symmetry

MDPI

Article

A Dynamic Adjusting Novel Global Harmony Search for Continuous Optimization Problems

Chui-Yu Chiu [1], Po-Chou Shih [2,*] and Xuechao Li [3]

[1] Industrial Engineering and Management, National Taipei University of Technology, Taipei 10632, Taiwan; cychiu@ntut.edu.tw
[2] College of Management, National Taipei University of Technology, Taipei 10632, Taiwan
[3] Department of Computer Science, Concordia University Chicago, Chicago, IL 60305, USA; Xuechao.Li@cuchicago.edu
* Correspondence: pojo0701@hotmail.com; Tel.: +886-921-942-200

Received: 10 July 2018; Accepted: 8 August 2018; Published: 12 August 2018

Abstract: A novel global harmony search (NGHS) algorithm, as proposed in 2010, is an improved algorithm that combines the harmony search (HS), particle swarm optimization (PSO), and a genetic algorithm (GA). Moreover, the fixed parameter of mutation probability was used in the NGHS algorithm. However, appropriate parameters can enhance the searching ability of a metaheuristic algorithm, and their importance has been described in many studies. Inspired by the adjustment strategy of the improved harmony search (IHS) algorithm, a dynamic adjusting novel global harmony search (DANGHS) algorithm, which combines NGHS and dynamic adjustment strategies for genetic mutation probability, is introduced in this paper. Moreover, extensive computational experiments and comparisons are carried out for 14 benchmark continuous optimization problems. The results show that the proposed DANGHS algorithm has better performance in comparison with other HS algorithms in most problems. In addition, the proposed algorithm is more efficient than previous methods. Finally, different strategies are suitable for different situations. Among these strategies, the most interesting and exciting strategy is the periodic dynamic adjustment strategy. For a specific problem, the periodic dynamic adjustment strategy could have better performance in comparison with other decreasing or increasing strategies. These results inspire us to further investigate this kind of periodic dynamic adjustment strategy in future experiments.

Keywords: metaheuristic; global optimization; harmony search algorithm; dynamic adjustment strategy

1. Introduction

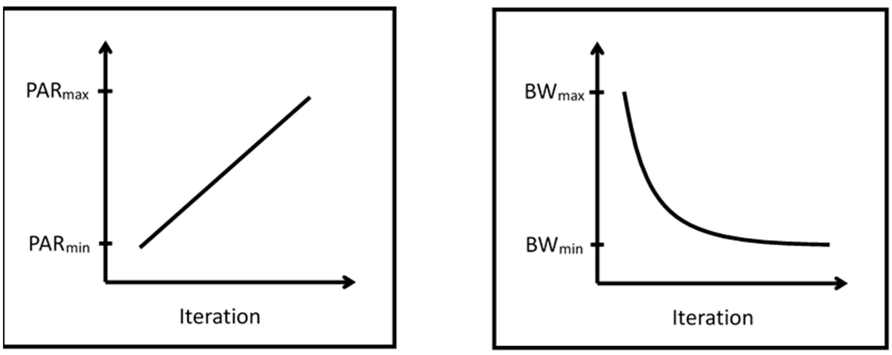
The last two decades have seen a significant increase in research into metaheuristic algorithms. The procedure of a metaheuristic algorithm can be divided into four steps: initialization, movement, replacement, and iteration [1]. The most popular metaheuristic algorithms to date are the particle swarm optimization (PSO) [2,3], genetic algorithm (GA) [4–6], and ant colony optimization (ACO) [7–9].

PSO was introduced by Kennedy and Eberhart in 1995 [10,11]. It imitates the foraging behavior of birds and fish, and provides a population-based search procedure, where each individual is abstracted as a "particle" that flies around in a multidimensional search space. The best positions encountered by a particle and its neighbors determine the particle's trajectory, along with other PSO parameters. In other words, a PSO system attempts to balance exploration and exploitation by combining global and local search methods [12].

The GA has been widely investigated since Holland proposed it in 1960 [13,14]. The GA was developed from Darwinian evolution. Based on the concept of natural genetics and evolutionary principles, GA is a stochastic search technique that can search the near optimum solution in a large and complicated space. As Gordini [15] points out, "the GA differs from other non-linear optimization techniques in that it searches by maintaining a population of solutions from which better solutions are created, rather than making incremental changes to a single solution to a problem." The GA is consisted of three operators: reproduction, crossover, and mutation [16]. Reproduction is a process of survival-of-the-fittest selection. Crossover is the partial swap between two parent strings in order to produce two offspring strings. Mutation is the occasional random inversion of bit values in order to generate a non-recursive offspring. One importance of the GA is that several metaheuristic algorithms have been developed from the GA, such as the honey-bee mating optimization (HBMO) algorithm [17] and the harmony search (HS) algorithm [16].

The harmony search (HS) algorithm is a modern metaheuristic intelligent evolution algorithm [18], and was inspired by the music improvisation process where musicians improvise their instruments' pitches searching for a perfect state of harmony [19]. The HS algorithm simulates the principle of the music improvisation process in the same way that the GA simulates biological evolution, the simulated annealing algorithm (SA) [20] simulates physical annealing, and the PSO algorithm simulates the swarm behavior of birds and fish [18], etc. The HS algorithm has excellent exploitation capabilities. However, the HS algorithm suffers a very serious limitation of premature convergence if one or more initially generated harmonies are in the vicinity of local optimal [21]. As Assad and Deep [22] point out, "The efficiency of evolutionary algorithms depends on the extent of balance between diversification and intensification during the course of the search. Intensification, also called exploitation, is the ability of an algorithm to exploit the search space in the vicinity of the current good solution, whereas diversification, also called exploration, is the process of exploring the new regions of a large search space and thus allows dissemination of the new information into the population. Proper balance between these two contradicting characteristics is a must to enhance the performance of the algorithm."

Therefore, in order to eradicate the aforementioned limitation, several improved HS algorithms have been proposed, such as the improved harmony search (IHS) algorithm [23], the self-adaptive global best harmony search (SHGS) [24], the novel global harmony search (NGHS) [25], the intelligent global harmony search (IGHS) algorithm [19], and so on. Of these algorithms, the IHS algorithm is the first to propose using the adjustment strategy to tune the pitch adjusting rate (PAR) and bandwidth (BW) parameters. In the HS algorithm, according to the value of PAR, the musicians will determine to adjust their instruments' pitches or not. Besides, the musicians will adjust the pitches within the BW distance. The PAR and BW values change dynamically with generation number, as shown in Figure 1. In Mahdavi's paper [23], the adjustment strategy was proofed; it can enhance the searching ability of the harmony search algorithm. In other words, the importance of the appropriate parameters was proofed in his paper.



