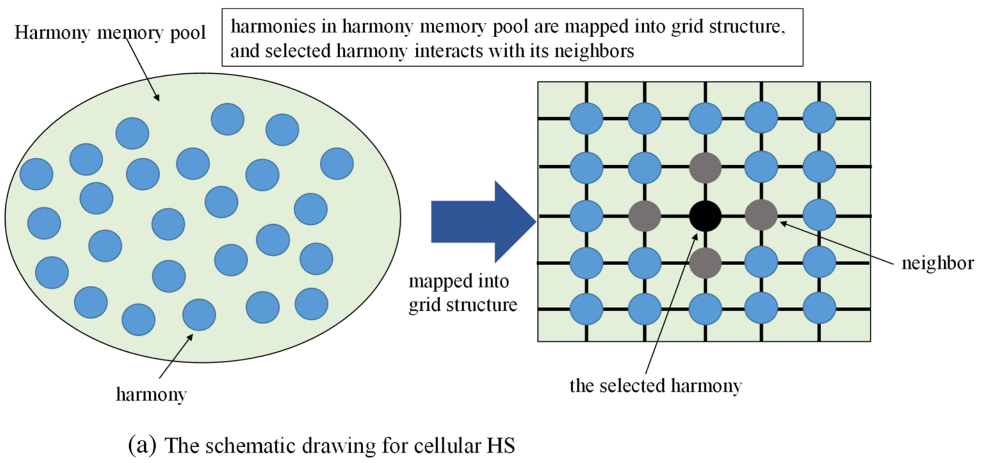
```
/*Improvise new harmonies*/
for $i = 1 : HMS$
    for $j = 1 : Dim$
        if $(r1 < HMCR)$
            $x_i^j = x_a^j$, where $a \in (1, 2, ..., HMS)$
            if $(r2 < PAR)$
                if $(i < M * HMS)$
                    $x_i^j = x_{wr1}^j + F \cdot (x_{br1}^j - x_{br2}^j),\ br1 \neq br1 \neq wr1$
                else
                    $x_i^j = x_{br1}^j + F \cdot (x_{wr1}^j - x_{wr2}^j),\ br1 \neq wr1 \neq wr2$
                end if
            end if
        else
            $x_i^j = LB^j + r \cdot (UB^j - LB^j)$, where $r \in (0,1)$
        end if
    end for
end for
```

Fig. 3 Pseudo code for improvising new harmonies

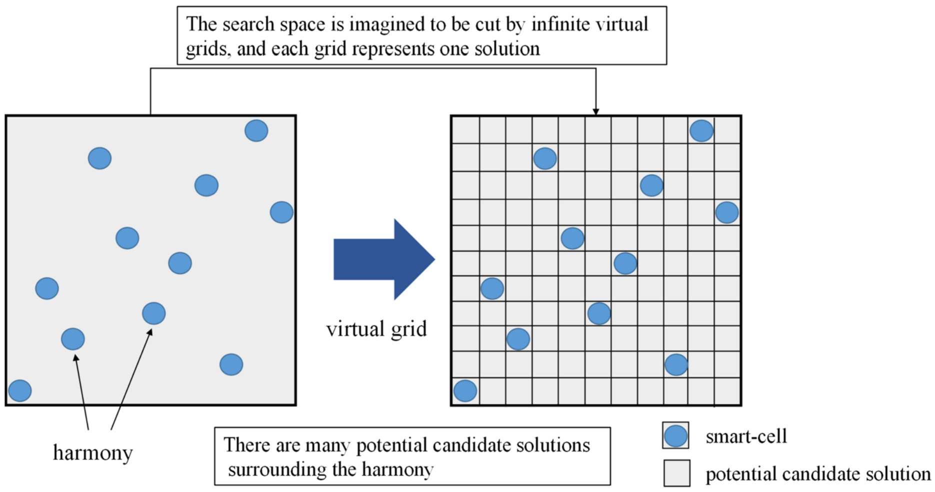
[28]. NGHS designed a novel location updating strategy to make it easier to converge, and a genetic mutation operation was utilized to prevent the NGHS from being trapped into local optima. The concept of cellular automata has also been applied in HS [29, 30]. The main idea was to arrange the topological structure of the population as a two-dimensional grid, where each individual in the grid is a cell and interacts with its neighbors. Castelli et al. proposed an innovative improved version of HS named Melody Search (MS), which adopted the basic idea of HS but with quite a different structure [31]. The MS algorithm mimics the musical performance processes of group improvisation to find the best succession of pitches within a melody. In such a group, the memories of different players interact with each other and maintain the diversity of musicians (with different tastes, ideas and experiences), hence leading to a faster achievement of the best subsequence of pitches. Al-Betar et al. introduced the island model concepts and embedded them into the framework of HS algorithm where the new algorithm is refer to island HS (iHS) [32]. In iHS, the individuals are distributed in separated sub-population (islands) and evolve separately within specific generations. Then, the individuals on different islands exchange their information through a process named migration. The main idea of iHS is to maintain the diversity of the population and allow individuals on different islands interact with each other. In utilizing efficient parameters tuning strategies, Mahdavi et al. proposed an improved harmony search (IHS) by dynamically updating the two control parameters *PAR* and *bw* [12]. Pan et al. presented a self-adaptive global best harmony (SGHS) where parameters such as *HMCR* and *PAR* were self-adapted by the proposed learning strategy and the parameter *bw* changed dynamically with iteration number [33]. The NDHS, which was developed by Chen et al. had a new memory consideration scheme based on the tournament selection rule, and the two parameters, *PAR* and *bw*, were dynamically adjusted with respect to the evaluation of the search process [23]. The intelligent tuned harmony search (ITHS), which was presented by Yadav et al., learned the concepts from decision making in a group [34]. The self-adaptive pitch adjustment strategy adopted by the dynamic sub-population based on the harmony memory helped ITHS to maintain the proper balance between diversification and intensification throughout the search process. Kattan et al. proposed a dynamic self-adaptive harmony search (DSHS) by introducing two criteria to drive the optimization process: the best-to-worst ratio to adjust the *PAR* value and a dynamic *bw* based on the standard deviation of each dimension variable in the harmony memory (HM) [35]. Enayatifar et al. proposed a novel harmony search algorithm based on learning automata (LAHS) [36]. The learning-based adjustment mechanism showed a new approach for parameter tuning. Kumar et al. proposed a parameter adaptive harmony search (PAHS) based on the combinations of linear and exponential changes of the parameters *HMCR* and *PAR* [37]. The numerical simulation results showed that linear change in *HMCR* and exponential change in *PAR*, provided better results than the other techniques in a noisy and scalable environment.

From the literature mentioned above, we can see that many novel improvements were inspired by other meta-heuristic algorithms, and the self-adaptation strategies became more and more complicated. Although these novel approaches performed much better than the classic HS algorithm, there was still considerable room for improvement in dealing with diverse problems. Based on the idea of balanced intensification (exploitation) and diversification (exploration), this study proposes a modified HS algorithm with an intersect mutation operator and a cellular local search, which is abbreviated as MHS. To improve the diversity of HM, an intersect mutation operation is embedded into the HS structure: first, the harmony vectors in the initial harmony memory are divided into a better part and a worse part. Then, an intersect mutation operation between the better part and the worse part is proposed to generate new harmonies. Furthermore, after each harmony improvising procedure, a cellular local search based on cellular neighborhoods is performed to enhance the local exploitation. To further analyze the effects of the parameters in proposed algorithm, an orthogonal test is conducted, and a parameter settings recommendation is presented. Finally, two sets of famous benchmark functions are used to test and evaluate the performance of the proposed MHS algorithm. Computational results and comparisons show that the proposed algorithm not only outperforms the compared HS variants

Fig. 4 Comparison of the basic idea between cellular HS and cellular local search in MHS


Harmony memory pool
harmonies in harmony memory pool are mapped into grid structure, and selected harmony interacts with its neighbors
mapped into grid structure
neighbor
harmony
the selected harmony

(a) The schematic drawing for cellular HS


The search space is imagined to be cut by infinite virtual grids, and each grid represents one solution
virtual grid
harmony
There are many potential candidate solutions surrounding the harmony
smart-cell
potential candidate solution

(b) The schematic drawing for cellular local search in MHS

but is also competitive with other meta-heuristic algorithms in terms of solution accuracy and efficiency.

The rest of this paper is organized as follows: Section 2 introduces the classic HS algorithm. Section 3 describes the proposed algorithm in detail. Section 4 shows the orthogonal experiment for parameter settings, computational results, and comparisons. Finally, Section 5 presents the concluding remarks.

2 The classic harmony search algorithm

In the classic HS algorithm, each solution is called “harmony” and represented by an n-dimension real vector. An initial population of harmony vectors are randomly generated and stored in a HM. Then, a new candidate harmony is improvised from all of the solutions in the HM using a memory consideration rule, a pitch adjustment rule, and a random re-initialization. Finally, the HM is updated by comparing the fitness between the new candidate harmony and the worst harmony in the current HM. The worst harmony vector is replaced by the new candidate harmony vector if it is better than the worst harmony vector in the HM. The improvisation and updating processes are repeated until a predefined stopping criterion is reached. The harmony search procedure is as follows [12]:

Step 1: Initialize the problem and algorithm parameters.

The optimization problem is specified as follows:

$$\text{Minimize } f(x) \text{ subject to } x_i \in [LB_i, UB_i], \quad i = 1, 2, \ldots, n$$

where $f(x)$ is an objective function; $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ is the set of decision variables; n is the number of decision variables; and LB_i, UB_i are the lower and upper bounds for decision variable x_i, respectively.

In addition, the HS algorithm parameters are specified in this step. These parameters are harmony memory size (*HMS*), or the number of

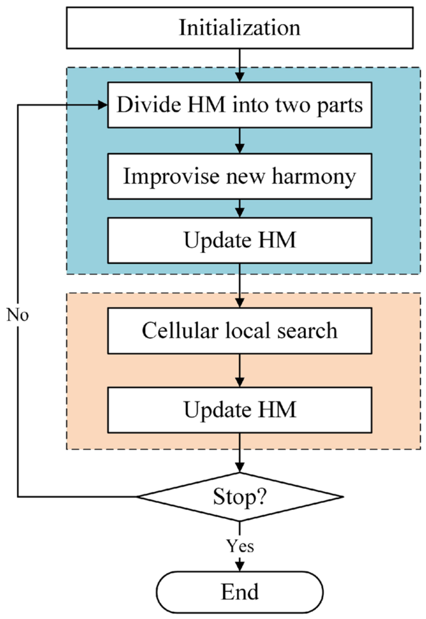


Fig. 5 Flowchart of MHS algorithm

solution vectors in the harmony memory; harmony memory considering rate (*HMCR*); pitch adjusting rate (*PAR*); distance bandwidth (*BW*); number of decision variables (N) and the number of improvisations (*NI*), or the stopping criterion.

Step 2: Initialize the harmony memory (HM).

In step 2, the initial solution variables are randomly generated from the feasible region and filled in the harmony memory matrix HM:

$$HM = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_N^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_N^2 & f(x^2) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_N^{HMS} & f(x^{HMS}) \end{array}\right] \quad (1)$$

Step 3: Improvise a new harmony.

A new harmony vector $\mathbf{X}_{\text{new}} = (x_{\text{new}}^1, x_{\text{new}}^2, ..., x_{\text{new}}^n)$ is improvised based on three rules: memory consideration, pitch adjustment and random selection. First, a random number r_1 is generated in the range [0, 1]. If r_1 is less than $HMCR$, the decision variable x_{new}^i is generated by memory consideration; otherwise, x_{new}^i is obtained by random selection. In the memory consideration, another random number r_2 is generated in the range [0, 1], if r_2 is less than the value of the pitch adjustment rate (PAR), x_{new}^i is selected from any harmony vector in the harmony memory. Otherwise, x_{new}^i is generated as follows:

$$x_{\text{new}}^i = x_{\text{new}}^i \pm r_3 \cdot BW \quad (2)$$

where r_3 is a random number in [0, 1].

Step 4: Update the harmony memory.

In this step, the decision of whether the new harmony vector improvised in step 3 should be included into the harmony memory is made. If the fitness of the new harmony vector is better than the worst harmony vector in the current harmony memory, it will be included into the HM. Meanwhile, the current worst harmony will be excluded from the HM.

Step 5: Check the stopping criterion.

The HS algorithm is terminated when the stopping criterion has been met, for example, when the maximum number of improvisations *NI* is met. Otherwise, Steps 3 and 4 are repeated.

Above steps are well illustrated using pseudo code in Fig. 1.

3 Proposed modified HS algorithm with intersect mutation operator and cellular local search

Intensification (exploitation ability) and diversification (exploration ability) are the two major aspects that determine the effectiveness of the meta-heuristic algorithms. If the algorithm focuses too much on diversification, it will converge very slowly. On the contrary, if the algorithm focuses too much on intensification, convergence can more easily be premature. Hence, algorithms should tradeoff these two aspects and maintain a good balance between them.

In this section, the modified HS algorithm with intersect mutation operator and cellular local search (MHS) is developed. MHS is similar to the HS algorithm, with the following main differences: (1) the parameter bandwidth *bw* is removed; (2) the algorithm includes an intersect mutation operation that uses the *PAR* parameter; (3) the algorithm performs a cellular local search after each new harmony improvisation. First, the initialized harmonies are ranked from better to worse according to their fitness values and then divided into two parts: the better part and the worse part. Here, we introduce a constant coefficient M (0<M<1), which stands for the proportion of better harmonies in the harmony memory pool. Then, the intersect mutation operator is embedded into the HS structure to maintain the diversity of harmony memory. Meanwhile, a cellular local search is adopted to enhance the exploitation ability of the MHS. In the proposed approach, intensification and diversification are well balanced.

3.1 HS with intersect mutation operator

Inspired by the work of Zhou et al. [38], we use the intersect mutation operation to improvise new harmonies with the

Table 1 Test functions

Function	Range	x^*	$f(x^*)$
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	0
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	$[0, 0, \ldots, 0]$	0
$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	0
$f_4(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	0
$f_5(x) = \sum_{i=1}^{D} i x_i^4 + rand[0, 1)$	$[-1.28, 1.28]^D$	$[0, 0, \ldots, 0]$	0
$f_6(x) = \sum_{i=1}^{D-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_1 - 1)^2\right)$	$[-30, 30]^D$	$[1, 1, \ldots, 1]$	0
$f_7(x) = 418.9829{*}D - \sum_{i=1} D\left(x_i \sin(\sqrt{\lvert x_i \rvert})\right)$	$[-500, 500]^D$	$[420.9687, \ldots, 420.9687]$	0
$f_8(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[-5.12, 5.12]^D$	$[0, 0, \ldots, 0]$	0
$f_9(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{n}\cos(2\pi x_i)\right)$	$[-32, 32]^D$	$[0, 0, \ldots, 0]$	0
$f_{10}(x) = \frac{1}{100}\sum_{i=1}^{D} x_i 2 - \prod_{i=1}^{D} \cos\left(x_i/\sqrt{i}\right) + 1$	$[-600, 600]^D$	$[0, 0, \ldots, 0]$	0
$f_{11}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right)\right]\right) - n\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k \cdot 0.5\right)\right]$, $a = 0.5, b = 3.0, k\max = 20.$	$[-0.5, 0.5]^D$	$[0, 0, \ldots, 0]$	0
$f_{12}(x) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^{D} x_i 2}\right) - 0.5}{\left(1 + 0.001\sum_{i=1}^{D} x_i^2\right)^2}$	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	0
$f_{13}(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2\left[1 + 10\sin^2(3\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{1}{4}(x_i + 1),\ u(x_i, a, k, m)\begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^2, & x_i < -a. \end{cases}$	$[-50, 50]^D$	$[-1, -1, \ldots, -1]$	0
$f_{14}(x) = \sum_{i=1}^{D} z_i^2 - 450$	$[-100, 100]^D$	$[o(1), o(2), \ldots, o(n)]$	0
$f_{15}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{D} x_j\right)^2 - 450$	$[-100, 100]^D$	$[o(1), 0(2), \ldots, o(n)]$	-450
$f_{16}(x) = \sum_{i=1}^{D-1}\left(100(z_{i+1} - zi)^2 + (z_i - 1)^2\right) + 390$	$[-100, 100]^D$	$[o(1), o(2), \ldots, o(n)]$	390
$f_{17}(x) = \sum_{i=1}^{D}\left(z_i^2 - 10\cos(2\pi z_i) + 10\right) - 330$	$[-100, 100]^D$	$[o(1), o(2), \ldots, o(n)]$	-330
$f_{18}(x) = 1 + 1/4000{*}\sum_{i=1}^{D} Z_i^D - \Pi_{i=1}^{D}\cos\left(\frac{Z_i}{i}\right) - 180$	$[-100, 100]^D$	$[o(1), 0(2), \ldots, o(n)]$	-180
$f_{19}(x) = \sum_{i=1}^{D}\left(z_1^D - 10\cos(2\pi z_i + 10) + 10\right) - 330$	$[-100, 100]^D$	$[o(1), o(2), \ldots, o(n)]$	-300

probability of *PAR*. The whole procedure is shown in Fig. 2. For the better part, we mutate the vectors with one harmony ($wr1$) chosen from the worse part and two harmonies ($br1$ and $br2$) chosen from the better part, as the formula below shows:

$$x_i = x_{wr1} + F \cdot (x_{br1} - x_{br2}),\ br1 \ne br2 \ne wr1 \ne i \quad (3)$$

where F ($0 < F < 1$) is the mutation parameter.