(**a**) (**b**)

Figure 1. (**a**) Variation of pitch adjusting rate (PAR) versus iteration number; (**b**) Variation of bandwidth (BW) versus iteration number.

Appropriate parameters can enhance the searching ability of a metaheuristic algorithm; their importance has been described in many studies. First, Pan et al. demonstrated that a good set of parameters can enhance an algorithm's ability to search for the global optimum or near optimum region with a high convergence rate [19,24]. Second, in the NGHS algorithm, the new trial solutions are generated by the parameter $step_j$. Therefore, Zou et al. [25,26] showed that the most reasonable design for $step_j$ in the NGHS algorithm can guarantee that the proposed algorithm has strong global search ability in the early optimization stage, and strong local search ability in the late optimization stage. In addition, a dynamically adjusted $step_j$ maintains a balance between the global search and the local search. In another paper, Zou et al. [27] demonstrated that an appropriate harmony memory considering rate (HMCR) and PAR value in the SGHS algorithm can be gradually learned to suit the particular problem and the particular phases of the search process. In addition, there is no single choice for the genetic mutation probability (p_m) in the NGHS algorithm; it should be adjusted according to practical optimization problems. Last, Valian, Tavakoli, and Mohanna [28] observed that there can be no single choice for HMCR in the IGHS algorithm, and it should be adjusted according to the given optimization problems.

However, in the NGHS algorithm, the value of the genetic mutation probability (p_m) is a fixed value that is given in the initialization step. According to the result of Mahdavi's paper [23], we supposed that the adjustment strategy could enhance the searching ability. Therefore, a dynamic adjusting novel global harmony search (DANGHS) algorithm was proposed in this paper. In the DANGHS algorithm, the mutation probability adjusts dynamically with the generation number by the adjustment strategy. However, we can adjust the mutation probability using different strategies. Therefore, this paper used 16 different strategies in the DANGHS algorithm in 14 well-known benchmark optimization problems. In other words, the performance of different strategies in the DANGHS algorithm for different problems was investigated. Besides, in general, one important characteristic of the metaheuristic algorithm is to be fast and efficient. A better metaheuristic algorithm cannot only search the more exact solution but also use less iterations than other algorithms. Therefore, we discuss the efficiency of the DANGHS algorithm in this paper. According to the numerical results, the DANGHS algorithm had better searching performance in comparison with other HS algorithms in most problems.

The remainder of this paper is arranged as follows. In Section 2, the HS, IHS, SGHS, and NGHS algorithms are introduced. Section 3 describes the DANGHS algorithm. A large number of experiments are carried out to test and compare the performance of 16 different strategies in the DANGHS algorithm in Section 4. Conclusions and suggestions for future research are given in Section 5.

2. HS, IHS, SGHS, and NGHS

In this section, the HS, the IHS, the SGHS, and the NGHS are reviewed.

2.1. Harmony Search Algorithm

The HS algorithm was proposed by Geem, Kim, and Loganathan in 2001 [16]. HS is similar in concept to other metaheuristic algorithms such as GA, PSO, and ACO in terms of combining the rules of randomness to imitate the process that inspired it. However, HS draws its inspiration not from biological or physical processes but from the improvisation process of musicians, such as that found in a Jazz trio [19,29].

In the musical improvisation process, each musician sounds any pitch within a possible range, and then together they make a single harmony. If all the pitches make a pleasing harmony, the experience is stored in each player's memory, and the possibility of making a more pleasing harmony the next time is increased [30]. Similarly, in engineering optimization, each decision variable initially chooses any value within a possible range, together making one solution vector [27]. In the HS algorithm, each harmony, which means the trial solution for the problem, is represented by a D-dimension real vector, and a pleasing harmony means the good trial solution for the problem [19].

If all the decision variable values make a good solution, then that experience is stored in each variable's memory, and the possibility of making a good solution the next time is also increased [27]. Figure 2 shows the comparison between music improvisation and engineering optimization. In Figure 2, there is a Jazz trio consisting of three musicians. Each musician plays an instrument at the same time to make a single harmony. The pitches of the three instruments mean the values of the three decision variables.

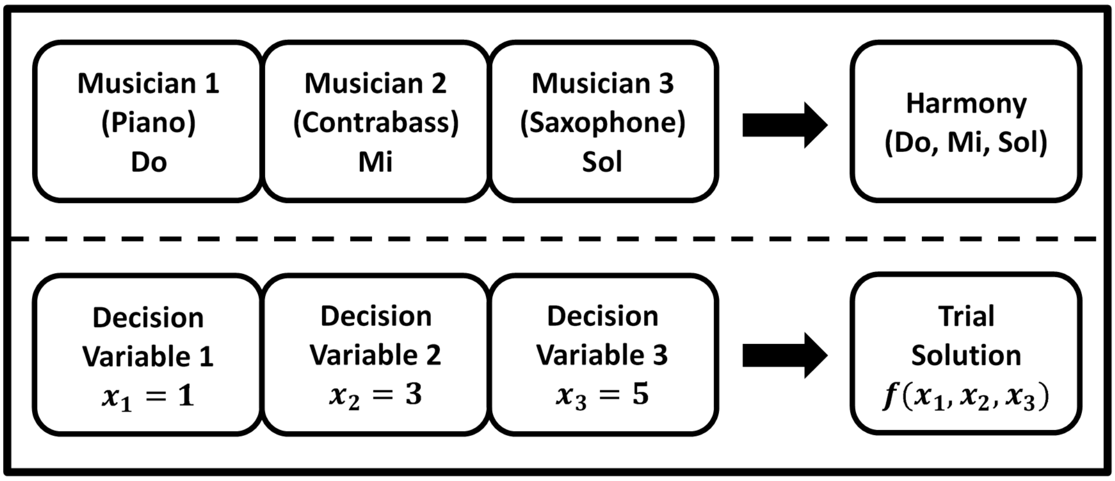


Figure 2. Comparison between music improvisation and engineering optimization.

In general, the HS algorithm works as follows [27]:

Step 1. Initialization: the algorithm and problem parameters

In this step, the parameters of the HS algorithm are determined. The parameters are the harmony memory size (m), the harmony memory considering rate (HMCR), the pitch adjusting rate (PAR), the bandwidth (BW), the current iteration (k = 1), and the maximum number of iterations (NI). Furthermore, the D-dimensional optimization problem is defined as Minimize f(x) subject to $x_{jL} \leq x_j \leq x_{jU}$ $(j = 1,\ 2,\ \ldots,\ D)$. x_{jL} and x_{jU} are the lower and upper bounds for decision variables x_j.

Step 2. Initialization: the decision variable values and the harmony memory

The initial decision variable values $x_{ij}^{k=0}(\mathrm{i} = 1,\ 2,\ \ldots,\ \mathrm{m})$ are generated by Equation (1). The harmony memory (HM) is as shown in Equation (2).

$$x_{ij}^{0} = x_{jL} + r \times \left(x_{jU} - x_{jL}\right) \tag{1}$$

$$\mathrm{HM} = \begin{bmatrix} x_{11}^{0} & x_{12}^{0} & \cdots & x_{1D}^{0} \\ x_{21}^{0} & x_{22}^{0} & \cdots & x_{2D}^{0} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1}^{0} & x_{m2}^{0} & \cdots & x_{mD}^{0} \end{bmatrix} \tag{2}$$

In Equation (1), r is the uniformly generated random numbers in the region of [0, 1].

Step 3. Movement: improvise a new harmony

Movement step is the most important step of any algorithm. The performance of global exploration and local exploitation are related to the design of the movement step. In the HS algorithm, the movement step is improvisation. The new harmony vector $x^{k+1} = \left(x_1^{k+1}, x_2^{k+1}, \ldots, x_D^{k+1}\right)$ is generated by memory consideration, pitch adjustment, and random selection mechanisms in this step. The HS movement steps (Pseudocode 1) are shown in Algorithm 1.

Algorithm 1 The Movement Steps of HS (Pseudocode 1)

For j = 1 to D **do**
  If $r_1 \leq$ HMCR **then**
    $x_j^{k+1} = x_{ij}^k$ % memory consideration
    If $r_2 \leq$ PAR **then**
      $x_j^{k+1} = x_j^{k+1} - BW + r_3 \times 2 \times BW$ % pitch adjustment
      If $x_j^{k+1} > x_{jU}$ **then**
        $x_j^{k+1} = x_{iU}$
      Else if $x_j^{k+1} < x_{jL}$ **then**
        $x_j^{k+1} = x_{iL}$
      End
    End
  Else
    $x_j^{k+1} = x_{jL} + r_4 \times (x_{jU} - x_{jL})$ % random selection
  End
End

Here, x_j^{k+1} is the j^{th} component of x^{k+1}. i is an uniformly generated random number in [1, m], and x_{ij}^k is the j^{th} component of the i^{th} candidate solution vector in the HM. r_1, r_2, r_3 and r_4 are the uniformly generated random numbers in the region of [0, 1], and BW is a given distance bandwidth.

Step 4. Replacement: update harmony memory

If the fitness value of the new harmony vector x^{k+1} is better than that of the worst harmony in the HM, replace the worst harmony vector by x^{k+1}.

Step 5. Iteration: check the stopping criterion

If the stopping criterion (maximum number of iterations NI) is satisfied, the computation is terminated; otherwise, the current iteration k = k + 1 and go back to step 3.

2.2. Improved Harmony Search Algorithm

The IHS algorithm was proposed by Mahdavi, Fesanghary, and Damangir in 2007 for solving optimization problems [23]. In their paper, they noted that PAR and BW are very important parameters in the HS algorithm when fine-tuning optimized solution vectors, and can be potentially useful in adjusting the convergence rate of the algorithm to the optimal solution. Fine adjustment of these parameters is therefore of particular interest. The key difference between the IHS and the traditional HS method is thus in the way PAR and BW are adjusted in each iteration by Equations (3) and (4):

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

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

In Equation (3), PAR^k is the pitch adjustment rate in the current iteration k; PAR_{min} and PAR_{max} are the minimum and maximum adjustment rates, respectively. In Equation (4), BW^k is the distance bandwidth in current iteration k, BW_{min} is the minimum bandwidth, and BW_{max} is the maximum bandwidth. Figure 1 shows that the PAR and BW values change dynamically with the iteration number.

2.3. Self-Adaptive Global Best Harmony Search Algorithm

The SGHS algorithm was presented by Pan et al. in 2010 for continuous optimization problems [24].

In the SGHS algorithm, the HMCR and PAR were dynamically adapted by the normal distribution and the BW was adjusted in each iteration. The value of $HMCR^k$ was generated by the mean $HMCR_m$ and the standard deviation. In the same way, the value of PAR^k was generated by the mean PAR_m and the standard deviation. Pan et al. assumed that the dynamic mean $HMCR_m$ is in the range of [0.9, 1.0] and the static standard deviation is 0.01; the dynamic mean PAR_m. is in the range of [0.0, 1.0] and the static standard deviation is 0.05. Furthermore, the $HMCR^k$ and PAR^k were recorded by their historic values when the generated harmony successfully replaced the worst harmony in the harmony memory. After a specified learning period (LP), the $HMCR_m$ and PAR_m were recalculated by averaging all the recorded $HMCR^k$ and PAR^k values during this period respectively. In the subsequent iterations, new $HMCR^k$ and PAR^k values were generated with the new mean $HMCR_m$ and PAR_m and the given standard deviation. In addition, the BW^k is decreased in each iteration by Equation (5).

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

In general, the SGHS algorithm works as follows:

Step 1. Initialization: the problem and algorithm parameters

Set parameters m, LP, NI, BW_{max}, BW_{min}, $HMCR_m$, PAR_m, the current iteration k = 1, and iteration counter lp = 1.

Step 2. Initialization: the decision variable values and the harmony memory

The initial decision variable values $x_{ij}^{k=0}(\mathrm{i} = 1,\ 2,\ \ldots,\ \mathrm{m})$ is generated by Equation (1). The harmony memory (HM) is as shown in Equation (2).

Step 3. Movement: generate the algorithm parameters

Generate $HMCR^k$ and PAR^k with $HMCR_m$ and PAR_m by the normal distribution respectively. Generate BW^k with BW_{max} and BW_{min} by Equation (5).

Step 4. Movement: improvise a new harmony

Improvise a new harmony x^{k+1}. The SGHS movement step (Pseudocode 2) is shown in Algorithm 2.