For the worse part, we mutate the vectors with one harmony ($br1$) chosen from the better part and two harmonies ($wr1$ and $wr2$) chosen from the worse part, as the formula below shows:

$$x_i = x_{br1} + F \cdot (x_{wr1} - x_{wr2}),\ br1 \ne br2 \ne wr1 \ne i \quad (4)$$

The detailed pseudo code explaining the various steps of improvising new harmonies is shown in Fig. 3. for easier implementation and understanding of the proposed approach.

3.2 Cellular local search

The cellular automata (CA) model is a discrete dynamical system that can produce complex phenomena [39]. The essential idea of the CA model is to simulate macro-behavior by the micro-dynamics, which is produced by the interaction of a population of individuals who are connected

Table 2 Influence factors and level values

Levels	*HMS*	*HMCR*	*PAR*	*M*	F
1	10	0.40	0.10	0.20	0.40
2	20	0.60	0.30	0.30	0.50
3	40	0.80	0.50	0.40	0.60
4	60	0.90	0.70	0.50	0.70
5	100	0.99	0.90	0.60	0.80

Table 3 Orthogonal experimental results

Test no.	Factor						Total score
	HMS(A)	*HMCR*(B)	*PAR*(C)	*M*(D)	*F*(E)	Empty	
1	1(10)	1(0.40)	1(0.10)	1(0.20)	1(0.40)	1	18
2	1	2(0.60)	2(0.30)	2(0.30)	2(0.50)	2	30
3	1	3(0.80)	3(0.50)	3(0.40)	3(0.60)	3	76
4	1	4(0.90)	4(0.70)	4(0.50)	4(0.70)	4	105
5	1	5(0.99)	5(0.90)	5(0.60)	5(0.80)	5	139
6	2(20)	1	2	3	4	5	51
7	2	2	3	4	5	1	70
8	2	3	4	5	1	2	114
9	2	5	1	2	3	4	157
10	2	5	1	2	3	4	166
11	3(40)	1	3	5	2	4	69
12	3	2	4	1	3	5	111
13	3	3	5	2	4	1	156
14	3	4	1	3	5	2	181
15	3	5	2	4	1	3	200
16	4(60)	1	4	2	5	3	119
17	4	2	5	3	1	4	111
18	4	3	1	4	2	5	145
19	4	4	2	5	3	1	215
20	4	5	3	1	4	2	198
21	5(100)	1	5	4	3	2	150
22	5	2	1	5	4	3	118
23	5	3	2	1	5	4	186
24	5	4	3	2	1	5	215
25	5	5	4	3	2	1	249
K_1	368	407	628	670	658	708	K =3349
K_2	558	440	682	686	650	672	P =448632.04
K_3	717	677	628	668	718	670	
K_4	778	873	698	670	628	637	
K_5	918	952	713	655	695	661	
Q_i	484909	497202.2	449893	448729	449679.4	449156.6	Q =536409
S_i	36276.96	48570.16	1260.96	96.96	1047.36	524.56	S_T =87776.96

in particular neighborhood structures. The CA model consists of the following parts: cells, cell space, cell states, neighborhood, and transition rule discrete time. The basic operational principle of CA is that, for a given time t, the next state of a cell S^{t+1} is the comprehensive result affected by its previous state S^t and the previous states of other neighborhoods N^t, as shown in (5):

$$S^{t+1} = f(S^t, N^t) \tag{5}$$

Since the concept of CA was first proposed, it has become a powerful tool to conduct scientific research in different field, including research in meta-heuristics. Alba et al. [40, 41], proposed cellular GAs, Shi et al. [42] proposed a Cellular PSO (CPSO) algorithm and cellular HS algorithms have also been proposed [43, 44]. In these cellular algorithms, the CA model was employed to study the swarm system and consider algorithm iteration in a CA way, where each individual exchanges information within its neighborhood before changing its current state. However, interactions within the swarm have limitations, in that only one part of search space can be explored. To improve the search capability, individuals should be endued with more talent to free themselves from neighborhood interactions to exploit more potential search space outside the swarm. This idea has led to the proposed cellular local search (CLS).

Table 4 Analysis of variance

Variable source	Squares	Freedom	Mean Square	F value	Significance
HMS (A)	36276.96	4	9069.24	69.157	*
HMCR (B)	48570.16	4	12142.54	92.592	*
PAR (C)	1260.96	4	315.24	2.404	
M (D)	96.96	4	24.24	0.185	
F (E)	1047.36	4	261.84	1.997	
Error	524.56	4	131.14		
Total	87776.96	24			

In CLS, we extend a generalized cellular automata model to guide the local search of harmonies in the current harmony memory and thus enhance their exploitation capability. At first, the whole search space is assumed to be the cell space, and we can imagine that the cell space is cut by infinite virtual grids, where the cell space could not be further subdivided. Every grid in the search space represents a single solution. To distinguish current harmonies and potential candidate solutions, we define a "smart-cell" as a harmony in the current harmony memory. In contrast, a cell that is not "smart" is one that represents a candidate solution in the search space that is not visited by any of the harmonies in the current harmony memory. The comparison of the basic idea between cellular HS and cellular local search in MHS is illustrated in Fig. 4. As we can see in Fig. 4a, the harmonies in the harmony memory pool are mapped onto a grid structure so that harmonies can interact with their neighborhood; however, in the proposed cellular local search (Fig. 4b), harmonies interact with potential candidate solutions outside the swarm.

The CA model for the proposed cellular local search is as follows:

1. Cell: current harmonies in harmony memory.
2. Cell space: the search space.
3. Cell state: the harmony's current position X_i^t and current best harmony X_g^t, $S_i^t = \left\{X_i^t, X_g^t\right\}$.
4. Neighborhood: a set of potential candidate solutions based on some type of lattice structure, defined by neighborhood functions. For details of neighborhood functions, see Eq. (7).
5. Transition rule:

$$S_i^{t+1} = f\left(S_i^t \cup S_{N(i)}^t\right) = f\left(S_i^t, S_{i+\delta_1}^t, ..., S_{i+\delta_l}^t\right) \\ (fitness(S_i^t), fitness(S_{i+\delta_1}^t), ..., fitness(S_{i+\delta_l}^t)) \tag{6}$$

where l is the size of the neighborhood, we set l =5, and δ is the index of the neighborhood, $\delta \in [1, l]$.

1. Discrete time step: iteration in MHS.

Every smart-cell constructs its neighborhood with a neighborhood function as follows:

$$N(i) = \begin{cases} X_i^t + \frac{fitness(X_g^t)}{fitness(X_i^t)} R_3 \circ \gamma_i^t & \neq 0, fitness(X_g^t) \geq 0 \\ X_i^t + \left|\frac{fitness(X_i^t)}{fitness(X_g^t)}\right| R_3 \circ \gamma_i^t & (X_i^t) \neq 0, fitness(X_g^t) \leq 0 \\ X_i^t + \left(\frac{e^{fitness(X_g^t)}}{e^{fitness(X_i^t)}}\right)^2 R_3 \circ \gamma_i^t & fitness(X_i^t)=0, fitness(X_g^t) \geq 0 \\ X_i^t + \left(\frac{e^{fitness(X_g^t)}}{e^{fitness(X_i^t)}}\right)^2 R_3 \circ \gamma_i^t & fitness(X_i^t)=0, fitness(X_g^t) < 0 \end{cases} \tag{7}$$

where R_3 is a1 $\times$ d matrix composed by d uniform random numbers in[-1, 1], and "$\circ$" is the operation symbol of the Hadamard product. For the jth dimension (j =1,2,…,d) of X_i^t, $N(i)$ generates random points within a specific radius from x_{ij}^t.

In general, Eq. (7) can be simplified as

$$N(i) = X_i^t + \xi_i^t \circ \gamma_i^t \tag{8}$$

where ξ_i^t is the radius ratio with a range of (-1,1) and γ_i^t is the search radius (or step size). In this research, the value of γ_i^t is based on Lévy flights [8, 45]. Lévy flights are random walks in which the steps are defined in terms of the step-lengths, which have a certain probability distribution, with the directions of the steps being isotropic and random. They are very useful in simulations for random or pseudo random natural phenomena. Experiments have pointed out that Lévy flights are efficient in exploring an unknown, large-scale search space [46]. A simple scheme discussed in detail by Yang et al. can be summarized as

$$\gamma_i^t = L\acute{e}vy(\beta) \sim 0.01 \frac{\mu}{|\nu|^{1/\beta}} (X_i^t - X_g^t) \tag{9}$$

where μ and ν are drawn from normal distributions. That is,

$$\mu \sim N(0, \sigma_u^2), \nu \sim N(0, \sigma_v^2) \tag{10}$$

with $\sigma_\mu = \left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}}\right\}^{1/\beta}$, $\sigma_\nu = 1$. Here, Γ is the standard Gamma function.

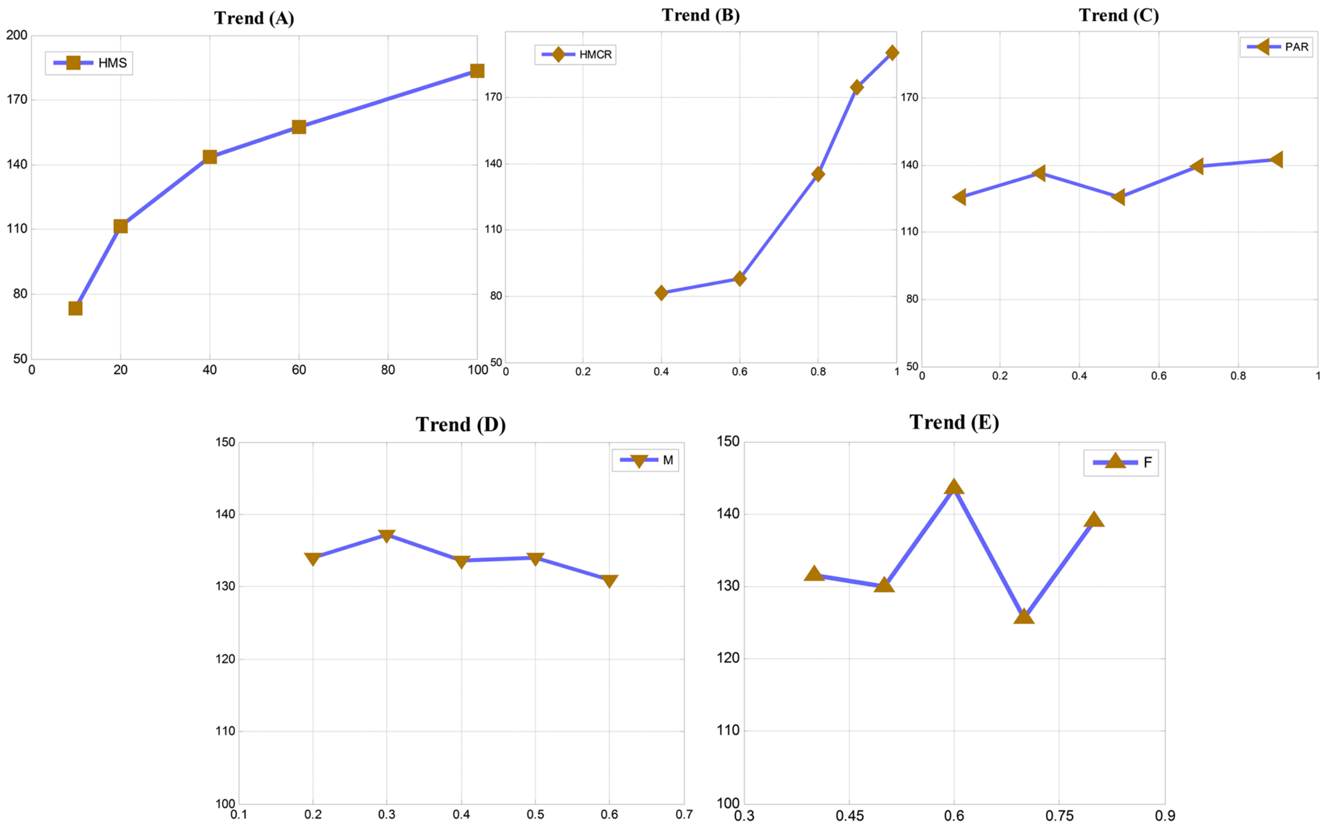


Fig. 6 Trends of the average scores

3.3 Working steps of MHS algorithm

The working steps of the proposed MHS algorithm are as follows:

Step 1: Initialization.

First, the parameters in MHS are specified: the harmony memory size (*HMS*), the harmony memory consideration rate (*HMCR*), the pitch adjusting rate (*PAR*), and the max iteration number (*NI*), the proportion of the better part in harmony memories $M(0 < M < 1)$, and the

Table 5 Parameter settings of the compared HS algorithms

Algorithm	*HMS*	*HMCR*	*PAR*	*BW*	*LP*	P_m	*M*	*F*	*ts*	I_r	F_m	R_m
HS	5	0.9	0.3	0.01	–	–	–	–	–	–	–	–
IHS	5	0.9	$PAR_{min} = 0.01$	$BW_{min} = 0.0001$	–	–	–	–	–	–	–	–
			$PAR_{max} = 0.99$	$BW_{max} = 0.05^*(UB-LB)$								
GHS	5	0.9	$PAR_{min} = 0.01$	–	–	–	–	–	–	–	–	–
			$PAR_{max} = 0.99$									
SGHS	5	0.98	0.9	$BW_{min} = 0.0005$	100	–	–	–	–	–	–	–
		(Initial value)	(Initial value)	$BW_{max} = 0.10^*(UB-LB)$								
NGHS	5	–	–	–	–	0.005	–	–	–	–	–	–
LAHS	5	$HMCR_{min} = 0.55$	$PAR_{min} = 0$	$BW_{min} = 0$	–	–	–	–	–	–	–	–
		$HMCR_{max} = 1$	$PAR_{max} = 1$	$BW_{max} = 1$								
GSHS	7	0.9	0.7	–	–	–	–	–	–	–	–	–
i-HS	100	0.98	0.3	0.01	–	–	–	–	–	10	600	0.2
MHS	100	0.99	0.9	–	–	–	0.3	0.6	–	–	–	–

Table 6 The optimization results of all variants of HS for test functions f_1-f_{19} (D=30)