Algorithm 2 The Movement Steps of SGHS (Pseudocode 2) [24]

For j = 1 to D **do**
  If $r_1 \leq HMCR^k$ **then**
    $x_j^{k+1} = x_{ij}^k - BW^k + r_2 \times 2 \times BW^k$
    If $x_j^{k+1} > x_{jU}$ **then**
      $x_j^{k+1} = x_{iU}$
    Else if $x_j^{k+1} < x_{jL}$ **then**
      $x_j^{k+1} = x_{iL}$
    End
    If $r_3 \leq PAR^k$ **then**
      $x_j^{k+1} = x_{best,j}^k$
    End
  Else
    $x_j^{k+1} = x_{jL} + r_4 \times (x_{jU} - x_{jL})$ % random selection
  End
End

Here, x_j^{k+1} is the j[th] component of x^{k+1}. i is an uniformly generated random number in [1, m], and x_{ij}^k is the j[th] component of the i[th] candidate solution vector in the HM. $x_{best,j}^k$ is the j[th] component of the best candidate solution vector in the HM. r_1, r_2, r_3 and r_4 are uniformly generated random numbers in [0, 1]. r_1 is used for position updating, r_2 determines the distance of the BW, r_3 is used for pitch adjustment, and r_4 is used for random selection.

Step 5. Replacement: update harmony memory

If the fitness value of the new harmony vector x^{k+1} is better than that of the worst harmony in the HM, replace the worst harmony vector by x^{k+1} and record the values of $HMCR^k$ and PAR^k.

Step 6. Replacement: update $HMCR_m$ and PAR_m

If lp = LP, recalculate $HMCR_m$ and PAR_m by averaging all the recorded $HMCR^k$ and PAR^k values respectively and reset lp = 1; otherwise, lp = lp +1.

Step 7. Iteration: check the stopping criterion

If NI is completed, return the best harmony vector x_{best} in the HM; otherwise, the current iteration k = k + 1 and go back to step 3.

2.4. Novel Global Harmony Search Algorithm

The NGHS algorithm [25,26] is an improved algorithm that combines HS, PSO, and GA. A prominent characteristic of PSO is that individual particles attempt to imitate the social experience. It means the particles are affected by other better particles in the PSO algorithm. A prominent characteristic of GA is that it is possible for the trial solution to escape from the local optimum by mutation. In other words, NGHS tries to generate a new solution by moving the worst solution toward the best solution or by mutation.

Figure 3 is used to illustrate the principle of position updating. $step_j = \left| x_{best,j}^k - x_{worst,j}^k \right|$ is defined as an adaptive step of the j[th] decision variable. This adaptive step can dynamically balance the performance of global exploration and local exploitation in the NGHS algorithm. As Zou et al. [26] points out, "In the early stage of optimization, all solution vectors are sporadic in the solution space, so most adaptive steps are large, and most trust regions are wide, which is beneficial to the global search of NGHS. However, in the late stage of optimization, all non-best solution vectors are inclined to move to the global best solution vector, so most solution vectors are close to each other. In this case, most adaptive steps are small and most trust regions are narrow, which is beneficial to the local search of NGHS."

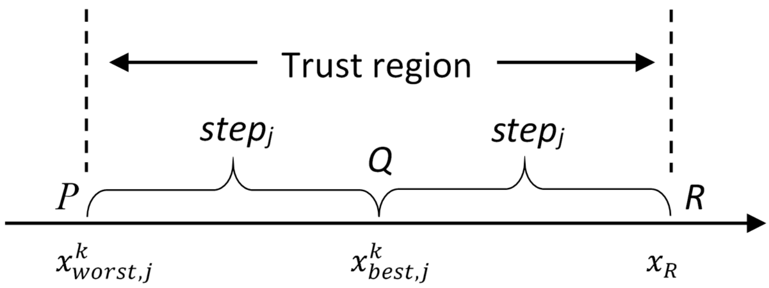


Figure 3. Schematic diagram of position updating.

According to this prominent characteristic, NGHS modifies the movement step of HS therefore the NGHS algorithm can imitate the current best harmony in the HM. In general, the NGHS algorithm works as follows:

Step 1. Initialization: the algorithm and problem parameters

(1) Set parameters m, NI, and the current iteration k = 1.
(2) The genetic mutation probability (p_m) is included in NGHS, while the harmony memory considering rate (HMCR), pitch adjusting rate (PAR) and the bandwidth (BW) are excluded from NGHS.

Step 2. Initialization: the decision variable values and the harmony memory

The initial decision variable values $x_{ij}^{k=0}(\mathrm{i} = 1,\ 2,\ \ldots,\ \mathrm{m})$ are generated by Equation (1). The HM is as shown in Equation (2).

Step 3. Movement: improvise a new harmony

NGHS modifies the movement step in HS. The NGHS movement step (Pseudocode 3) is shown in Algorithm 3.

Algorithm 3 The Movement Steps of NGHS (Pseudocode 3) [25–27].

For j = 1 to D **do**
  $\mathrm{x_R} = 2 \times \mathrm{x}^{\mathrm{k}}_{\mathrm{best,j}} - \mathrm{x}^{\mathrm{k}}_{\mathrm{worst,j}}$
  If $\mathrm{x_R} > \mathrm{x_{jU}}$ **then**
    $\mathrm{x_R} = \mathrm{x_{jU}}$
  Else if $\mathrm{x_R} < \mathrm{x_{jL}}$ **then**
    $\mathrm{x_R} = \mathrm{x_{jL}}$
  End
  $\mathrm{x}_{\mathrm{j}}^{\mathrm{k+1}} = \mathrm{x}^{\mathrm{k}}_{\mathrm{worst,j}} + \mathrm{r_1} \times (\mathrm{x_R} - \mathrm{x}^{\mathrm{k}}_{\mathrm{worst,j}})$ % position updating
  If $\mathrm{r_2} \le \mathrm{p_m}$ **then**
    $\mathrm{x}_{\mathrm{j}}^{\mathrm{k+1}} = \mathrm{x_{jL}} + \mathrm{r_3} \times (\mathrm{x_{jU}} - \mathrm{x_{jL}})$ % genetic mutation
  End
End

Here, $x^k_{best,j}$ and $x^k_{worst,j}$ are the best harmony and the worst harmony in the HM, respectively. r_1, r_2 and r_3 are uniformly generated random numbers in [0, 1]. r_1 is used for position updating, r_2 determines whether NGHS should carry out genetic mutation, and r_3 is used for genetic mutation.

Genetic mutation with a small probability is carried out for the current worst harmony in the HM after position updating [25–27].

Step 4. Replacement: update harmony memory

NGHS replaces the worst harmony $x^k_{worst,j}(\mathrm{j} = 1,\ 2,\ \ldots,\ \mathrm{D})$ in the HM by the new harmony x^{k+1}, even if the new harmony is worse than the worst harmony.