Functions		HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS	MHS
f_1	Best	3.404800E + 00	1.936221E − 01	0.000000E + 00	5.965786E − 09	3.461213E − 15	4.870240E − 05	1.007831E − 16	1.148437E + 00	2.597973E − 26
	Worst	6.806400E + 00	3.546566E + 01	5.930546E + 00	1.899329E − 08	2.909419E − 12	2.860152E − 04	1.083726E − 10	3.639500E + 00	1.160016E − 24
	Mean	5.218027E + 00	5.503993E + 00	1.453843E + 00	1.100729E − 08	2.194050E − 13	1.382487E − 04	6.116801E − 12	2.120597E + 00	1.630832E − 25
	Std	9.023945E − 01	7.424841E + 00	1.751049E + 00	3.067727E − 09	5.331643E − 13	6.764053E − 05	1.973134E − 11	5.351317E − 01	2.182091E − 25
f_2	Best	6.480000E − 01	8.448786E − 03	1.068547E − 04	2.575323E − 04	5.280912E − 09	2.167937E − 02	0.000000E + 00	1.662800E − 01	8.808440E − 15
	Worst	9.480000E − 01	2.545121E + 00	4.358409E − 01	4.972895E − 04	2.133828E − 07	2.479073E − 01	3.901615E − 08	2.848000E − 01	9.455257E − 14
	Mean	8.037333E − 01	2.583149E − 01	1.101754E − 01	3.746567E − 04	3.694963E − 08	6.798270E − 02	1.884905E − 09	2.105840E − 01	3.402760E − 14
	Std	8.595151E − 02	4.509888E − 01	1.120621E − 01	5.712998E − 05	3.969181E − 08	6.347886E − 02	7.025549E − 09	2.704212E − 02	2.011058E − 14
f_3	Best	1.279664E + 02	6.340934E + 02	4.061933E + 02	2.433682E + 00	3.617152E + 00	2.359465E + 02	3.442802E − 02	3.027997E + 03	3.227410E − 04
	Worst	6.991072E + 02	2.330522E + 04	3.141696E + 03	1.393169E + 01	1.290869E + 02	2.515780E + 03	2.489537E − 01	1.934137E + 04	6.196566E − 02
	Mean	3.039050E + 02	3.062289E + 03	9.859893E + 02	5.533018E + 00	3.811219E + 01	9.120883E + 02	1.424409E − 01	1.322987E + 04	1.056694E − 02
	Std	1.348160E + 02	4.314989E + 03	5.205985E + 02	2.642203E + 00	2.790709E + 01	5.164950E + 02	6.409992E − 02	3.690841E + 03	1.453778E − 02
f_4	Best	3.000000E + 00	6.000000E + 00	0.000000E + 00	0.000000E + 00	0.000000E + 00	8.000000E + 00	0.000000E + 00	0.000000E + 00	0.000000E + 00
	Worst	7.000000E + 00	5.800000E + 01	1.200000E + 01	0.000000E + 00	0.000000E + 00	3.200000E + 01	0.000000E + 00	4.000000E + 00	0.000000E + 00
	Mean	4.366667E + 00	2.360000E + 01	2.766667E + 00	0.000000E + 00	0.000000E + 00	1.893333E + 01	0.000000E + 00	6.666667E − 01	0.000000E + 00
	Std	9.826269E − 01	1.316713E + 01	3.221628E + 00	0.000000E + 00	0.000000E + 00	6.845599E + 00	0.000000E + 00	1.490712E + 00	0.000000E + 00
f_5	Best	4.628153E − 07	1.947683E − 11	3.056527E − 23	1.648249E − 17	5.706260E − 36	9.746667E − 10	2.652640E − 29	2.150636E − 07	1.448028E − 47
	Worst	1.244890E − 06	9.638381E − 06	1.135811E − 05	5.041992E − 16	1.162896E − 28	8.393735E − 09	5.527653E − 24	1.467747E − 06	3.171475E − 43
	Mean	8.440023E − 07	9.666567E − 07	1.445517E − 06	1.359170E − 16	3.880182E − 30	3.357656E − 09	4.330169E − 25	7.699376E − 07	2.180890E − 44
	Std	2.205555E − 07	2.078143E − 06	3.253533E − 06	1.253110E − 16	2.087391E − 29	1.737814E − 09	1.121934E − 24	2.715454E − 07	6.077528E − 44
f_6	Best	6.343655E + 01	1.341168E + 02	1.138574E + 02	1.549004E − 02	1.302374E − 02	2.812191E + 01	2.703937E + 01	1.470885E + 02	1.558092E + 01
	Worst	3.062906E + 02	1.081309E + 03	4.152855E + 02	2.453067E + 02	8.019087E + 01	1.443904E + 02	1.725964E + 02	3.956137E + 02	7.767783E + 01
	Mean	1.904024E + 02	3.733928E + 02	2.614162E + 02	6.675450E + 01	2.832640E + 01	9.184528E + 01	7.241998E + 01	2.490856E + 02	2.733222E + 01
	Std	5.163731E + 01	2.174285E + 02	7.514824E + 01	5.536235E + 01	3.192124E + 01	2.146787E + 01	2.891477E + 01	6.396663E + 01	1.333224E + 01
f_7	Best	9.175499E + 00	2.801764E + 00	1.051774E + 00	8.291705E − 07	8.269884E − 07	4.512588E + 00	3.285086E + 00	1.199003E + 00	8.269870E − 07
	Worst	2.172527E + 01	6.150477E + 01	2.868081E + 01	5.409530E − 03	8.279515E − 07	3.360755E + 01	6.876877E + 00	5.320676E + 00	1.184383E + 02
	Mean	1.588279E + 01	1.167570E + 01	7.404515E + 00	3.399120E − 04	8.270292E − 07	1.557372E + 01	5.441912E + 00	2.762588E + 00	1.174643E + 01
	Std	2.885507E + 00	1.252397E + 01	6.234729E + 00	1.114855E − 03	1.717255E − 10	8.210903E + 00	8.529730E − 01	9.713337E − 01	3.399427E + 01

Table 6 (continued)

Functions		HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS	MHS
f_8	Best	1.047518E+01	1.172684E−02	5.544348E−02	1.072471E−06	1.065814E−14	2.010508E+00	0.000000E+00	3.270314E−01	1.776357E−14
	Worst	2.375497E+01	3.275890E+01	7.806198E+00	1.000168E+00	9.841372E−11	1.295212E+01	2.354614E−06	1.544239E+00	2.984877E+00
	Mean	1.692476E+01	5.057350E+00	2.709819E+00	1.334103E−01	8.494361E−12	6.431961E+00	1.249041E−07	5.870276E−01	8.014047E−01
	Std	2.661545E+00	5.432708E+00	1.828345E+00	3.384536E−01	2.293000E−11	2.632535E+00	4.767224E−07	2.693198E−01	8.302905E−01
f_9	Best	8.347615E−01	7.225298E−02	6.788045E−03	5.170091E−05	5.965251E−09	9.698686E−03	1.189168E−09	2.339653E−01	4.618528E−14
	Worst	1.432403E+00	1.377288E+00	1.206989E+00	9.955321E−05	6.698767E−07	1.778707E+00	2.205013E−04	4.396209E−01	1.350031E−13
	Mean	1.094557E+00	8.949257E−01	4.875226E−01	7.515962E−05	9.208034E−08	1.283391E+00	8.507823E−06	2.944222E−01	7.709389E−14
	Std	1.363812E−01	2.810178E−01	3.821948E−01	9.884683E−06	1.282798E−07	3.657520E−01	3.942360E−05	5.034005E−02	2.372883E−14
f_{10}	Best	1.023399E+00	2.134082E−01	1.468715E−01	1.146161E−02	3.252953E−14	1.709817E−01	2.574607E−13	8.631900E−01	0.000000E+00
	Worst	1.060555E+00	1.759025E+00	1.109674E+00	1.480611E−01	2.028751E−01	1.170270E+00	4.435253E−02	1.028799E+00	1.969736E−02
	Mean	1.043853E+00	9.514587E−01	5.969689E−01	5.969829E−02	6.309198E−02	6.894918E−01	1.227086E−02	9.912304E−01	2.464035E−03
	Std	9.641044E−03	2.690895E−01	2.597000E−01	3.559623E−02	5.279860E−02	2.606328E−01	1.338829E−02	3.741751E−02	4.885642E−03
f_{11}	Best	2.576945E+01	1.286131E+01	0.000000E+00	1.087125E+00	4.854318E−05	2.392698E+01	0.000000E+00	9.741078E+00	2.519920E−06
	Worst	3.691112E+01	5.667286E+01	6.058007E+00	3.179990E+00	7.461037E−03	4.371500E+01	1.073692E−01	1.350285E+01	9.566887E−05
	Mean	3.222415E+01	2.708650E+01	1.495794E+00	1.991843E+00	8.709712E−04	3.197743E+01	7.886364E−03	1.139763E+01	3.324608E−05
	Std	2.785502E+00	1.191685E+01	1.582880E+00	5.559638E−01	1.299375E−03	4.741514E+00	2.136919E−02	8.919429E−01	2.701344E−05
f_{12}	Best	7.819978E−02	1.915258E−01	1.782230E−01	7.818918E−02	1.782223E−01	2.276901E−01	2.276901E−01	1.269905E−01	1.048571E−02
	Worst	2.276902E−01	3.455064E−01	3.732906E−01	3.121031E−01	3.960979E−01	3.455063E−01	3.960979E−01	2.277015E−01	3.722408E−02
	Mean	1.532438E−01	2.859001E−01	2.749659E−01	1.621413E−01	3.115691E−01	3.035745E−01	3.216333E−01	1.779554E−01	3.633280E−02
	Std	3.100146E−02	4.012105E−02	4.342179E−02	5.205542E−02	5.307844E−02	3.412698E−02	4.300631E−02	2.916114E−02	4.799684E−03
f_{13}	Best	1.976647E−02	5.846653E−03	1.495375E−06	3.899854E−11	2.733936E−18	1.499107E−06	1.470719E−06	1.693917E−03	2.245049E−25
	Worst	1.731311E+00	1.879927E−01	2.615483E−02	1.716870E−10	5.275258E−15	1.095961E−01	2.559175E−06	1.942228E−02	1.965506E−22
	Mean	2.338642E−01	3.859178E−02	5.995259E−03	9.521915E−11	5.597858E−16	8.164767E−03	1.939177E−06	4.880367E−03	2.335771E−23
	Std	4.068137E−01	4.029668E−02	6.211114E−03	3.773285E−11	1.196013E−15	2.653841E−02	2.765353E−07	3.645486E−03	3.823297E−23
f_{14}	Best	−4.468480E+02	−4.497929E+02	−4.500000E+02	−4.500000E+02	−4.500000E+02	−4.499999E+02	−4.495562E+02	−4.486389E+02	−4.500000E+02
	Worst	−4.429680E+02	−4.283388E+02	−4.319300E+02	−4.500000E+02	−4.500000E+02	−4.499989E+02	−4.492400E+02	−4.456455E+02	−4.500000E+02
	Mean	−4.449211E+02	−4.419836E+02	−4.475010E+02	−4.500000E+02	−4.500000E+02	−4.499997E+02	−4.493938E+02	−4.476079E+02	−4.500000E+02
	Std	9.352252E−01	6.199950E+00	3.378080E+00	3.840824E−09	3.146135E−13	2.218050E−04	7.767486E−02	6.256843E−01	0.000000E+00

Table 6 (continued)

Functions		HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS	MHS
f_{15}	Best	$8.645376E+02$	$4.118840E+03$	$3.896641E+03$	$-4.465699E+02$	$-3.819638E+02$	$6.388968E+03$	$1.058445E+03$	$9.745316E+03$	$-4.499904E+02$
	Worst	$3.307125E+03$	$1.681056E+04$	$1.369440E+04$	$-4.288003E+02$	$-9.874774E+01$	$1.650513E+04$	$6.827261E+03$	$2.632732E+04$	$-4.483044E+02$
	Mean	$2.264740E+03$	$1.089936E+04$	$8.924759E+03$	$-4.406143E+02$	$-2.549591E+02$	$1.270345E+04$	$4.525013E+03$	$1.861428E+04$	$-4.496983E+02$
	Std	$5.496460E+02$	$2.471470E+03$	$2.434720E+03$	$4.460475E+00$	$6.601617E+01$	$1.920343E+03$	$1.541904E+03$	$4.538971E+03$	$3.325013E-01$
f_{16}	Best	$1.231972E+03$	$3.606522E+03$	$1.473737E+03$	$4.109582E+02$	$3.900003E+02$	$4.713843E+02$	$4.682129E+02$	$1.012648E+03$	$4.021389E+02$
	Worst	$4.957835E+03$	$1.533925E+04$	$5.967187E+03$	$7.592892E+02$	$2.504951E+03$	$2.844368E+03$	$1.208090E+03$	$9.607206E+03$	$4.617642E+02$
	Mean	$2.012449E+03$	$7.659714E+03$	$3.664320E+03$	$4.824504E+02$	$4.730135E+02$	$9.396148E+02$	$6.591399E+02$	$2.942594E+03$	$4.152372E+02$
	Std	$8.862394E+02$	$2.906701E+03$	$1.131660E+03$	$7.453995E+01$	$3.780569E+02$	$6.151350E+02$	$2.368735E+02$	$1.954987E+03$	$9.054748E+00$
f_{17}	Best	$-2.602843E+02$	$-3.231492E+02$	$-3.300000E+02$	$-3.230353E+02$	$-3.300000E+02$	$-3.219821E+02$	$-1.517155E+02$	$-3.012806E+02$	$-3.300000E+02$
	Worst	$-2.091214E+02$	$-2.931449E+02$	$-3.062845E+02$	$-3.108652E+02$	$-3.290050E+02$	$-2.741720E+02$	$2.024156E+01$	$-2.820550E+02$	$-3.260202E+02$
	Mean	$-2.310323E+02$	$-3.055398E+02$	$-3.217904E+02$	$-3.188125E+02$	$-3.299005E+02$	$-3.071097E+02$	$-7.297800E+01$	$-2.934170E+02$	$-3.286734E+02$
	Std	$1.302454E+01$	$6.681357E+00$	$5.505605E+00$	$3.145665E+00$	$2.984763E-01$	$1.052567E+01$	$4.744662E+01$	$4.507928E+00$	$9.021937E-01$
f_{18}	Best	$-1.799312E+02$	$-1.799267E+02$	$-1.799283E+02$	$-1.800000E+02$	$-1.800000E+02$	$-1.799978E+02$	$-1.799974E+02$	$-1.799902E+02$	$-1.800000E+02$
	Worst	$-1.797905E+02$	$-1.792644E+02$	$-1.795654E+02$	$-1.798428E+02$	$-1.796132E+02$	$-1.797580E+02$	$-1.796311E+02$	$-1.798623E+02$	$-1.799680E+02$
	Mean	$-1.798754E+02$	$-1.797364E+02$	$-1.797420E+02$	$-1.799574E+02$	$-1.799099E+02$	$-1.799366E+02$	$-1.799066E+02$	$-1.799514E+02$	$-1.799969E+02$
	Std	$4.278789E-02$	$1.359743E-01$	$1.040371E-01$	$3.687776E-02$	$1.001942E-01$	$5.815675E-02$	$8.795043E-02$	$3.378940E-02$	$7.042818E-03$
f_{19}	Best	$-1.955908E+02$	$-2.838783E+02$	$-3.299995E+02$	$-3.160705E+02$	$-3.300000E+02$	$-2.936758E+02$	$-1.610415E+02$	$-2.215410E+02$	$-3.300000E+02$
	Worst	$-7.760203E+01$	$-1.473016E+01$	$-2.603676E+02$	$-2.931863E+02$	$-3.290050E+02$	$2.867212E+01$	$-6.050929E+01$	$-1.279593E+02$	$-3.240302E+02$
	Mean	$-1.386363E+02$	$-1.858396E+02$	$-3.029648E+02$	$-3.067934E+02$	$-3.297015E+02$	$-1.346771E+02$	$-1.161564E+02$	$-1.865426E+02$	$-3.285739E+02$
	Std	$2.164090E+01$	$6.304873E+01$	$1.704586E+01$	$5.931706E+00$	$4.559482E-01$	$7.984685E+01$	$2.677927E+01$	$2.125876E+01$	$1.490964E+00$

Table 7 The optimization results of all variants of HS for test functions f_1-f_{19} (D=50)

Functions		HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS	MHS
f_1	Best	2.753120E + 01	8.552865E + 01	4.822538E + 01	1.378976E − 07	2.335274E − 09	9.843889E − 04	6.072350E − 13	6.190708E + 00	7.843839E − 21
	Worst	4.874720E + 01	3.341128E + 02	1.044267E + 02	3.160078E − 07	6.157340E − 08	2.633452E − 02	9.219338E − 10	2.067146E + 01	1.115203E − 19
	Mean	3.645755E + 01	1.899530E + 02	7.498746E + 01	2.188959E − 07	1.509473E − 08	3.609746E − 03	5.101649E − 11	1.375798E + 01	3.457662E − 20
	Std	5.060762E + 00	6.566877E + 01	1.415647E + 01	3.846593E − 08	1.605332E − 08	4.618565E − 03	1.686216E − 10	4.083996E + 00	2.283783E − 20
f_2	Best	2.372000E + 00	3.948663E + 00	2.046183E + 00	1.760248E − 03	8.614734E − 06	8.239562E − 02	5.659121E − 16	6.321600E − 01	7.971164E − 12
	Worst	4.016000E + 00	1.034355E + 01	3.619192E + 00	8.134046E − 03	4.938767E − 05	9.255759E − 01	2.111692E − 08	1.138640E + 00	4.055281E − 11
	Mean	3.084533E + 00	5.467204E + 00	2.871210E + 00	2.832595E − 03	2.194437E − 05	5.320769E − 01	3.110784E − 09	8.585133E − 01	2.235763E − 11
	Std	3.231804E − 01	1.369973E + 00	4.351770E − 01	1.385661E − 03	9.123323E − 06	2.183102E − 01	5.962132E − 09	1.329743E − 01	9.580753E − 12
f_3	Best	1.733576E + 03	1.079910E + 04	2.118571E + 03	3.406932E + 01	1.203549E + 02	2.527651E + 03	1.169599E + 00	5.381107E + 04	2.073472E + 00
	Worst	1.395834E + 04	6.148301E + 04	1.204348E + 04	3.631079E + 02	8.288988E + 02	1.077912E + 04	7.032932E + 00	8.416552E + 04	1.516593E + 01
	Mean	5.053221E + 03	2.591148E + 04	4.846754E + 03	1.509115E + 02	3.556051E + 02	6.387840E + 03	3.547683E + 00	7.077447E + 04	7.368250E + 00
	Std	2.396482E + 03	1.219055E + 04	2.271882E + 03	7.454848E + 01	1.695957E + 02	2.419385E + 03	1.383548E + 00	8.507457E + 03	3.611518E + 00
f_4	Best	3.100000E + 01	1.390000E + 02	4.500000E + 01	0.000000E + 00	0.000000E + 00	4.800000E + 01	0.000000E + 00	4.000000E + 00	0.000000E + 00
	Worst	5.200000E + 01	3.230000E + 02	1.400000E + 02	1.000000E + 00	0.000000E + 00	1.040000E + 02	0.000000E + 00	2.400000E + 01	0.000000E + 00
	Mean	4.090000E + 01	2.301667E + 02	7.993333E + 01	3.333333E − 02	0.000000E + 00	7.533333E + 01	0.000000E + 00	1.306667E + 01	0.000000E + 00
	Std	5.002333E + 00	4.457210E + 01	2.411491E + 01	1.795055E − 01	0.000000E + 00	1.576776E + 01	0.000000E + 00	4.836895E + 00	0.000000E + 00
f_5	Best	5.086354E − 05	2.992787E − 04	8.285875E − 05	1.134510E − 14	5.574024E − 25	3.500180E − 08	2.452524E − 26	1.333560E − 05	9.919628E − 36
	Worst	1.395021E − 04	4.165837E − 03	4.037938E − 04	7.100598E − 14	3.820629E − 22	2.647064E − 07	2.467427E − 23	8.226246E − 05	1.614155E − 31
	Mean	8.350978E − 05	1.174279E − 03	2.003053E − 04	3.142679E − 14	2.854383E − 23	1.175057E − 07	1.753815E − 24	3.872272E − 05	8.791887E − 33
	Std	2.306531E − 05	7.336359E − 04	7.072216E − 05	1.151084E − 14	7.074809E − 23	5.082345E − 08	4.410188E − 24	1.650436E − 05	2.886733E − 32
f_6	Best	5.833980E + 02	1.909173E + 03	7.872266E + 02	2.886260E + 01	1.103678E − 02	9.522701E + 01	3.434628E + 01	6.008167E + 02	3.051051E + 01
	Worst	1.304621E + 03	6.871949E + 03	2.445634E + 03	5.567449E + 02	1.418949E + 02	4.570690E + 02	2.089643E + 02	1.051326E + 03	1.865428E + 02
	Mean	8.649920E + 02	3.709023E + 03	1.291885E + 03	1.137870E + 02	5.554297E + 01	1.676866E + 02	1.110546E + 02	8.318712E + 02	6.591172E + 01
	Std	1.670217E + 02	1.055274E + 03	3.607080E + 02	9.777479E + 01	4.091344E + 01	7.293644E + 01	4.414754E + 01	1.288616E + 02	3.353121E + 01
f_7	Best	7.276449E + 01	3.771998E + 02	2.026417E + 02	1.825801E − 03	1.391270E − 06	6.327896E + 01	1.351052E + 01	6.055316E + 00	1.378315E − 06
	Worst	1.582911E + 02	7.220222E + 02	4.780075E + 02	8.970124E − 02	1.916352E − 06	1.618946E + 02	1.813168E + 01	5.098897E + 01	3.553150E + 02
	Mean	1.133936E + 02	5.341444E + 02	2.923677E + 02	2.976736E − 02	1.522992E − 06	1.158579E + 02	1.574870E + 01	1.982309E + 01	3.553549E + 01
	Std	1.747359E + 01	9.592163E + 01	6.020369E + 01	2.483221E − 02	1.330632E − 07	2.694616E + 01	1.311054E + 00	1.021057E + 01	8.176934E + 01

Table 7 (continued)

Functions		HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS	MHS
f_8	Best	$4.121843E+01$	$3.665012E+01$	$2.715732E+01$	$3.407841E-05$	$6.545198E-09$	$1.322332E+01$	$6.039613E-14$	$1.755267E+00$	$9.949593E-01$
	Worst	$6.282328E+01$	$7.547888E+01$	$5.969465E+01$	$8.119927E-04$	$2.056395E-06$	$2.716124E+01$	$7.124820E-06$	$5.894076E+00$	$1.293446E+01$
	Mean	$5.263601E+01$	$5.072894E+01$	$4.146907E+01$	$7.200791E-05$	$2.049287E-07$	$1.967136E+01$	$2.938061E-07$	$4.090222E+00$	$5.506734E+00$
	Std	$6.446348E+00$	$9.596422E+00$	$6.468948E+00$	$1.377353E-04$	$4.377602E-07$	$3.164700E+00$	$1.287494E-06$	$1.122962E+00$	$2.935431E+00$
f_9	Best	$2.233465E+00$	$2.756344E+00$	$2.077178E+00$	$2.030518E-04$	$6.593152E-06$	$1.565613E+00$	$2.584614E-08$	$5.421619E-01$	$1.727329E-11$
	Worst	$3.014751E+00$	$7.503388E+00$	$3.198397E+00$	$5.145466E-01$	$5.705578E-05$	$2.285823E+00$	$7.716354E-06$	$1.209900E+00$	$9.370638E-11$
	Mean	$2.585350E+00$	$3.680297E+00$	$2.595433E+00$	$3.456350E-02$	$2.034662E-05$	$1.956465E+00$	$1.043120E-06$	$8.596041E-01$	$5.280481E-11$
	Std	$2.131481E-01$	$1.005895E+00$	$2.678623E-01$	$1.220374E-01$	$1.048101E-05$	$1.690791E-01$	$1.647084E-06$	$1.937356E-01$	$1.981233E-11$
f_{10}	Best	$1.212573E+00$	$2.008884E+00$	$1.429350E+00$	$1.815923E-02$	$3.164857E-09$	$1.277914E+00$	$7.498446E-13$	$1.067817E+00$	$0.000000E+00$
	Worst	$1.422899E+00$	$4.216935E+00$	$2.024396E+00$	$7.903281E-02$	$1.332965E-01$	$1.630335E+00$	$3.697374E-02$	$1.207533E+00$	$1.477241E-02$
	Mean	$1.324126E+00$	$2.698403E+00$	$1.716354E+00$	$3.905789E-02$	$4.147289E-02$	$1.441843E+00$	$9.263656E-03$	$1.124583E+00$	$3.039063E-03$
	Std	$4.911436E-02$	$4.662798E-01$	$1.613613E-01$	$1.775773E-02$	$4.114649E-02$	$1.086944E-01$	$1.045821E-02$	$3.080200E-02$	$4.873116E-03$
f_{11}	Best	$1.290406E+02$	$9.773624E+01$	$3.744164E+01$	$1.016341E+01$	$5.858324E-02$	$1.183300E+02$	$0.000000E+00$	$5.478865E+01$	$2.585016E-02$
	Worst	$1.770557E+02$	$2.793970E+02$	$8.972939E+01$	$2.626200E+01$	$1.541761E-01$	$1.923150E+02$	$4.445454E-01$	$6.995222E+01$	$7.267203E-02$
	Mean	$1.552245E+02$	$1.484633E+02$	$6.047741E+01$	$1.781765E+01$	$9.641920E-02$	$1.531987E+02$	$4.730171E-02$	$6.179322E+01$	$4.373959E-02$
	Std	$1.166211E+01$	$3.844992E+01$	$1.221456E+01$	$4.167259E+00$	$2.278448E-02$	$1.562797E+01$	$1.018241E-01$	$3.716636E+00$	$1.065110E-02$
f_{12}	Best	$1.782233E-01$	$3.121533E-01$	$3.121047E-01$	$1.782223E-01$	$3.455063E-01$	$3.732906E-01$	$3.732906E-01$	$2.727411E-01$	$3.722408E-02$
	Worst	$3.285969E-01$	$4.597812E-01$	$4.297229E-01$	$3.121031E-01$	$4.517757E-01$	$4.419083E-01$	$4.419083E-01$	$3.487498E-01$	$7.819103E-02$
	Mean	$2.632932E-01$	$3.862831E-01$	$3.930012E-01$	$2.682631E-01$	$4.006684E-01$	$4.261256E-01$	$4.155122E-01$	$3.017239E-01$	$6.594479E-02$
	Std	$3.918735E-02$	$3.288981E-02$	$2.726693E-02$	$3.335912E-02$	$2.538939E-02$	$1.639583E-02$	$1.973571E-02$	$2.318212E-02$	$1.695766E-02$
f_{13}	Best	$5.705582E-01$	$3.349532E-01$	$1.004608E-01$	$5.757726E-10$	$2.316684E-12$	$1.963040E-03$	$2.586664E-06$	$2.325840E-02$	$2.656303E-18$
	Worst	$1.988877E+00$	$2.038459E+00$	$3.567428E-01$	$1.223264E-09$	$1.415282E-10$	$3.009090E-01$	$3.930851E-03$	$8.179697E-02$	$8.529808E-14$
	Mean	$1.153706E+00$	$7.656105E-01$	$2.315210E-01$	$8.301639E-10$	$2.137842E-11$	$7.300043E-02$	$1.341361E-04$	$4.276407E-02$	$2.988251E-15$
	Std	$4.007592E-01$	$3.557295E-01$	$6.489318E-02$	$1.596434E-10$	$3.010345E-11$	$7.110675E-02$	$7.050324E-04$	$1.468807E-02$	$1.528751E-14$
f_{14}	Best	$-4.205088E+02$	$-3.087133E+02$	$-4.065913E+02$	$-4.500000E+02$	$-4.500000E+02$	$-4.499988E+02$	$-4.487750E+02$	$-4.398475E+02$	$-4.500000E+02$
	Worst	$-3.932096E+02$	$2.851067E+02$	$-3.253833E+02$	$-4.500000E+02$	$-4.500000E+02$	$-4.465600E+02$	$-4.479403E+02$	$-4.208565E+02$	$-4.500000E+02$
	Mean	$-4.078173E+02$	$-1.379640E+02$	$-3.662246E+02$	$-4.500000E+02$	$-4.500000E+02$	$-4.498199E+02$	$-4.483270E+02$	$-4.292350E+02$	$-4.500000E+02$
	Std	$6.808237E+00$	$1.164232E+02$	$2.205965E+01$	$3.302488E-08$	$7.101525E-09$	$6.158927E-01$	$1.749163E-01$	$4.960250E+00$	$1.467691E-14$

Table 7 (continued)

Functions		HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS	MHS
f_{15}	Best	2.002956E + 04	3.300219E + 04	2.396979E + 04	−2.393227E + 02	1.394164E + 02	2.749395E + 04	1.036679E + 04	8.150706E + 04	−3.746328E + 02
	Worst	3.083466E + 04	1.425389E + 05	3.535590E + 04	9.055705E + 01	2.473596E + 03	4.018218E + 04	2.039941E + 04	1.232784E + 05	−5.732375E + 00
	Mean	2.580770E + 04	5.183523E + 04	2.873709E + 04	−1.205647E + 02	9.672928E + 02	3.300141E + 04	1.514338E + 04	1.009972E + 05	−2.146677E + 02
	Std	2.926647E + 03	2.151690E + 04	2.989662E + 03	8.541496E + 01	4.895458E + 02	3.199935E + 03	2.372181E + 03	1.094758E + 04	1.001654E + 02
f_{16}	Best	1.727068E + 04	1.831845E + 05	3.345403E + 04	4.358256E + 02	3.900243E + 02	6.659310E + 02	5.859631E + 02	2.080967E + 04	4.232677E + 02
	Worst	5.594140E + 04	1.603562E + 06	2.434128E + 05	2.952023E + 03	3.471921E + 03	9.377155E + 03	1.374489E + 03	5.699824E + 04	6.654711E + 02
	Mean	3.065851E + 04	6.058109E + 05	8.909101E + 04	7.830484E + 02	7.082558E + 02	1.729015E + 03	8.145463E + 02	3.099749E + 04	4.668990E + 02
	Std	9.619147E + 03	3.080533E + 05	4.109948E + 04	5.750355E + 02	7.223504E + 02	1.690275E + 03	2.418572E + 02	7.103570E + 03	4.721484E + 01
f_{17}	Best	−9.795295E + 00	−8.088757E + 01	−2.038218E + 02	−3.094507E + 02	−3.300000E + 02	−2.698105E + 02	6.262127E + 01	−1.747559E + 02	−3.256078E + 02
	Worst	1.619300E + 02	3.097488E + 02	−8.838029E + 01	−2.876165E + 02	−3.280086E + 02	−1.824155E + 02	2.664181E + 02	−1.187927E + 02	−3.124604E + 02
	Mean	6.781523E + 01	7.678913E + 01	−1.353727E + 02	−3.019768E + 02	−3.291422E + 02	−2.279321E + 02	1.720800E + 02	−1.452079E + 02	−3.201447E + 02
	Std	4.005518E + 01	9.427554E + 01	2.872004E + 01	5.555753E + 00	7.174106E − 01	2.073139E + 01	5.403538E + 01	1.446896E + 01	2.860507E + 00
f_{18}	Best	−1.795928E + 02	−1.788932E + 02	−1.794736E + 02	−1.800000E + 02	−1.800000E + 02	−1.799727E + 02	−1.799946E + 02	−1.798451E + 02	−1.800000E + 02
	Worst	−1.793924E + 02	−1.784939E + 02	−1.788924E + 02	−1.798096E + 02	−1.798518E + 02	−1.797924E + 02	−1.797100E + 02	−1.794883E + 02	−1.799877E + 02
	Mean	−1.795129E + 02	−1.787039E + 02	−1.791794E + 02	−1.799460E + 02	−1.799664E + 02	−1.799098E + 02	−1.799222E + 02	−1.797237E + 02	−1.799961E + 02
	Std	5.834045E − 02	8.811431E − 02	1.589859E − 01	4.784203E − 02	3.753781E − 02	4.425051E − 02	6.595093E − 02	8.142684E − 02	5.497564E − 03
f_{19}	Best	3.442566E + 02	1.128342E + 03	2.692419E + 02	−3.011452E + 02	−3.299997E + 02	1.377953E + 02	4.449340E + 01	4.120218E + 01	−3.240542E + 02
	Worst	6.108416E + 02	5.199258E + 03	1.049253E + 03	−2.245338E + 02	−3.260116E + 02	1.118109E + 03	3.301246E + 02	3.404639E + 02	−3.122820E + 02
	Mean	4.765599E + 02	2.445341E + 03	6.338051E + 02	−2.653050E + 02	−3.283024E + 02	5.074509E + 02	1.244184E + 02	2.149794E + 02	−3.175777E + 02
	Std	6.294108E + 01	8.101627E + 02	1.582231E + 02	1.420566E + 01	1.072236E + 00	2.291012E + 02	5.336749E + 01	6.546093E + 01	2.741681E + 00

mutation parameter F. Then, the initial harmony memory of *HMS* harmonies is randomly generated: $HM = \{X_1, X_2, \ldots, X_{HMS}\}$, with $X_i = \{x_i^1, x_i^2, \ldots, x_i^n\}$, $i = 1, 2, \ldots, HMS$ uniformly distributed in the range $[X_{min}, X_{max}]$.

Step 2: Evaluate each harmony in the harmony memory, $f_{\text{HM}} = \{f(X_1), f(X_2) \ldots f(X_{HMS})\}$.

Rank the harmonies according to their fitness value in ascending order, and then divide the harmonies into two parts with constant coefficient *M*.

Step 3: Improvise new harmonies.

HMS new harmony vectors are improvised in this step. The detailed procedure is given in Section 3.1. If the new harmonies are better than the current harmonies, then replace current harmonies.

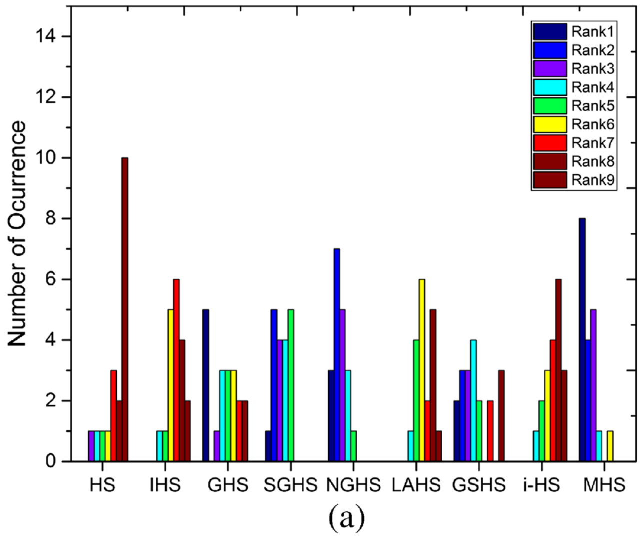
Step4: Cellular local search.

For each of the improvised harmonies, a cellular local search is conducted: see Section 3.2. If the fitness of the neighbors is better than the fitness of the current harmonies in HM, then replace the harmonies with the neighbors. Then, divide the updated harmonies into two parts again.

Step 5: Check the stopping criterion.

The MHS algorithm will repeat step 4 and step 5 until reaching the max iteration number *NI*.

The flowchart of the proposed MHS is shown in Fig. 5.