Step 5. Iteration: check the stopping criterion

If the stopping criterion (maximum number of iterations NI) is satisfied, the computation is terminated; otherwise, the current iteration k = k + 1 and go back to step 3.

3. Dynamic Adjusting Novel Global Harmony Search (DANGHS) Algorithm

Appropriate parameters can enhance the searching ability of a metaheuristic algorithm. Inspired by this concept, a dynamic adjusting NGHS (DANGHS) is presented in this section. In the DANGHS, the genetic mutation probability (p_m) is dynamically adjusted in each iteration. However, we can enhance the searching ability of the NGHS algorithm by many kinds of dynamic adjustment strategies. Therefore, we introduced 16 dynamic adjustment strategies in this paper. All 16 strategies are shown as follows, and Figures 4–6 are used to illustrate the 16 strategies.

(1) Straight linear increasing strategy (Straight_1):

The genetic mutation probability is increased by Equation (6), which is a linear function.

$$p_m^k = p_{m\_min} + \frac{(p_{m\_max} - p_{m\_min})}{NI} \times k. \tag{6}$$

Here, p_{m_min} is the minimum genetic mutation probability, and p_{m_max} is the maximum genetic mutation probability.

(2) Straight linear decreasing strategy (Straight_2):

The genetic mutation probability is decreased by Equation (7), which is a linear function.

$$p_m^k = p_{m\_max} + \frac{(p_{m\_min} - p_{m\_max})}{NI} \times k \tag{7}$$

(3) Threshold linear prior increasing strategy (Threshold_1):

The genetic mutation probability is increased by Equation (8), which is a linear function with a threshold. The genetic mutation probability is raised before the threshold, but the genetic mutation probability is a fixed maximum value after the threshold.

$$p_m^k = \begin{cases} p_{m\_min} + \frac{P_{m\_max} - P_{m\_min}}{NI} \times 2k & if\ k < NI/2 \\ p_{m\_max} & if\ k \geq NI/2 \end{cases} \tag{8}$$

(4) Threshold linear prior decreasing strategy (Threshold_2):

The genetic mutation probability is decreased by Equation (9), which is a linear function with a threshold. The genetic mutation probability is reduced before the threshold, but the genetic mutation probability is a fixed minimum value after the threshold.

$$p_m^k = \begin{cases} p_{m\_max} + \frac{P_{m\_min} - P_{m\_max}}{NI} \times 2k & if\ k < NI/2 \\ p_{m\_min} & if\ k \geq NI/2 \end{cases} \tag{9}$$

(5) Threshold linear posterior increasing strategy (Threshold_3):

The genetic mutation probability is increased by Equation (10), which is a linear function with a threshold. The genetic mutation probability is a fixed minimum value before the threshold, but the genetic mutation probability is raised after the threshold.

$$p_m^k = \begin{cases} p_{m\_min} & if\ k < NI/2 \\ p_{m\_min} + \frac{P_{m\_max} - P_{m\_min}}{NI} \times 2k & if\ k \geq NI/2 \end{cases} \tag{10}$$

(6) Threshold linear posterior decreasing strategy (Threshold_4):

The genetic mutation probability is decreased by Equation (11), which is a linear function with a threshold. The genetic mutation probability is a fixed maximum value before the threshold, but the genetic mutation probability is reduced after the threshold.

$$p_m^k = \begin{cases} p_{m\_max} & if\ k < NI/2 \\ p_{m\_max} + \frac{P_{m\_min} - P_{m\_max}}{NI} \times 2k & if\ k \geq NI/2 \end{cases} \tag{11}$$

(7) Natural exponential increasing strategy (Exponential_1):

The genetic mutation probability is increased by Equation (12), which is a non-linear function.

$$p_m^k = p_{m\_min} \times e^{(\ln(\frac{p_{m\_max}}{p_{m\_min}}) \times k/NI)} \tag{12}$$

(8) Natural exponential decreasing strategy (Exponential_2):

The genetic mutation probability is decreased by Equation (13), which is a non-linear function.

$$p_m^k = p_{m\_max} \times e^{(\ln(\frac{p_{m\_min}}{p_{m\_max}}) \times k/NI)} \tag{13}$$

(9) Exponential increasing strategy:

The genetic mutation probability is increased by Equation (14), which is a non-linear function. We can control the increasing rate by the modification rate (*mr*).

$$p_m^k = p_{m\_min} + (p_{m\_max} - p_{m\_min}) \times mr^{(NI-k)/NI} \tag{14}$$

In this paper, the *mr* is equal to 0.01 or 0.001. Therefore, in this paper, the 9th strategy (Exponential_3) is the exponential increasing strategy with $mr = 0.01$, and the 10th strategy (Exponential_5) is the exponential increasing strategy with $mr = 0.001$.

(10) Exponential decreasing strategy:

The genetic mutation probability is decreased by Equation (15), which is a non-linear function. We can control the decreasing rate by the modification rate (*mr*).

$$p_m^k = p_{m\_min} + (p_{m\_max} - p_{m\_min}) \times mr^{k/NI} \tag{15}$$

In this paper, the *mr* is equal to 0.01 or 0.001. Therefore, in this paper, the 11th strategy (Exponential_4) is the exponential decreasing strategy with $mr = 0.01$, and the 12th strategy (Exponential_6) is the exponential decreasing strategy with $mr = 0.001$.

(11) Concave cosine strategy:

The genetic mutation probability is changed by Equation (16), which is a periodic function. The shape of this function is a concave, and we can control the cycle time of this function by the coefficient of cycle (cc).

$$p_m^k = \frac{p_{m\_max} + p_{m\_min}}{2} + \frac{p_{m\_max} - p_{m\_min}}{2} \times \cos\frac{k \times cc \times 2\pi}{NI} \tag{16}$$

In this paper, the *cc* is equal to 1 or 3. Therefore, in this paper, the 13th strategy (Cosine_1) is the concave cosine strategy with $cc = 1$, and the 14th strategy (Cosine_3) is the concave cosine strategy with $cc = 3$.

(12) Convex cosine strategy:

The genetic mutation probability is changed by Equation (17), which is a periodic function. The shape of this function is a convex, and we can control the cycle time of this function by the coefficient of cycle (cc).

$$p_m^k = \frac{p_{m\_max} + p_{m\_min}}{2} - \frac{p_{m\_max} - p_{m\_min}}{2} \times \cos\frac{k \times cc \times 2\pi}{NI} \tag{17}$$

In this paper, the *cc* is equal to 1 or 3. Therefore, in this paper, the 15th strategy (Cosine_2) is the convex cosine strategy with $cc = 1$, and the 16th strategy (Cosine_4) is the convex cosine strategy with $cc = 3$.