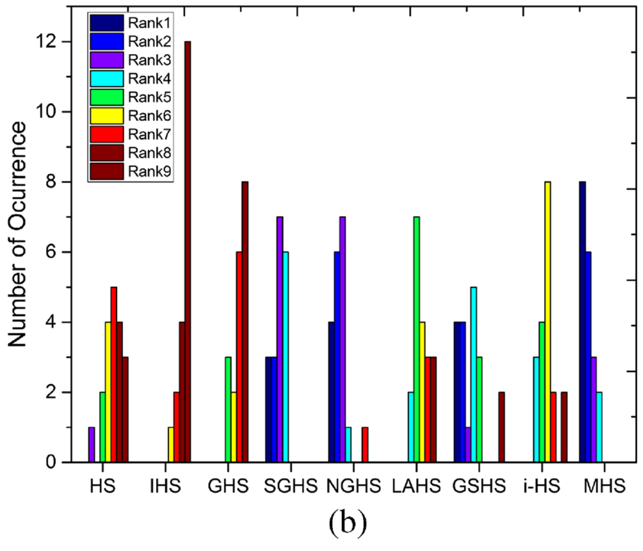


Fig. 7 Histogram of individual best ranks (**a**) 30-dimensional, (**b**) 50-dimensional

4 Experimental results

To evaluate the performance of the proposed MHS algorithm, two comparison tests with different benchmark problems are performed. The first is the comparison of MHS and other HS variants, which uses the benchmark functions shown in Table 1, which have been widely used in previous HS literature. The other is the comparison of MHS and other meta-heuristic algorithms, which uses the CEC2013 benchmark functions. The CEC2013 benchmark functions are more complicated than previous CECbbenchmarks.

This section is organized as follows. A detailed description of the benchmark problems used in the study is described first, followed by the impact of the parameters on the MHS algorithm. Next, a performance evaluation of MHS on the benchmark problems shown in Table 1 are compared with the well-known HS variants. Further, a rank based analysis, statistical comparison analysis and convergence analysis are provided. Last, the performance of MHS and the other meta-heuristics in solving the CEC2013 benchmark problems are compared in this section.

4.1 Benchmark functions

The first set of benchmark problems consists of 19 classical benchmark functions [33, 38, 42, 47] with dimensions of D. Functions $f_1 - f_5$ are unimodal functions, functions $f_6 - f_{13}$ are multimodal functions and $f_{14} - f_{19}$ are shifted or shifted and rotated unimodal and multimodal functions. A brief description of the objective function, search range, and bias value for all 19 functions are shown in Table 1. The shifted global optimum for all of the functions is provided as $\boldsymbol{o} = \{o_1, o_2, \ldots, o_D\}$, and the functions are defined as

Table 8 Rank table for the best values of 30-dimensional and 50-dimensional

Dim	Algo.	Individual ranking of benchmark functions																			Avg. rank
		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}	f_{17}	f_{18}	f_{19}	
30	HS	9	9	5	7	9	6	9	9	9	9	9	3	9	9	4	7	8	7	8	7.63 (9)
	IHS	7	6	8	8	6	8	6	5	7	7	7	7	8	6	7	9	4	9	6	6.89 (7)
	GHS	1	4	7	1	4	7	4	6	5	5	1	6	5	1	6	8	1	8	3	4.37 (4)
	SGHS	5	5	3	2	5	2	3	4	4	4	5	2	3	2	2	3	5	1	4	3.37 (3)
	NGHS	4	3	4	3	2	1	2	2	3	2	4	5	2	3	3	1	2	2	1	2.58 (2)
	LAHS	6	7	6	9	7	5	8	8	6	6	8	8	6	5	8	5	6	4	5	6.47 (6)
	GSHS	3	1	2	4	3	4	7	1	2	3	2	9	4	7	5	4	9	5	9	4.42 (5)
	i-HS	8	8	9	5	8	9	5	7	8	8	6	4	7	8	9	6	7	6	7	7.11 (8)
	MHS	2	2	1	6	1	3	1	3	1	1	3	1	1	4	1	2	3	3	2	2.16 (1)
50	HS	7	8	5	6	7	6	7	9	8	6	9	3	9	7	5	6	8	7	8	6.89 (7)
	IHS	9	9	8	9	9	9	9	8	9	9	7	6	8	9	8	9	7	9	9	8.42 (9)
	GHS	8	7	6	7	8	8	8	7	7	8	5	5	7	8	6	8	5	8	7	7.00 (8)
	SGHS	4	4	3	1	4	2	3	3	4	4	4	2	3	1	2	3	3	1	3	2.84 (3)
	NGHS	3	3	4	2	3	1	2	2	3	3	3	7	2	2	3	1	1	2	1	2.52 (2)
	LAHS	5	5	7	8	5	5	6	6	6	7	8	8	5	4	7	5	4	5	6	5.89 (5)
	GSHS	2	1	1	3	2	4	5	1	2	2	1	9	4	5	4	4	9	4	5	3.58 (4)
	i-HS	6	6	9	5	6	7	4	5	5	5	6	4	6	6	9	7	6	6	4	5.89 (5)
	MHS	1	2	2	4	1	3	1	4	1	1	2	1	1	3	1	2	2	3	2	1.95 (1)

Table 9 Rank table for the worst values of 30-dimensional and 50-dimensional

Dim	Algo.	Individual ranking of benchmark functions																			Avg. rank
		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}	f_{17}	f_{18}	f_{19}	
30	HS	8	8	5	6	6	6	5	8	8	6	7	2	9	7	4	6	8	4	6	6.26 (6)
	IHS	9	9	9	9	8	9	8	9	7	9	9	6	8	9	8	9	5	9	8	8.26 (9)
	GHS	7	7	7	7	9	8	6	6	6	7	5	7	6	8	6	7	4	8	4	6.58 (8)
	SGHS	4	4	3	1	4	5	2	3	3	3	4	4	3	1	2	2	3	3	3	3.00 (3)
	NGHS	2	3	4	2	2	2	1	1	2	4	2	8	2	2	3	4	1	7	1	2.79 (2)
	LAHS	5	5	6	8	5	3	7	7	9	8	8	5	7	4	7	5	7	5	9	6.32 (7)
	GSHS	3	2	2	3	3	4	4	2	4	2	3	9	4	5	5	3	9	6	7	4.21 (4)
	i-HS	6	6	8	5	7	7	3	4	5	5	6	3	5	6	9	8	6	2	5	5.58 (5)
	MHS	1	1	1	4	1	1	9	5	1	1	1	1	1	3	1	1	2	1	2	2.00 (1)
50	HS	7	8	7	6	7	7	5	8	7	6	7	3	8	7	5	6	7	7	6	6.53 (7)
	IHS	9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	8.95 (9)
	GHS	8	7	6	8	8	8	8	7	8	8	6	5	7	8	6	8	6	8	7	7.21 (8)
	SGHS	4	4	3	4	4	5	2	3	4	3	4	2	3	1	2	3	3	3	3	3.16 (3)
	NGHS	3	3	4	1	3	1	1	1	3	4	2	8	2	2	3	4	1	2	1	2.58 (2)
	LAHS	5	5	5	7	5	4	6	6	6	7	8	6	6	5	7	5	4	4	8	5.74 (6)
	GSHS	2	2	1	2	2	3	3	2	2	2	3	7	4	4	4	2	8	5	4	3.26 (4)
	i-HS	6	6	9	5	6	6	4	4	5	5	5	4	5	6	8	7	5	6	5	5.63 (5)
	MHS	1	1	2	3	1	2	7	5	1	1	1	1	1	3	1	1	2	1	2	1.95 (1)

$z = x - o$ for shifted functions and $z = (x - o) * M$ for shifted and rotated functions, where M is the transformation matrix for the rotating matrices.

The second set of benchmark problems consists of 28 CEC2013 [48] functions with dimensions of D. Functions $F_1 - F_5$ are unimodal functions, functions $F_6 - F_{20}$ are basic multimodal functions and $F_{21} - F_{28}$ are composition functions.

4.2 Comparison of MHS and other HS variants

4.2.1 Study and effects of parameters in MHS

Before the performance comparison of MHS and other HS variants, it is first necessary to analyze how the five parameters *HMS*, *HMCR*, *PAR*, *M* and *F* affect the performance of MHS, and then to search for an optimum combination of them.

In this section, an analysis of these five parameters based on the orthogonal experimental design (OED) method is investigated. The OED method was first introduced by Taguchi [49]. This simplified design method has since been applied widely and rapidly all over the world. OED takes advantage of the two characteristics of an orthogonal array, namely, uniform distribution and regular comparability, thus ensuring that the representative experimental points are uniformly scattered over the design region and simultaneously providing mathematical tractability to the experimental data. Therefore, the OED method can make a comprehensive survey of the influence on experimental results due not only to the individual factors but also to the interaction effects between factors.

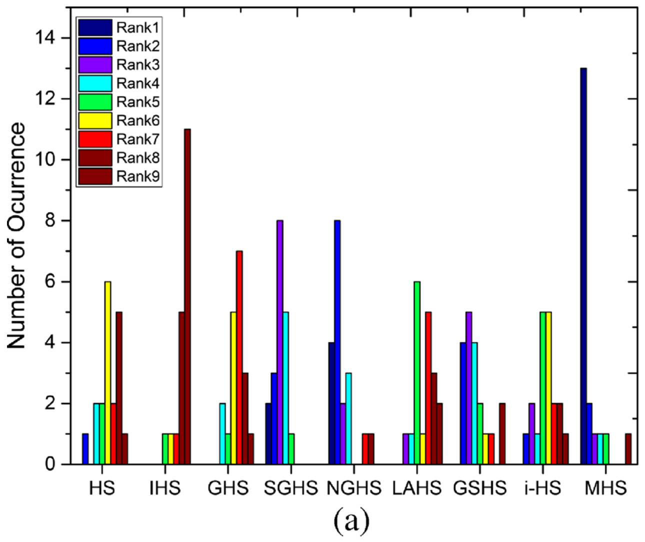
The OED method arranges experiments using an orthogonal array. The orthogonal array is represented as $L_a(b^c)$, in which L is the orthogonal array, a is the number of experiments, b is the level of factors, and c is the number of factors or the number of columns. For example, consider the orthogonal array $L_9(3^4)$, which represents nine experiments.

Considering the advantages of OED method, we apply it to our parameter analysis. Before the experiments begin, we assume that each of the five parameters, has five different level values. Detailed information on all of the factors is shown in Table 2.

After the factors and their levels are determined, an appropriate orthogonal table can be chosen by considering the number of interactions between the factors from different levels. Because there are five influence factors and five level values for each factor, $L_{25}(5^6)$ is chosen from the orthogonal table to arrange the orthogonal design. To obtain more convincing experimental results, an evaluation

Table 10 Rank table for the mean values of 30-dimensional and 50-dimensional

Dim	Algo.	Individual ranking of benchmark functions																			Avg. rank
		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}	f_{17}	f_{18}	f_{19}	
30	HS	8	9	5	7	7	6	9	9	8	9	9	2	9	8	4	6	8	7	7	7.21 (8)
	IHS	9	8	8	9	8	9	6	7	7	7	7	6	8	9	7	9	6	9	6	7.63 (9)
	GHS	6	6	7	6	9	8	5	6	6	5	4	5	6	7	6	8	3	8	4	6.05 (6)
	SGHS	4	4	3	1	4	3	2	3	4	3	5	3	3	1	2	3	4	2	3	3.00 (3)
	NGHS	2	3	4	2	2	2	1	1	2	4	2	8	2	2	3	2	1	5	1	2.58 (2)
	LAHS	5	5	6	8	5	5	8	8	9	6	8	7	7	4	8	5	5	4	8	6.37 (7)
	GSHS	3	2	2	3	3	4	4	2	3	2	3	9	4	5	5	4	9	6	9	4.32 (4)
	i-HS	7	7	9	5	6	7	3	4	5	8	6	4	5	6	9	7	7	3	5	5.95 (5)
	MHS	1	1	1	4	1	1	7	5	1	1	1	1	1	3	1	1	2	1	2	1.89 (1)
50	HS	7	8	6	6	7	7	6	9	7	6	9	2	9	7	5	6	7	7	6	6.68 (7)
	IHS	9	9	8	9	9	9	9	8	9	9	7	5	8	9	8	9	8	9	9	8.42 (9)
	GHS	8	7	5	8	8	8	8	7	8	8	5	6	7	8	6	8	6	8	8	7.21 (8)
	SGHS	4	4	3	4	4	4	2	3	4	3	4	3	3	1	2	3	3	3	3	3.16 (3)
	NGHS	3	3	4	1	3	1	1	1	3	4	3	7	2	2	3	2	1	2	1	2.47 (2)
	LAHS	5	5	7	7	5	5	7	6	6	7	8	9	6	4	7	5	4	5	7	6.05 (6)
	GSHS	2	2	1	2	2	3	3	2	2	2	2	8	4	5	4	4	9	4	4	3.42 (4)
	i-HS	6	6	9	5	6	6	4	4	5	5	6	4	5	6	9	7	5	6	5	5.74 (5)
	MHS	1	1	2	3	1	2	5	5	1	1	1	1	1	3	1	1	2	1	2	1.84 (1)



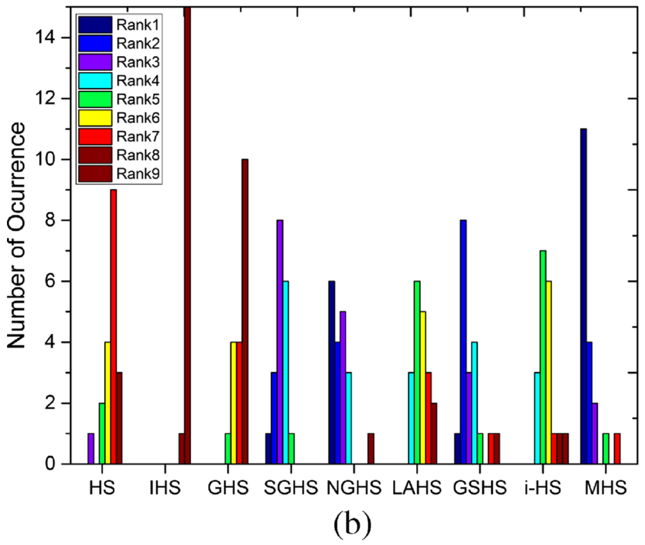


Fig. 8 Histogram of individual worst ranks (**a**) 30-dimensional, (**b**) 50-dimensional

criterion is proposed to calculate the total score: ten benchmark functions with different characteristics, for example, unimodal or multimodal, are chosen from the test functions in Table 1. Each benchmark function earns a score s_i in the range of [1, 25] based on its mean fitness of 50 runs. Finally, the total scores are calculated as follow:

$$T = \sum_{i=1}^{10} \omega_i \cdot s_i \tag{11}$$

The orthogonal experimental results are shown in Table 3, where K_i is the sum of the total score at the i level. The statistics are calculated by the following equations:

$$K = \sum_{i=1}^{25} T_i \tag{12}$$

$$P = \frac{1}{n} \cdot K^2 \tag{13}$$

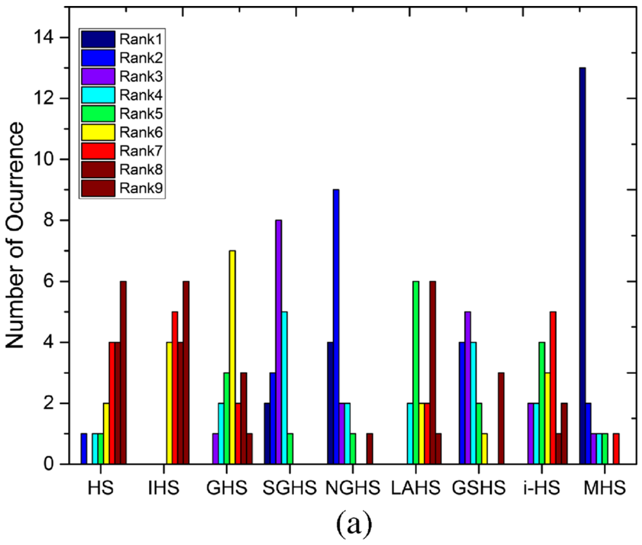
$$Q_i = \frac{1}{5} \cdot \sum_{i=1}^{5} K_i \tag{14}$$

$$S_i = Q_i - P \tag{15}$$

$$Q = \sum_{i=1}^{25} T_i^2 \tag{16}$$

$$S_T = Q - P \tag{17}$$

The results of the analysis of variance are listed in Table 4. $F_A \sim F_E$ obey the F distribution with freedom as



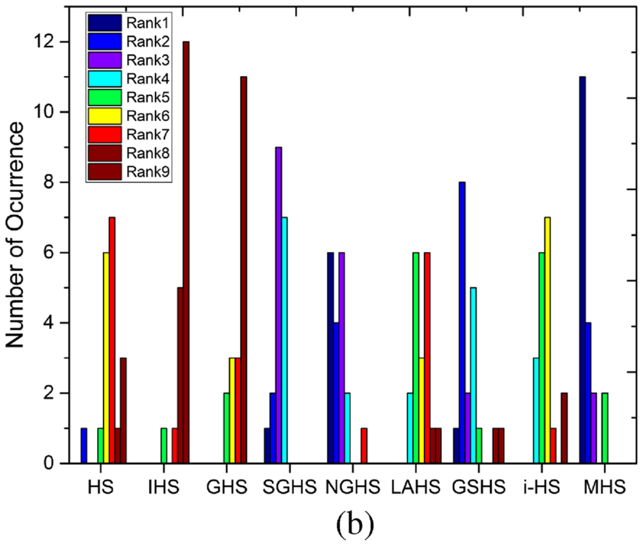


Fig. 9 Histogram of individual mean ranks (**a**) 30-dimensional, (**b**) 50-dimensional

(4, 4). For a specified significance level α, we can believe that factor A has a significant effect on the experimental results with probability $1-\alpha$ if $F_A > F_{1-\alpha}$. From Rice[50], we can find $F_{0.90}(4,4) = 4.11$, $F_{0.95}(4,4) = 6.39$ and $F_{0.99}(4,4) = 15.98$, so it is easy to conclude that *HMS* and *HMCR* are the two main factors with significant effects on the behavior of MHS algorithm. In contrast, *PAR*, *M* and *F* are subordinate factors that affect the search ability of the MHS algorithm less significantly. The trends of the average score are shown in Fig. 6: for the average score, the bigger the better. From the result that $A_5 > A_4 > A_3 > A_2 > A_1$ for factor A, $B_5 > B_4 > B_3 > B_2 > B_1$ for factor B, $C_5 > C_4 > C_2 > C_3 > C_1$ for factor C, $D_2 > D_1 > D_4 > D_3 > D_5$ for factor D, and $E_3 > E_5 > E_1 > E_2 > E_4$ for factor E, we can obtain an optimal combination of these five parameters, $A_5B_5C_5D_2E_3$ in Table 2, namely $HMS = 100$, $HMCR = 0.99$, $PAR = 0.90$, $M = 0.30$ and $F = 0.60$.

4.2.2 Computational results

This section focuses on comparing MHS and other HS variants in solving the first set of benchmark problems. These HS variants include not only the well-known HS variants, such as HS [11], IHS [12], GHS [24], SGHS [33] and NGHS [28], but also some new HS variants, such as LAHS [36], GSHS [51] and Island-based HS (i-HS) [32]. All of the listed HS variants were coded in C++ and implemented on a computer with a 2.3 GHz CPU and 4GB RAM. The detailed settings of the parameters in these algorithms are shown in Table 5.

In these experiments, each benchmark problem runs 30 independent replications. The maximum number of *FEs* is set to 6.0×10^4 for 30-D functions and 1.0×10^5 for 50-D functions. The optimization results for functions with 30-D functions and 50-D functions are reported in Tables 6-7, respectively. Here, “Best”, “Worst” “Mean” and “Std” are the best results, worst results, average values over 30 replications and the corresponding standard deviations, respectively.

From Table 6, one can observe that MHS performs better than the other HS variants. MHS has provided better mean, best and worst performances and has outperformed all of the other algorithms by a significant margin. For unimodal functions $f_1 - f_5$, most of the HS variants shows relatively good performances. In the case of function f_4, SGHS, NGHS, GSHS, and MHS converge to the global optimum. However, in case of function f_3, the gaps in performance are obvious. The best mean value obtained by MHS is e-02, while the worst is e+04, which is obtained by i-HS. Based on these observations, it may be inferred that MHS provides significantly better solutions for unimodal functions than other HS variants. For multimodal functions $f_6 - f_{13}$, MHS also provides competitive results. In the cases of function f_9, f_{10}, f_{12}, and f_{13}, MHS provides the best results for the mean, best, and worst values. In the cases of function f_6 and f_{11}, MHS provides the best mean values. In the cases of functions f_7 and f_8, NGHS finds the best results. For the shifted and rotated functions $f_{14} - f_{19}$, MHS provides satisfactory results again. In the cases of function f_{14}, f_{15}, and f_{18}, MHS provides the best results on both mean, best and worst values. In case of function f_{16}, MHS provides the best mean values. In the cases of function f_{17} and f_{19}, NGHS shows the best performance, followed by MHS.

For most of the meta-heuristic optimization algorithms, their performances sharply decrease when the search space is enlarged and dimensionality increases [42]. As shown in Table 7, the solution accuracy of all of the HS variants are reduced when the dimension of the functions increases from 30 to 50. However, from Table 7, it can be seen that the MHS algorithm still performs better than the other HS variants. As a whole, out of the 19 functions studied, MHS has outperformed all other variants of HS in 11 benchmark functions for the “best” fitness values, 13 benchmark functions for the “worst” fitness values, and 12 benchmark functions for the 50-dimensional “mean” fitness values. Furthermore, for the cases of functions f_3, f_6, f_{17} and f_{19}, the performance of MHS ranked in 2nd place among the compared HS variants. The worst performance of MHS occurs in solving function f_8, for which it ranked in 4th place among all of the compared algorithms.

4.2.3 Rank based analysis

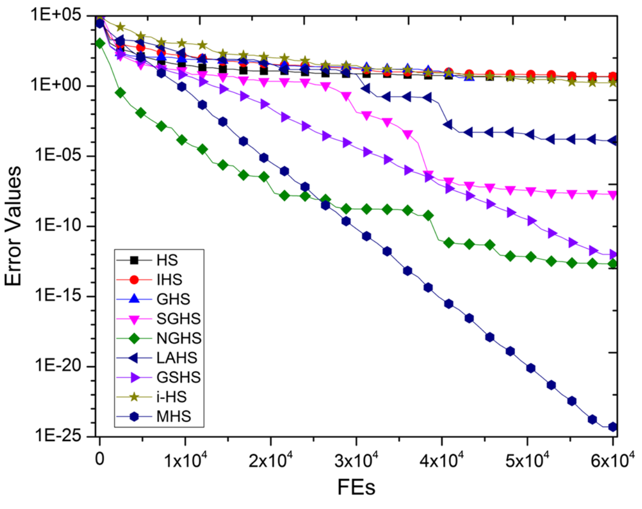
In this section, a rank based analysis method is utilized to evaluate the performance of the compared HS variants. The rank rules are as follows: all of the algorithms obtain a rank value based on their performance on “best”, “worst” and “mean” values, and algorithms receive the same rank if they show the same performance. The next algorithms will be assigned with a gap determined by the number of algorithms that show the same performance. The ranks of the HS variants and the average rank for all of the functions based on both the cases of the 30 and 50-dimensional best performances are shown in Table 8. Based on the average ranking, the order of performance obtained is MHS, followed in order by NGHS, SGHS, GHS, GSHS, LAHS, IHS, i-HS and HS in the 30-dimensional case, and it is MHS followed in order by NGHS, SGHS, GSHS, LAHS, i-HS, HS, GHS and IHS in the 50-dimensional case.

Table 11 P-values calculated for Wilcoxon's Rank Sum Test for all of the benchmarks problems(30D)

P-values	HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS
f_1	3.01986E-11	3.01986E-11	5.57265E-10	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_2	3.00287E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	8.48075E-9	3.01986E-11
f_3	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	4.50432E-11	3.01986E-11
f_4	1.00127E-12	1.198E-12 4.	56206E-8	NA	NA	1.07324E-12	NA	2.14200E-2
f_5	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_6	3.01986E-11	3.01986E-11	3.01986E-11	1.440E-3	3.18208E-4	3.68973E-11	3.4742E-10	3.01986E-11
f_7	9.73349E-8	1.86492E-7	8.26678E-7	7.94582E-6	8.16498E-5	1.86492E-7	8.92148E-7	1.51193E-6
f_8	2.68759E-11	2.84489E-10	7.33159E-6	8.2935E-1	1.66700E-02	3.63407E-11	5.21600E-2	5.78320E-1
f_9	2.96169E-11	2.96169E-11	2.96169E-11	2.96169E-11	2.96169E-11	2.96169E-11	2.96169E-11	2.96169E-11
f_{10}	7.84553E-12	7.85105E-12	7.85105E-12	1.82371E-11	1.90715E-11	7.85105E-12	1.33763E-7	7.85105E-12
f_{11}	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01797E-11	3.01986E-11	3.38200E-2	3.01986E-11
f_{12}	1.31881E-12	1.45662E-12	1.29757E-12	7.51866E-13	1.55451E-12	1.22171E-12	1.40133E-12	1.0759E-12
f_{13}	3.01986E-11	3.01797E-11	5.57265E-10	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01797E-11
f_{14}	1.21178E-12	1.21178E-12	4.57359E-12	NA	NA	7.71192E-13	1.21178E-12	1.21178E-12
f_{15}	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_{16}	3.01986E-11	3.01986E-11	3.01986E-11	1.04066E-4	1.40669E-4	3.01986E-11	3.01986E-11	3.01986E-11
f_{17}	2.12926E-11	2.12926E-11	1.08838E-7	2.09782E-11	1.55969E-9	2.12926E-11	2.12926E-11	2.12926E-11
f_{18}	6.38241E-12	6.38697E-12	6.38697E-12	7.98427E-9	3.98299E-8	1.43452E-10	5.7634E-11	4.69439E-11
f_{19}	2.47287E-11	2.47287E-11	9.92095E-10	2.44618E-11	1.04821E-4	2.47287E-11	2.47287E-11	2.47287E-11

Table 12 P-values calculated for Wilcoxon's Rank Sum Test for all of the benchmarks problems(50D)

P-values	HS	IHS	GHS	SGHS	NGHS	LAHS	GSHS	i-HS
f_1	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_2	3.0123E-11	3.01986E-11	3.01986E-11	3.01418E-11	3.01986E-11	3.01986E-11	1.42984E-5	3.01986E-11
f_3	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	5.09117E-6	3.01986E-11
f_4	1.17088E-12	1.21079E-12	1.20586E-12	NA	NA	1.168E-12	NA	9.70201E-13
f_5	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_6	3.01986E-11	3.01986E-11	3.01986E-11	2.81300E-2	5.6922E-1	8.89099E-10	2.59736E-5	3.01986E-11
f_7	1.44142E-6	1.60722E-11	4.28674E-10	3.30634E-4	1.690E-3	1.06544E-6	5.35137E-5	5.35137E-5
f_8	2.98596E-11	2.98596E-11	2.98596E-11	2.98596E-11	2.98596E-11	2.98596E-11	2.98596E-11	7.006E-2
f_9	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_{10}	1.09147E-11	1.09147E-11	1.09147E-11	3.23867E-11	5.4051E-10	1.09072E-11	3.612E-6	1.09147E-11
f_{11}	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	1.61323E-10	3.01986E-11	1.01848E-5	3.01986E-11
f_{12}	1.75689E-11	1.7203E-11	1.47618E-11	1.38718E-11	1.60085E-11	1.24126E-11	1.52008E-11	1.75113E-11
f_{13}	3.01986E-11	3.01797E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_{14}	1.21178E-12	1.21178E-12	1.21178E-12	NA	NA	1.2098E-12	1.21178E-12	1.21178E-12
f_{15}	3.01986E-11	3.01986E-11	3.01986E-11	2.12646E-4	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_{16}	3.01986E-11	3.01986E-11	3.01986E-11	5.46203E-6	8.4180E-1	3.01986E-11	1.09367E-10	3.01986E-11
f_{17}	3.01986E-11	3.01986E-11	3.01986E-11	3.01797E-11	2.8936E-11	3.01986E-11	3.01986E-11	3.01986E-11
f_{18}	1.22042E-11	1.22042E-11	1.22042E-11	3.07201E-7	1.87334E-4	1.21794E-11	3.42383E-11	1.21960E-11
f_{19}	3.01986E-11	3.01986E-11	3.01986E-11	3.01986E-11	3.01608E-11	3.01986E-11	3.01986E-11	3.01986E-11



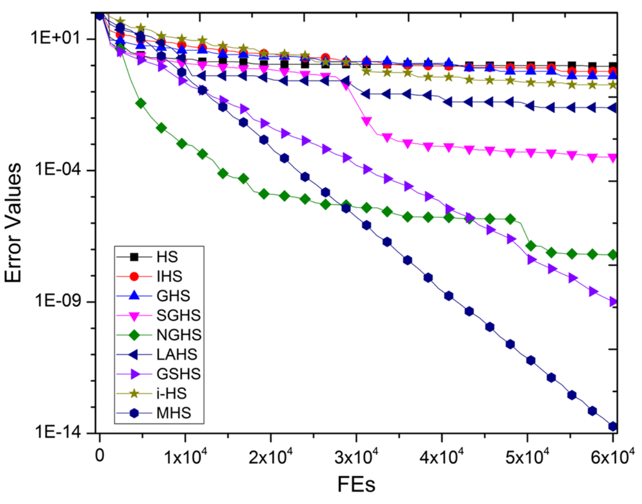
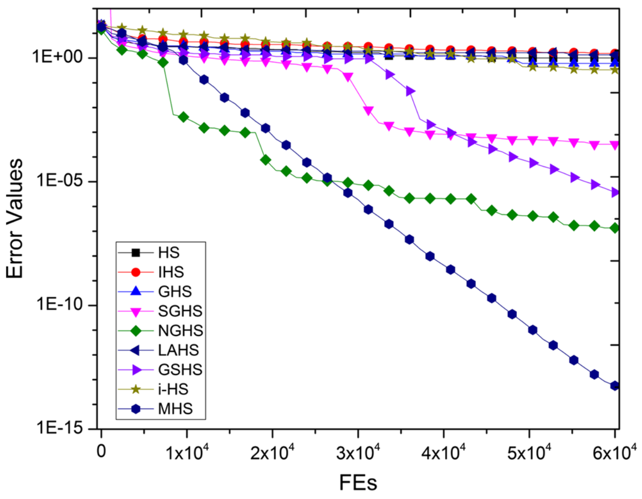


Fig. 10 Convergence graphs of the nine compared algorithms on unimodal functions (in 30D). **a** Sphere function (f_1). **b** Schewefel's function 2.22. (f_2)

Figure 7 shows the histograms that indicate the number of times each HS variant achieved ranks in the range of 1 to 9. In detail, Fig. 7a shows the results for the 30-dimensional case, and Fig. 7b shows the same for the 50-dimensional case. The performance of an algorithm can be judged by the height of the histogram: a higher rectangle with cool colors indicates better performance, while a higher rectangle with warm colors indicates worse performance. It can be seen that MHS achieves the top rank five times more than NGHS, which is the second best algorithm in the 30-dimensional case. For the 50-dimensional case, MHS is the top ranked algorithm four times more than the second best algorithm, which is NGHS again.

Next, the compared algorithms are ranked based on their performances in the competition of "worst" and "mean" value. The rankings of the compared algorithms and the average rank based on all of the functions for both the 30-dimensional and 50-dimensional mean performances are presented in Tables 9 and 10. As can be seen from Table 9, MHS is ranked first based on the average rank in the 30 dimensional case, followed by NGHS, SGHS, GSHS, i-HS, HS, LAHS, GHS and IHS. The rank order in the 50 dimensional case is MHS, NGHS, SGHS, GSHS, i-HS, GHS, LAHS, HS and IHS.

As with the "best" value rank analysis, the histograms of the "worst" and "mean" values ranks are depicted in Figs. 8



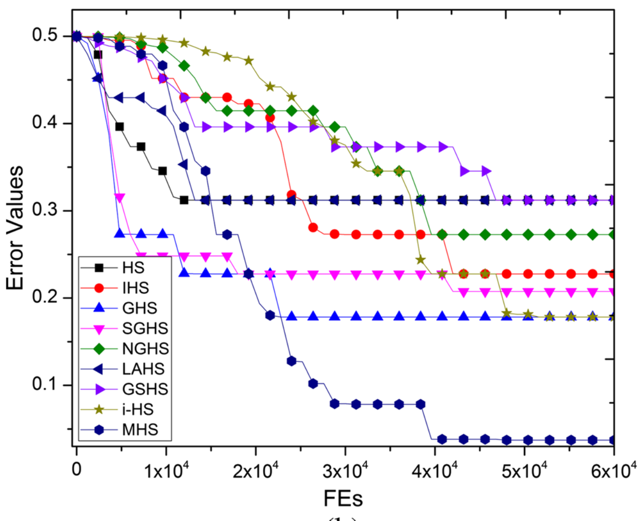
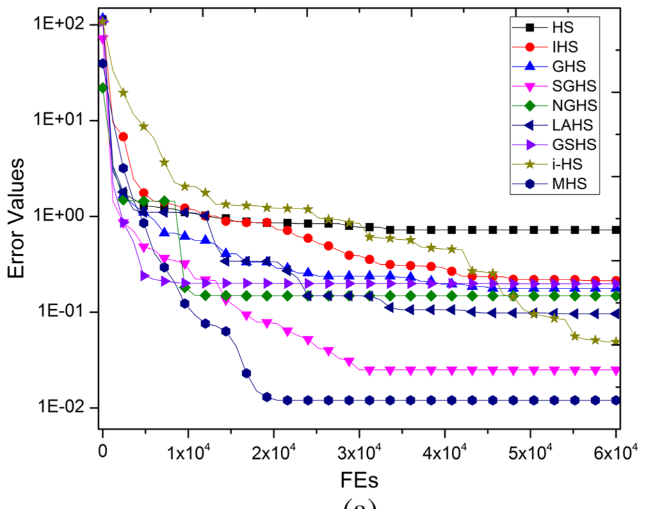
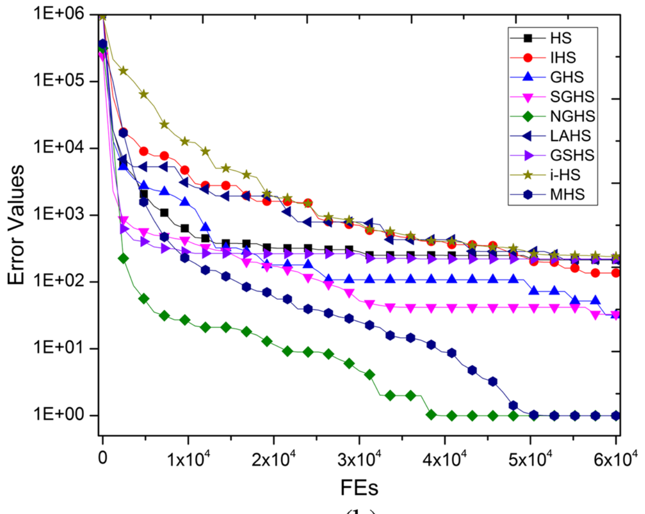


Fig. 11 Convergence graphs of the nine compared algorithms on multimodal functions (in 30D). **a** Ackley function (f_9). **b** Shaffer function (f_{12})



(a)



(b)

Fig. 12 Convergence graphs of the nine compared algorithms on shifted and rotated functions (in 30D). **a** Shifted rotated Griewank function (f_{18}). **b** Shifted rotated Rastrigin function (f_{19})

and 9, respectively. It is again clear that MHS outperforms the other HS variants.