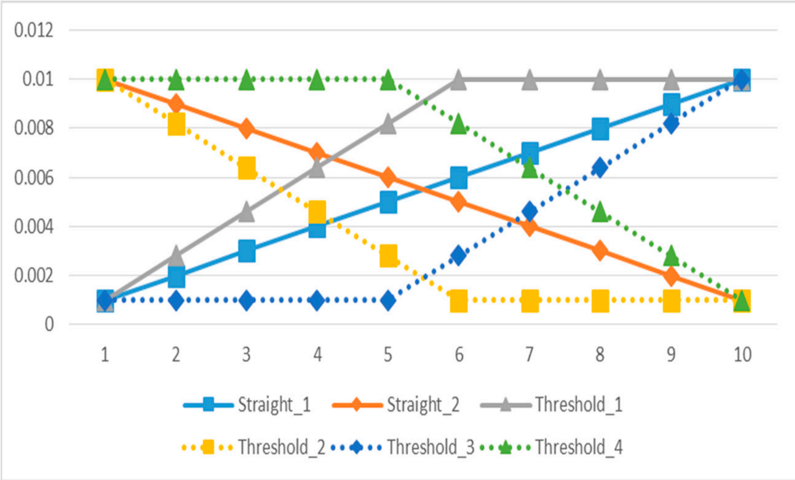


Figure 4. Straight linear and threshold linear strategies.

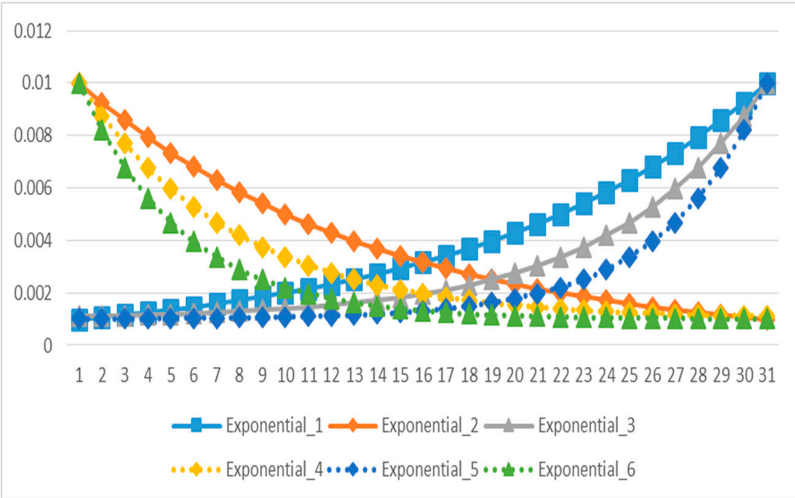


Figure 5. Exponential strategies.

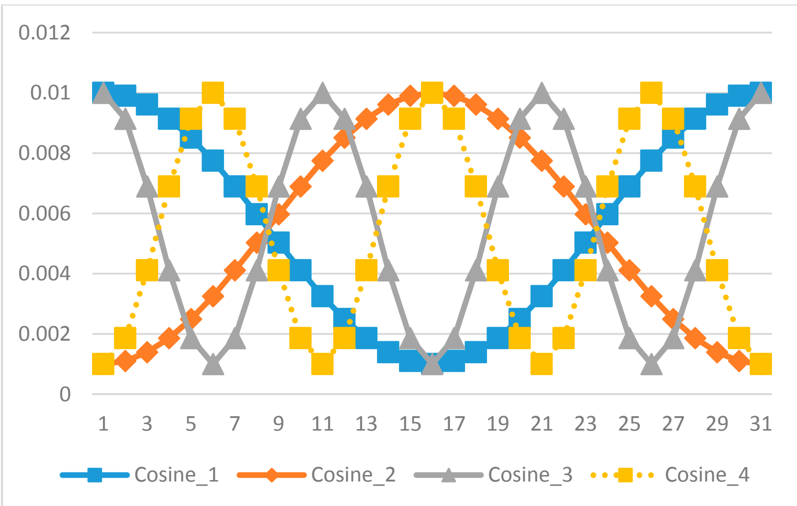


Figure 6. Concave cosine and convex cosine strategies.

In general, the DANGHS algorithm works as follows:

Step 1. Initialization: the problem and algorithm parameters

The parameters are the harmony memory size (m), the current iteration k = 1, and the maximum number of iterations (NI).

Step 2. Initialization: the decision variable values and the harmony memory

The initial decision variable values $x_{ij}^{k=0}(\mathrm{i} = 1, 2, \ldots, \mathrm{m})$ is generated by Equation (1). The HM is as shown in Equation (2).

Step 3. Movement: generate the algorithm parameters

Generate the genetic mutation probability (p_m^k) in each iteration by dynamic adjustment strategies.

Step 4. Movement: improvise a new harmony

The DANGHS movement step (Pseudocode 4) is shown in Algorithm 4.

Algorithm 4 The Movement Steps of DANGHS (Pseudocode 4)

For $\mathrm{j} = 1$ to D **do**
  If $\mathrm{r}_1 > \mathrm{p}_\mathrm{m}^\mathrm{k}$ **then**
    $\mathrm{x}_\mathrm{R} = 2 \times \mathrm{x}_{\mathrm{best,j}}^\mathrm{k} - \mathrm{x}_{\mathrm{worst,j}}^\mathrm{k}$
    If $\mathrm{x}_\mathrm{R} > \mathrm{x}_{\mathrm{jU}}$ **then**
      $\mathrm{x}_\mathrm{R} = \mathrm{x}_{\mathrm{jU}}$
    Else if $\mathrm{x}_\mathrm{R} < \mathrm{x}_{\mathrm{jL}}$ **then**
      $\mathrm{x}_\mathrm{R} = \mathrm{x}_{\mathrm{jL}}$
    End
    $\mathrm{x}_\mathrm{j}^{\mathrm{k}+1} = \mathrm{x}_{\mathrm{worst,j}}^\mathrm{k} + \mathrm{r}_2 \times (\mathrm{x}_\mathrm{R} - \mathrm{x}_{\mathrm{worst,j}}^\mathrm{k})$ % position updating
  Else
    $\mathrm{x}_\mathrm{j}^{\mathrm{k}+1} = \mathrm{x}_{\mathrm{jL}} + \mathrm{r}_3 \times (\mathrm{x}_{\mathrm{jU}} - \mathrm{x}_{\mathrm{jL}})$ % genetic mutation
  End
End

Here, $x_{best,j}^k$ and $x_{worst,j}^k$ are the best harmony and the worst harmony in the HM, respectively. r_1, r_2 and r_3 are uniformly generated random numbers in [0, 1]. r_1 determines whether DANGHS should carry out genetic mutation, r_2 is used for position updating, and r_3 is used for genetic mutation.