4.2.4 Statistical comparison by Wilcoxon's rank sum test

A rank based analysis method has been applied to validate the superiority of MHS, but it is not a conclusive measure. Hence, in this section, a nonparametric statistical test called Wilcoxon's rank sum test for independent samples is conducted at the significance level of 5 % to judge whether the results obtained using the MHS algorithm differ from the final results of the other competitors in a statistically significant way.

The P-values calculated through the rank sum between MHS and other HS variants over all 19 benchmark functions are provided in Tables 11 and 12, for the 30 dimensional case and the 50 dimensional case, respectively. In these tables, NA appears when the related algorithms obtained the same solution with MHS. It can be observed from Table 11 that MHS achieved statistically superior performance compared to all of the other HS variants for 16 functions out of 19 in the 30 dimensional case. The differences among MHS, SGHS, NGHS, and GSHS are not statistically significant for test function f_4; the differences among MHS, SGHS, and GSHS are not statistically significant for test function f_8; the differences among MHS, SGHS and NGHS are not statistically significant for test function f_{14}. Similar observations can be found in Table 12 for the 50 dimensional case, where MHS also achieved statistically superior performance compared to all of the other HS variants for 16 functions out of 19. The differences among MHS, SGHS, NGHS, and GSHS are not statistically significant for test function f_4; the differences between MHS and NGHS are not statistically significant for test function f_6; the differences among MHS, SGHS, and NGHS are not statistically significant for test function f_{14}.

4.2.5 Comparison of the convergence properties

In this section, the convergence properties of all of the compared HS variants in 30 dimensional case are investigated. To save space, only some of the benchmark functions with different characteristics are performed. Figures 10, 11 and 12 give the convergence graphs of unimodal functions, multimodal functions and shifted and rotated functions, respectively. From Fig. 10, it can be seen that NGHS converges faster than other other algorithms in the first half of *FEs*, while MHS catches up during the second half. The same situation happens for function f_9, which is depicted in Fig. 11a. For function f_{12}, GHS and SGHS converge faster than other algorithms during the initial stage of iteration, but they are then trapped into the local optima, while MHS can jump out of the local optima and continue converging. Finally, for the shifted and rotated functions, MHS converges faster than the other algorithms for function f_{18} (Fig. 12a) while converging slower than NGHS for function f_{19} (Fig. 12b).

4.3 Comparison of MHS and other meta-heuristic algorithms

To further evaluate the performance of MHS, 28 benchmarks of CEC2013 are tested, and the results are compared

Table 13 Mean and standard deviation ($\pm SD$) of CEC2013 optimization results (D=10)

Functions		DE/rand1/bin	SHADE	JADE	SPSO2011	HPSO	ABC	MHS
F_1	AE	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	SD	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F_2	AE	2.4200E+03	0.0000E+00	0.0000E+00	3.6340E+04	1.4400E+05	1.7917E+05	0.0000E+00
	SD	5.2700E+03	0.0000E+00	0.0000E+00	7.356E+04	1.0400E+05	7.5278E+04	0.0000E+00
F_3	AE	1.4100E+00	1.2663E-01	4.1000E+01	2.682E+05	6.7500E+05	4.1799E+06	6.6834E-01
	SD	2.5100E+00	8.8399E-01	8.4100E+01	1.6560E+07	1.9600E+06	4.0847E+06	1.5193E+00
F_4	AE	2.7100E+01	0.0000E+00	0.0000E+00	8.8690E+03	4.1600E+02	8.1650E+03	0.0000E+00
	SD	8.3400E+01	0.0000E+00	0.0000E+00	4.556E+03	3.3700E+02	2.2174E+03	0.0000E+00
F_5	AE	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	SD	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F_6	AE	3.2900E+00	7.8884E+00	8.0800E+00	9.8000E+00	2.6400E+00	5.6284E-01	4.3722E-02
	SD	4.3000E+00	3.9346E+00	3.7400E+00	4.9740E+00	2.0500E+00	1.3639E+00	1.4009E-01
F_7	AE	1.4400E-03	3.2593E-03	9.3900E-02	2.1100E+01	1.9200E+00	2.8871E+01	4.2752E-06
	SD	9.5000E-03	4.5447E-03	1.4000E-01	1.3270E+01	2.7000E+00	7.2569E+00	9.7290E-06
F_8	AE	2.0400E+01	2.0353E+01	2.0400E+01	2.0300E+01	2.0300E+01	2.0403E+01	2.0201E+01
	SD	6.6400E-02	8.9488E-02	8.7700E-02	6.7220E-02	8.7100E-02	6.7017E-01	4.5956E-02
F_9	AE	1.1400E+00	3.3895E+00	3.8200E+00	4.8000E+00	2.7500E+00	4.7328E+00	5.0622E-01
	SD	9.9000E-01	7.3507E-01	5.8500E-01	1.4990E+00	1.0900E+00	6.7482E-01	5.4319E-01
F_{10}	AE	4.9200E-02	1.1987E-02	1.7200E-02	3.0000E-01	5.1300E-01	1.1049E+00	1.0055E-03
	SD	2.5800E-02	8.9885E-03	8.7600E-03	2.7130E-01	3.1500E-01	2.5239E-01	2.5267E-03
F_{11}	AE	1.1400E+00	0.0000E+00	0.0000E+00	1.0900E+01	1.7600E-01	0.0000E+00	0.0000E+00
	SD	2.1300E+00	0.0000E+00	0.0000E+00	5.6580E+00	3.7900E-01	0.0000E+00	0.0000E+00
F_{12}	AE	8.2400E+00	3.1413E+00	4.5900E+00	1.3900E+01	7.0400E+00	2.1054E+01	2.8675E+00
	SD	5.6200E+00	9.7343E-01	1.3900E+00	6.5600E+00	2.9500E+00	4.2957E+00	7.3821E-01
F_{13}	AE	1.2100E+01	3.7708E+00	4.8000E+00	2.0800E+01	1.1500E+01	2.7017E+01	1.9102E+00
	SD	6.3100E+00	1.8530E+00	2.1800E+00	9.8220E+00	4.7700E+00	6.3217E+00	7.7834E-01
F_{14}	AE	2.2000E+02	4.8984E-03	1.2200E-02	8.3380E+02	3.7800E+01	6.9677E-02	2.4781E-01
	SD	2.8000E+02	1.6958E-02	2.7700E-02	2.3350E+02	4.2100E+01	4.3649E-02	6.5820E-01
F_{15}	AE	1.1300E+03	4.2075E+02	4.7400E+02	7.7430E+02	4.5400E+02	5.9421E+02	2.5181E+02
	SD	2.6900E+02	1.1444E+02	1.1900E+02	2.5070E+02	1.6700E+02	1.4225E+02	1.1021E+02
F_{16}	AE	1.0100E+00	7.0799E-01	1.0800E+00	5.0000E-01	4.0700E-01	7.2100E-01	2.2466E-01
	SD	2.4200E-01	2.1188E-01	1.8200E-01	2.4570E-01	1.3600E-01	1.1243E-01	4.1409E-02
F_{17}	AE	1.7800E+01	1.0122E+01	1.0100E+01	1.8900E+01	1.1000E+01	8.3298E+00	1.0151E+01
	SD	5.3600E+00	0.0000E+00	7.9100E-15	5.8730E+00	3.2800E+00	2.7494E+00	6.9414E-02
F_{18}	AE	3.1600E+01	1.6878E+01	1.8600E+01	1.7800E+01	1.5600E+01	3.7214E+01	1.3489E+01
	SD	5.5800E+00	1.5363E+00	2.2100E+00	4.5340E+00	2.3300E+01	5.6907E+00	1.0213E+00
F_{19}	AE	1.0700E+00	3.4352E-01	3.5000E-01	9.0000E-01	5.0100E-01	2.1969E-02	2.9686E-01
	SD	4.9800E-01	4.8987E-02	5.1500E-02	3.8860E-01	1.3800E-01	1.3272E-02	8.0853E-02
F_{20}	AE	2.3600E+00	2.1572E+00	2.2900E+00	3.4000E+00	2.5200E+00	3.1228E+00	8.2376E-01
	SD	3.5800E-01	3.5157E-02	3.8000E-01	4.1940E-01	4.7700E-01	2.5798E-01	2.5365E-01
F_{21}	AE	3.7300E+02	4.0019E+02	4.0000E+02	4.0000E+02	3.7500E+02	1.3153E+02	4.0019E+02
	SD	6.9600E+01	0.0000E+00	0.0000E+00	0.0000E+00	7.1000E+01	3.2981E+01	0.0000E+00
F_{22}	AE	2.2300E+02	4.8396E+00	3.8300E+00	9.0600E+02	1.2200E+02	1.2146E+01	1.9664E+01
	SD	2.3600E+02	6.1972E+00	3.8000E+00	3.4310E+02	7.5700E+01	1.1862E+01	2.4803E+01
F_{23}	AE	9.7700E+02	4.6061E+02	5.0400E+02	9.1000E+02	5.1500E+02	9.2780E+02	4.2203E+02
	SD	3.3400E+02	1.7839E+02	1.4700E+02	3.5960E+02	1.9700E+02	1.9761E+02	2.6741E+02
F_{24}	AE	2.0200E+02	1.9344E+02	2.0100E+02	2.1400E+02	2.0300E+02	1.2699E+02	1.9876E+02
	SD	1.4600E+01	2.4638E+01	1.1900E+01	9.1660E+00	1.9100E+01	5.4003E+00	1.3694E+01

Table 13 (continued)

Functions		DE/rand1/bin	SHADE	JADE	SPSO2011	HPSO	ABC	MHS
F_{25}	AE	2.0200E+02	2.0015E+02	2.0000E+02	2.0900E+02	2.0500E+02	1.5008E+02	1.9876E+02
	SD	1.4600E+01	7.0243E-01	5.3900E+00	5.9430E+00	1.4400E+01	1.1909E+01	1.3761E+01
F_{26}	AE	1.6700E+02	1.3333E+02	1.2707E+02	2.0000E+02	1.8900E+02	1.2713E+02	1.7151E+02
	SD	4.8600E+01	4.3581E+01	3.8500E+01	5.5130E+01	5.1300E+01	6.9580E+00	4.3013E+01
F_{27}	AE	3.3700E+02	3.0000E+02	3.0000E+02	3.3600E+02	3.7000E+02	3.8873E+02	3.9019E+02
	SD	7.5800E+01	1.4604E-08	1.8500E+00	7.3590E+01	3.2200E+01	1.5714E+01	2.9737E+01
F_{28}	AE	2.9200E+02	3.0000E+02	3.0000E+02	3.0000E+02	3.2600E+02	1.1522E+02	3.7448E+02
	SD	3.9200E+01	0.0000E+00	0.0000E+00	8.3620E+01	1.2500E+02	3.3762E+01	9.3356E+01
Wins		2	7	6	2	4	9	17

with the following algorithms which include ABC, two PSO variants, and three DE variants:

1. The latest Standard PSO 2011 [52] with *Np*=40, acceleration coefficients $c_1, c_2 = 0.5+ln(2)$ and constant inertia weight $\omega = 1/(2*ln(2))$;
2. Self-adaptive Heterogeneous PSO (SaHPSO) [53] with *Np*=50, other parameters such as the stagnation threshold, k and the tournament size are self-adapted during the search process;
3. Artificial bee colony algorithm (ABC) [6] with *Np*=40, percentage of onlooker bees $p = 50\ \%$;
4. DE/rand1/bin [54]with *Np*=50, F =0.5 and *Cr*=0.90;
5. JADE [55] with *Np*=100, c =0.1, p =0.05, and optional external archive;
6. SHADE [56] with *Np*=100, memory size H =*Np*=100.

All of the results are taken from the referenced literature except ABC. The ABC code was download from the website (http://sci2s.ugr.es/eamhco/src/ABC.C). All of the benchmarks are tested in 10 and 30 dimensions. The maximum number of *FEs* was set to 1×10^5 for 10-D and to 3×10^5 for 30-D, according to the guidelines provided in CEC2013 special session.

Tables 13 and 14 show the mean and standard deviation of the best-of-run errors for 51 independent runs of each of the seven algorithms for D=10 and D=30, respectively. Note that the best-of-run error corresponds to the absolute difference between the best-of-run value $f(\vec{X}_{\text{best}})$ and the actual global optimum f^* of a particular objective function $|f(\boldsymbol{X}_{\text{best}}) - f^*|$. When the error value is smaller than 10^{-8}, it is taken as 0.

From Table 13 we can see that, considering the mean of the error values for 10D problems, MHS outperformed all of the other contestant algorithms, obtaining 17 best solutions out of 28 test functions, followed by ABC, obtaining nine best solutions. Moreover, for f_1 and f_5, all the contestant algorithms obtain the global optimum for every run. For function f_{11}, four algorithms obtain the global optimum, including SHADE, JADE, ABC, and MHS. For f_2 and f_4, three algorithms obtain the global optimum, including SHADE, JADE and MHS.

Table 14 shows that the superiority of MHS is more evident when the dimensionality increases. MHS obtains the best solutions for 18 test functions out of 28 in this case, followed by JADE, which obtained seven of the best solutions; SHADE, which obtained six, and ABC obtained five. The performance of DE/rand/1 is not good when dealing with 30-D problems, obtaining only one best solutions.

From the experiment showed above, we can see that the MHS algorithm, which maintains a good balance between exploration and exploitation, outperforms the state-of-the-art HS variants and several other meta-heuristic algorithms. The success of MHS could come from two aspects: first, the intersect mutation operation in MHS maintains the diversity of harmonies well; second, the local search based on the CA mechanism further enhances the local exploitation ability of MHS.