Step 5. Replacement: update harmony memory

DANGHS replaces the worst harmony $x_{worst,j}^k(\mathrm{j} = 1, 2, \ldots, \mathrm{D})$ in the HM by the new harmony x^{k+1}, even if the new harmony is worse than the worst harmony.

Step 6. Iteration: check the stopping criterion

If the stopping criterion (maximum number of iterations NI) is satisfied, terminate the computation and return the best harmony vector x_{best} in the HM; otherwise, the current iteration k = k + 1 and go back to step 3.

4. Experiments and Analysis

In order to verify the performance of the 16 dynamic adjustment strategies in the DANGHS algorithm, 14 well-known benchmark optimization problems [24,28,31] are considered, as shown in Table 1. This study used Python 3.6.2 (64-bit) as the complier to write the program for finding the solution. The solution-finding equipment was an Intel Core (TM) i7-4720HQ (2.6 GHz) CPU, 8G of memory, and Windows 10 home edition (64-bit) OS.

Table 1. 14 well-known benchmark optimization problems.

Name	Function	Search Space	Optimum
f_1 Sphere function	$\min f(x_i) = \sum_{i=1}^{N} x_i^2$	$[-100, 100]^n$	0
f_2 Step function	$\min f(x_i) = \sum_{i=1}^{N} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^n$	0
f_3 Schwefel's problem 2.22	$\min f(x_i) = \sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	$[-10, 10]^n$	0
f_4 Rotated hyper-ellipsoid function	$\min f(x_i) = \sum_{i=1}^{N} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^n$	0
f_5 Griewank function	$\min f(x_i) = \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$	0
f_6 Ackley's function	$\min f(x_i) = 20 + e - 20\exp\left(-0.2\sqrt{\sum_{i=1}^{N} x_i^2/n}\right) - \exp\left(\sum_{i=1}^{N} \cos(2\pi x_i)/n\right)$	$[-32, 32]^n$	0
f_7 Rosenbrock function	$\min f(x_i) = \sum_{i=1}^{N-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right)$	$[-30, 30]^n$	0
f_8 Rastrigin function	$\min f(x_i) = \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^n$	0
f_9 Schwefel's problem 2.26	$\min f(x_i) = 418.9829N - \sum_{i=1}^{N} \left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$	$[-500, 500]^n$	0
f_{10} Shifted Sphere function	$\min f(x_i) = \sum_{i=1}^{N} z_i^2 - 450$	$[-100, 100]^n$	−450
f_{11} Shifted Rotated hyper-ellipsoid function	$\min f(x_i) = \sum_{i=1}^{N} (\sum_{j=1}^{i} z_i)^2 - 450$	$[-100, 100]^n$	−450
f_{12} Shifted Rotated Griewank function	$\min f(x_i) = \frac{1}{4000} \sum_{i=1}^{N} z_i^2 - \prod_{i=1}^{N} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 - 180$	$[-600, 600]^n$	−180
f_{13} Shifted Rosenbrock function	$\min f(x_i) = \sum_{i=1}^{N-1} \left(100\left(z_{i+1} - z_i^2\right)^2 + (1 - z_i)^2\right) + 390$	$[-30, 30]^n$	390
f_{14} Shifted Rastrigin function	$\min f(x_i) = \sum_{i=1}^{N} (z_i^2 - 10\cos(2\pi z_i) + 10) - 330$	$[-5.12, 5.12]^n$	−330

Problems 1–4, 10 and 11, which are Sphere function, Step function, Schwefel's problem 2.22, Rotated hyper-ellipsoid function, Shifted Sphere function, and Shifted Rotated hyper-ellipsoid function, are unimodal problems. Problems 5–9 and 12–14, which are Griewank function, Ackley's function, Rosenbrock function, Rastrigin function, Schwefel's problem 2.26, Shifted Rotated Griewank function, Shifted Rosenbrock function, and Shifted Rastrigin function, are difficult multimodal problems; i.e., there are several local optima in these problems and the number of local optima increases with the problem dimension (D) [24].

In order to verify the performance of the DANGHS algorithm, this paper compared the extensive experiment results of the DANGHS algorithm with other different HS algorithms. In the experiments, the parameters of the compared HS algorithms are shown in Table 2 [28].

Table 2. Parameters of compared harmony search (HS) algorithms.

Algorithm	m [1]	HMCR [2]	PAR [3]	BW [4]	LP [5]	p_m [6]
HS	5	0.9	0.3	0.01	–	–
IHS	5	0.9	$PAR_{min} = 0.01$ $PAR_{max} = 0.99$	$BW_{max} = \left(x_{jU} - x_{jL}\right)/20$ $BW_{min} = 0.0001$	–	–
SGHS	5	$HMCR_m = 0.98$	$PAR_m = 0.9$	$BW_{max} = \left(x_{jU} - x_{jL}\right)/10$ $BW_{min} = 0.0005$	100	–
NGHS	5	–	–	–	–	0.005
DANGHS	5	–	–	–	–	$P_{min} = 0.001$ $P_{max} = 0.010.$

[1] m: the harmony memory size; [2] HMCR: the harmony memory considering rate; [3] PAR: the pitch adjusting rate; [4] BW: the bandwidth; [5] LP: the learning period; [6] p_m: the genetic mutation probability.

In all HS algorithms, the harmony memory size (m) is 5. For each problem, two different dimension sizes (D) are tested, and they are equal to 30 and 100. Therefore, the iteration number are equal to 60,000 and 150,000, respectively. Thirty independent experiments (n) are carried out for each problem. The experimental results obtained using the 16 proposed adjustment strategies in the DANGHS algorithms and those obtained using different HS algorithms are shown in Tables 3 and 4, respectively. In the two tables, SD represents the standard deviation.