Table 14 Mean and standard deviation ($\pm SD$) of CEC2013 optimization results (D=30)

Functions		DE/rand1/bin	SHADE	JADE	SPSO2011	SaHPSO	ABC	MHS
F_1	AE	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	SD	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F_2	AE	1.5400E+05	9.0022E+03	9.6100E+03	3.0880E+05	1.5900E+06	1.2471E+07	8.5494E+03
	SD	6.5800E+04	7.4746E+03	5.8900E+03	1.8750E-13	8.0300E+05	2.5545E+06	5.3429E+03
F_3	AE	3.6500E+06	4.0206E+01	4.1200E+05	1.1880E+08	2.4000E+08	6.5819E+08	1.8287E+00
	SD	5.8400E+06	2.1298E+02	1.6000E+06	5.2430E+08	3.7100E+08	3.6187E+08	6.8254E+00
F_4	AE	4.6200E+02	1.9216E-04	3.2700E+03	3.1940E+04	4.7800E+02	6.7303E+03	4.7457E-07
	SD	4.1500E+02	3.0100E-04	1.0200E+04	6.7020E+03	1.9600E+02	7.2158E+03	1.7442E-06
F_5	AE	3.0900E-05	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	SD	1.1700E-04	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
F_6	AE	1.9800E+01	5.9596E-01	1.1100E+00	2.8300E+01	2.9900E+01	1.3018E+01	1.9023E+00
	SD	1.3700E+01	3.7286E+00	5.1400E+00	2.8250E+01	1.7600E+01	3.1147E+00	5.4532E+00
F_7	AE	1.5800E+00	4.6016E+00	3.0100E+00	8.6900E+01	6.3900E+01	1.1345E+02	1.9203E-01
	SD	1.6000E+00	5.3852E+00	4.3400E+00	2.1070E+01	3.0900E+01	1.0746E+01	2.6793E-01
F_8	AE	2.0900E+01	2.0723E+01	2.0900E+01	2.0900E+01	2.0900E+01	2.0918E+01	2.0817E+01
	SD	4.9900E-02	1.7608E-01	9.1200E-02	5.8930E-02	6.2800E-02	4.2949E-02	7.2505E-02
F_9	AE	9.1700E+00	2.7466E+01	2.6200E+01	2.8400E+01	1.8500E+01	2.9243E+01	3.7421E+00
	SD	2.2400E+00	1.7694E+00	1.6200E+00	4.4260E+00	2.6900E+00	1.3807E+00	1.7233E-01
F_{10}	AE	7.6200E-02	7.6855E-02	5.3600E-02	3.0000E-01	2.2900E-01	3.4365E-01	1.1122E-03
	SD	4.9200E-02	3.5751E-02	3.9600E-02	1.4780E-01	1.3200E-01	9.4863E-02	3.4324E-03
F_{11}	AE	1.4200E+01	0.0000E+00	0.0000E+00	1.0840E+02	2.3600E+01	0.0000E+00	9.9496E-02
	SD	4.5600E+00	0.0000E+00	0.0000E+00	2.7400E+01	8.7600E+00	0.0000E+00	2.9848E-01
F_{12}	AE	1.1400E+02	2.3039E+01	2.4700E+01	9.4500E+01	5.6400E+01	2.4485E+02	2.1673E+01
	SD	6.6900E+01	3.7320E+00	4.5700E+00	3.5390E+01	1.5100E+01	2.6112E+01	5.7683E+00
F_{13}	AE	1.5300E+02	5.0337E+01	4.8800E+01	1.9768E+02	1.2300E+02	2.9120E+02	7.9139E+01
	SD	3.9000E+01	1.3371E+01	1.0500E+01	3.8620E+01	2.1900E+01	2.5383E+01	9.6661E+00
F_{14}	AE	5.7200E+02	3.1841E-02	3.6300E-02	4.0230E+03	7.0400E+02	2.5457E+00	3.0792E+00
	SD	5.4600E+02	2.3315E-02	2.6300E-02	6.1940E+02	2.3800E+02	1.4114E+00	1.8908E+00
F_{15}	AE	7.0100E+03	3.2179E+03	3.3300E+01	3.8040E+03	3.4200E+03	3.9241E+03	4.8971E+03
	SD	5.4200E+02	2.6367E+02	7.9600E-15	6.9380E+02	5.1600E+02	2.8754E+02	3.8821E+02
F_{16}	AE	2.4500E+00	9.1315E-01	2.1500E+00	1.4000E+00	8.4800E-01	1.5278E+00	7.7581E-01
	SD	3.0200E-01	1.8549E-01	4.1000E-01	3.5880E-01	2.2000E-01	2.0277E-01	4.3645E-01
F_{17}	AE	5.6200E+01	3.0434E+01	3.0400E+01	1.1520E+02	5.2600E+01	3.0497E+01	3.0866E+01
	SD	1.9600E+01	3.8300E-14	0.0000E+00	2.0180E+01	7.1100E+00	2.7619E-02	2.8468E-01
F_{18}	AE	1.9900E+02	7.2457E+01	7.7500E+01	1.1680E+02	6.8100E+01	3.1254E+02	1.9126E+01
	SD	9.8700E+00	5.5837E+00	6.5200E+00	2.4600E+01	9.6800E+00	3.2919E+01	1.0802E+01
F_{19}	AE	3.9300E+00	1.3557E+00	1.9100E+00	9.0000E+00	3.1200E+00	6.2702E-01	4.5265E-01
	SD	2.8700E+00	1.2013E-01	1.2400E-01	4.4180E+00	9.8300E-01	1.6141E-01	6.4395E-02
F_{20}	AE	1.1900E+01	1.0473E+01	1.0500E+02	1.4000E+01	1.2000E+01	1.4484E+01	1.9545E+00
	SD	3.4100E-01	6.0435E-01	1.8100E+01	1.1090E+00	9.2600E-01	2.0247E-01	4.8024E-01
F_{21}	AE	3.0700E+02	3.0904E+02	3.0000E+02	3.0000E+02	3.1100E+02	2.0926E+02	3.5835E+02
	SD	7.9200E+01	5.6473E+01	6.9200E+01	6.7960E+01	7.9200E+01	2.6236E+01	9.6275E+01
F_{22}	AE	4.4400E+02	9.8095E+01	1.1500E+02	4.3510E+03	8.5900E+02	9.8947E+01	6.9012E+01
	SD	2.0100E+02	2.5213E+01	1.9100E+00	7.6700E+02	3.1000E+02	3.0061E+01	1.6137E+01
F_{23}	AE	7.1100E+03	3.5078E+03	3.4200E+03	4.7630E+03	3.5700E+03	4.7965E+03	5.0464E+03
	SD	2.6400E+02	4.1093E+02	3.8000E+02	8.2270E+02	5.9000E+02	3.7277E+02	7.0221E+02
F_{24}	AE	2.1700E+02	2.0525E+02	2.0800E+01	2.6400E+02	2.4800E+02	2.8485E+02	2.0004E+02
	SD	1.0900E+01	5.2904E+00	8.3900E+00	1.2460E+01	8.1100E+00	5.3224E+00	5.5927E-02

Table 14 (continued)

Functions		DE/rand1/bin	SHADE	JADE	SPSO2011	SaHPSO	ABC	MHS
F_{25}	AE	2.4800E+02	2.5944E+02	2.7600E+02	3.0000E+02	2.4900E+02	3.0322E+02	2.3860E+02
	SD	4.3800E+00	1.9645E+01	1.5100E+01	1.0450E+01	7.8200E+00	4.4790E+00	3.6734E+00
F_{26}	AE	2.3700E+02	2.0208E+02	2.1500E+02	3.4000E+02	2.9500E+02	2.0074E+02	2.0000E+02
	SD	5.5200E+01	1.4844E+01	4.1200E+01	8.2400E+01	7.0600E+01	1.3501E-01	2.7815E-03
F_{27}	AE	4.9500E+02	3.8763E+02	6.5700E+02	1.0260E+03	7.7600E+02	4.0000E+02	3.4485E+02
	SD	9.3000E+01	1.0897E+02	2.1500E+02	1.1190E+02	7.1100E+01	3.4142E-02	8.1123E+01
F_{29}	AE	3.0000E+02	3.0000E+02	3.0000E+02	3.0000E+02	4.0100E+02	2.2060E+02	3.0000E+02
	SD	0.0000E+00	0.0000E+00	0.0000E+00	4.7610E+02	3.4800E+02	5.6722E+01	0.0000E+00
Wins		1	6	7	2	2	5	18

5 Conclusions

A modified HS algorithm with intersect mutation operator and cellular local search (MHS) is presented in this paper. First, the MHS algorithm utilizes an intersect mutation operation. In the MHS algorithm, all harmonies in the harmony memory are divided into a better part and a worse part according to their fitness, and then novel mutation and crossover operations are developed to generate new-harmony vectors. Second, a cellular local search is adopted to further enhance the local exploitation ability of MHS. From the experimental results listed in Section 4, we can see that the proposed MHS algorithm performs better than the state-of-the-art HS variants and is competitive with other meta-heuristic algorithms in terms of solution accuracy and efficiency. The superiority of MHS in both efficiency and accuracy could come from the intersect mutation strategy, cellular local search, and a better balance of global exploration and local exploitation.

The following are the main contributions of this study:

1. It proposes a new variant of the HS algorithm in which intensification and diversification are well balanced.
2. The proposed MHS algorithm is successfully applied to function optimization in this paper. It could be applied to other problems.
3. MHS may also provide a new improvement strategy for other algorithms.

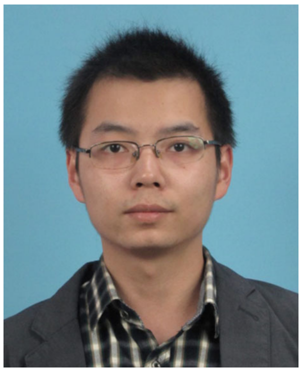
In the future, intelligent tuning strategies for MHS algorithm parameters can be studied for seeking further improvements. Furthermore, the proposed MHS algorithm will be expanded to solve certain constrained optimization problems, multi-objective problems, and optimization problems with high computational cost. Some engineering applications of MHS will also be investigated in the near future.

Acknowledgments The authors would like to thank anonymous reviewers for their helpful comments. This research work is supported by the National Basic Research Program of China (973 Program) under grant no. 2014CB046705 and the National Natural Science Foundation of China (NSFC) under Grant Nos. 51375004, 51435009 and 51421062.

References

1. Boussaid I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117
2. Holland JH (1975) Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence
3. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
4. Zou DX, Wu JH, Gao LQ, Li S (2013) A modified differential evolution algorithm for unconstrained optimization problems. Neurocomputing 120:469–481
5. Kennedy J, Eberhart R (1995) particle swarm optimization, proceedings of IEEE International Conference on neural networks (ICNN'95) in
6. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471
7. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents, Systems, Man, and Cybernetics. IEEE Transactions on Part B: Cybernetics 26:29–41
8. Yang X-S, Deb S (2009) Cuckoo search via Lévy flights nature & biologically inspired computing, 2009. NaBIC World Congress on IEEE, pp 210–214
9. Kirkpatrick S (1984) annealing, Optimization by simulated Quantitative studies. J Stat Phys 34:975–986
10. Lam AY, Li VO (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14:381–399
11. Geem ZW, Kim JH, Loganathan G (2001) A new heuristic optimization algorithm: harmony search. Simulation 76:60–68
12. Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188:1567–1579
13. Geem ZW (2009) Particle-swarm harmony search for water network design. Eng Optim 41:297–311
14. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798

15. Panchal A (2009) Harmony search in therapeutic medical physics, in: Music-inspired Harmony search algorithm. Springer, pp 189–203
16. Alia O, Mandava R, Ramachandram D, Aziz ME (2009) Dynamic fuzzy clustering using harmony search with application to image segmentation. In: Signal Processing and Information Technology (ISSPIT) IEEE International Symposium on IEEE, pp 538–543
17. Zarei O, Fesanghary M, Farshi B, Saffar RJ, Razfar MR (2009) Optimization of multi-pass face-milling via harmony search algorithm. J Mater Process Tech 209:2386–2392
18. Liu L, Zhou H (2013) Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem. Inform Sci 226:68–92
19. Cuevas E (2013) Block-matching algorithm based on harmony search optimization for motion estimation. Appl Intell 39:165–183
20. Arul R, Ravi G, Velusami S (2013) Chaotic self-adaptive differential harmony search algorithm based dynamic economic dispatch. Int J Electr Power Energy Syst 50:85–96
21. Kulluk S, Ozbakir L, Baykasoglu A (2012) Training neural networks with harmony search algorithms for classification problems. Eng Appl Artif Intell 25:11–19
22. Mun S, Cho YH (2012) Modified harmony search optimization for constrained design problems. Expert Systems with Applications 39:419–423
23. Chen J, Pan QK, Li JQ (2012) Harmony search algorithm with dynamic control parameters. Applied Mathematics and Computation 219:592–604
24. Omran MGH, Mahdavi M (2008) Global-best harmony search. Appl Math Comput 198:643–656
25. Wang L, Li LP (2013) An effective differential harmony search algorithm for the solving non-convex economic load dispatch problems. Int J Electr Power Energy Syst 44:832–843
26. Banerjee A, Mukherjee V, Ghoshal S (2013) An opposition-based harmony search algorithm for engineering optimization problems. Ain Shams Eng J
27. Wang H, Ouyang H, Gao L, Qin W (2014) Opposition-based learning harmony search algorithm with mutation for solving global optimization problems, In: Control and Decision Conference (2014 CCDC), The 26th Chinese, IEEE, pp 1090–1094
28. Zou D, Gao L, Wu J, Li S (2010) Novel global harmony search algorithm for unconstrained problems. Neurocomputing 73:3308–3318
29. Im SS, Yoo DG, Kim JH (2013) Smallest-small-world cellular harmony search for optimization of unconstrained benchmark problems. J Appl Math:2013
30. Al-Betar MA, Khader AT, Awadallah MA, Alawan MH, Zaqaibeh B (2013) Cellular harmony search for optimization problems. J Appl Math:2013
31. Ashrafi SM, Dariane AB (2013) Performance evaluation of an improved harmony search algorithm for numerical optimization: Melody Search (MS). Eng Appl Artif Intell 26:1301–1321
32. Al-Betar MA, Awadallah MA, Khader AT, Abdalkareem ZA (2015) Island-based harmony search for optimization problems. Expert Systems with Applications 42:2026–2035
33. Pan QK, Suganthan PN, Tasgetiren MF, Liang JJ (2010) A self-adaptive global best harmony search algorithm for continuous optimization problems. Appl Math Comput 216:830–848
34. Yadav P, Kumar R, Panda SK, Chang CS (2012) An Intelligent Tuned Harmony Search algorithm for optimisation. Inform Sci 196:47–72
35. Kattan A, Abdullah R (2013) A dynamic self-adaptive harmony search algorithm for continuous optimization problems. Appl Math Comput 219:8542–8567
36. Enayatifar R, Yousefi M, Abdullah AH, Darus AN (2013) LAHS: a novel harmony search algorithm based on learning automata. Commun Nonlinear Sci 18:3481–3497
37. Kumar V, Chhabra JK, Kumar D (2014) Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems. J Comput Sci-Neth 5:144–155
38. Zhou YZ, Li XY, Gao L (2013) A differential evolution algorithm with intersect mutation operator. Appl Soft Comput 13:390–401
39. Schiff JL (2011) Cellular automata: a discrete view of the world. Wiley
40. Alba E, Dorronsoro B (2005) The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans Evol Comput 9:126–142
41. Alba E, Alfonso H, Dorronsoro B (2005) Advanced models of cellular genetic algorithms evaluated on SAT, Gecco. Genetic and Evolutionary Computation Conference 1 and 2:1123–1130
42. Shi Y, Liu HC, Gao L, Zhang GH (2011) Cellular particle swarm optimization. Inform Sci 181:4460–4493
43. Im SS, Yoo DG, Kim JH (2013) Smallest-Small-World Cellular Harmony Search for Optimization of Unconstrained Benchmark Problems
44. Al-Betar MA, Khader AT, Awadallah MA, Alawan MH, Zaqaibeh B (2013) Cellular Harmony Search for Optimization Problems
45. Yang X-S, Deb S (2010) Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation 1:330–343
46. Yang X-S, Deb S (2013) Multiobjective cuckoo search for design optimization. Comput Oper Res 40:1616–1624
47. Yao X, Liu Y, Lin GM (1999) Evolutionary programming made faster. Ieee T Evolut Comput 3:82–102
48. Liang J, Qu B, Suganthan P, Hernández-Díaz AG (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report, 201212
49. Zhu J, Chew DA, Lv S, Wu W (2013) Optimization method for building envelope design to minimize carbon emissions of building operational energy consumption using orthogonal experimental design (OED). Habitat International 37:148–154
50. Rice J (2006) Mathematical statistics and data analysis. Cengage Learning
51. Castelli M, Silva S, Manzoni L, Vanneschi L (2014) Geometric selective harmony search. Inform Sci 279:468–482
52. Zambrano-Bigiarini M, Clerc M, Rojas R (2013) Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements. In: Evolutionary Computation (CEC), IEEE Congress on IEEE, pp 2337–2344
53. Nepomuceno FV, Engelbrecht AP (2013) A self-adaptive heterogeneous pso for real-parameter optimization. In: Evolutionary Computation (CEC), IEEE Congress on IEEE, pp 361–368
54. Qin A, Li X (2013) Differential evolution on the CEC-2013 single-objective continuous optimization testbed. In: Evolutionary Computation (CEC), IEEE Congress on IEEE, pp 1099–1106
55. Zhang C, Gao L (2013) An effective improvement of JADE for real-parameter optimization. In: Advanced Computational Intelligence (ICACI), Sixth International Conference on IEEE, pp 58–63
56. Tanabe R, Fukunaga A (2013) Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: Evolutionary Computation (CEC) IEEE Congress on IEEE, pp 1952–1959



Jin Yi received his Bachelor's degree from form Huazhong University of Science and Technology in 2012. He is currently a Ph.D. candidate in the School of Mechanical Science and Engineering, Huazhong University of Science and Technology. His research interests include computation intelligence and its applications.

Liang Gao received his BS degree in Mechatronic Engineering from Xi'dian University, China, in 1996, and the PhD degree in Mechatronic Engineering from Huazhong University of Science and Technology, China, 2002. Now, as a Professor and Head of the Department of Industrial & Manufacturing System Engineering, Huazhong University of Science and Technology, his research interests include operations research and optimization, scheduling, etc. He had published more than 70 journal papers, including Computer & Operations Research, Expert Systems with Applications, Computers & Industrial Engineering, and so on.

Xinyu Li received his Ph. D degree in Industrial Engineering from Huazhong University of Science and Technology (HUST), China, 2009. He is now an Associate Professor in the Department of Industrial & Manufacturing Systems Engineering, School of Mechanical Science and Engineering, HUST. His research interests include intelligent algorithm, scheduling, etc. He had published more than 25 journal papers, including Computer & Operations Research, Expert Systems with Applications, International Journal of Production Economics and so on.

Jie Gao received his bachelor's degree in Mechanical Design, Manufacturing and Automation in 2014. He is currently pursuing her PhD degree in the State Key Laboratory of Digital Manufacturing Equipment and Technology, University of Science and Technology. His research interests include topology optimization, material microstructure design and Material structure integration design.