In Table 3, several experimental results are given. First of all, the best results given by the same strategy for different dimension sizes are obtained for problems 1, 3, 6, 9 and 13. Among these problems, the decreasing strategy can find the best objective function value for problems 1, 3, 6, and 9. According to the experimental results, the exponential decreasing strategy with mr = 0.001 (Exponential_6) can find the best objective function value for problems 1 (1.8344×10^{-31}; 1.2209×10^{-14}), 3 (1.9511×10^{-18} 7.9778×10^{-9}) and 6 (9.6308×10^{-14}; 9.3030×10^{-9}); the threshold linear prior decreasing strategy (Threshold_2) can find the best objective function value for problem 9 (3.8183×10^{-4}; 1.2728×10^{-3}). More specifically, the convex cosine strategy with k = 3 (Cosine_4), which is the periodic strategy, can find the best objective function value for problem 13 (3.9875×10^{2}; 5.3644×10^{2}).

On the other hand, the best results given by different strategies for different dimension sizes are obtained for problems 4, 5, 7, 8, 10, 11, 12, and 14. Among these problems, the increasing strategy can find the best objective function value for problem 7. According to the experimental results, the straight linear increasing strategy (Straight_1) can find the best objective function value for problem 7 with D = 30 (1.0089×10^{1}). However, the threshold linear posterior increasing strategy (Threshold_3) can find the best objective function value for problem 7 with D = 100 (6.1559×10^{1}). Besides, the decreasing strategy can find the best objective function value for problems 4, 5, 10, 11, 12, and 14. According to the experimental results, the threshold linear posterior decreasing strategy (Threshold_4) can find the best objective function value for problems 4 (6.0249×10^{1}), 5 (3.1209×10^{-2}) and 11 (-3.7419×10^{2}) with D = 30. However, the natural exponential decreasing strategy (Exponential_2) can find the best objective function value for problems 4 (8.6301×10^{3}), 5 (6.4754×10^{-3}), and 11 (1.1471×10^{4}) with D = 100. The natural exponential decreasing strategy (Exponential _2) can find the best objective function value for problem 10 with D = 30 (-4.5000×10^{2}). However, the threshold linear prior decreasing strategy (Threshold_2) can find the best objective function value for problem 10 with D = 100 (-4.5000×10^{2}). The straight linear decreasing strategy (Straight_2) can find the best objective function value for problem 12 with D = 30 (-1.7821×10^{2}). However, the exponential decreasing strategy with mr = 0.001 (Exponential_6) can find the best objective function value for problem 12 with D = 100 (-1.6037×10^{2}). The straight linear decreasing strategy (Straight_2) can find the best objective function value for problem 14 with D = 30 (-3.3000×10^{2}). However, the exponential decreasing strategy with mr = 0.01 (Exponential_4) can find the best objective function value for problem 14 with D = 100 (-3.2997×10^{2}).

Particularly, for problem 8, the decreasing strategy can find the best objective function value when D = 30, however the increasing strategy can find the best objective function value when D = 100. In other words, the threshold linear prior decreasing strategy (Threshold_2) can find the best objective function value for problem 8 with D = 30 (0.0000). However, the exponential increasing strategy with mr = 0.01 (Exponential_3) can find the best objective function value for problem 8 with D = 100 (2.7729×10^{-2}).

In Table 3, the best results are presented by the boldface type. For example, the Threshold_2 strategy had the best minimum objective function value for problems 1 with D = 30 (7.1381×10^{-39}). The Exponential_6 strategy had the best maximum objective function value (3.7601×10^{-30}) and had the minimum standard deviation value (7.0604×10^{-31}) for problems 1 with D = 30.

In Table 4, among all problems for D = 30, the DANGHS algorithm can find the best objective function value for problems 1–3, 6–10, and 14. The SGHS algorithm can find the best objective function value for problems 2, 4, and 11. The NGHS algorithm can find the best objective function value for problems 2, 12, and 13. The IHS algorithm can find the best objective function value for problem 5. The best algorithms and the best results are presented by the boldface type in Table 4.

On the other hand, among all problems for D = 100, the DANGHS algorithm can find the best objective function value for problems 1–14. The NGHS algorithm can find the best objective function value for problem 2.

Figure 7 presents a typical solution history graph of five different algorithms along iterations for problems 1 to 8 with D = 30, and Figure 8 presents a typical solution history graph of five different algorithms along iterations for problems 9 to 14 with D = 30. Figure 9 presents a typical solution history graph of five different algorithms along iterations for problems 1 to 8 with D = 100, and Figure 10 presents a typical solution history graph of five different algorithms along iterations for problems 9 to 14 with D = 100.

Finally, we will discuss and analyze the efficiency of the DANGHS algorithm. In Figures 7–10, we can easily find out that the DANGHS algorithm obviously had the better searching performance and convergence ability than other algorithms in most low-dimensional problems and in all high-dimensional problems. In other words, the DANGHS algorithm can use the less iterations to solve the problem and is more efficient than other HS algorithms. Besides, according to the experimental results, the DANGHS with Pseudocode 3 spent 603.5025 seconds to run 30 experiments; while the DANGHS with Pseudocode 4 spent 532.7705 seconds only to run 30 experiments. The DANGHS algorithm with Pseudocode 4 reduces 11.72% of the running time, as compared with Pseudocode 3. Therefore, the DANGHS algorithm with the proposed Pseudocode 4 is more efficient than with Pseudocode 3.

Table 3. Experimental results of 16 strategies in the DANGHS algorithms.

No.	Dimension (D) = 30					Dimension (D) = 100				
	Adjustment strategy	**Min**	**Max**	**Mean**	**SD**	**Adjustment strategy**	**Min**	**Max**	**Mean**	**SD**
f_1	Straight_1	1.0461×10^{-17}	4.3759×10^{-15}	6.7177×10^{-16}	1.1026×10^{-15}	Straight_1	7.4075×10^{-6}	3.1563×10^{-4}	3.3196×10^{-5}	5.3403×10^{-5}
	Straight_2	3.1999×10^{-23}	2.5783×10^{-18}	3.6341×10^{-19}	6.7121×10^{-19}	Straight_2	1.0715×10^{-7}	7.1045×10^{-6}	8.4864×10^{-7}	1.2362×10^{-6}
	Threshold_1	1.5431×10^{-13}	7.3069×10^{-11}	6.6315×10^{-12}	1.3094×10^{-11}	Threshold_1	1.0024×10^{-3}	9.1452×10^{-3}	2.7025×10^{-3}	1.6455×10^{-3}
	Threshold_2	**7.1381×10^{-39}**	2.0446×10^{-26}	6.8264×10^{-28}	3.6700×10^{-27}	Threshold_2	3.9739×10^{-16}	8.5814×10^{-14}	1.7158×10^{-14}	2.2275×10^{-14}
	Threshold_3	1.5233×10^{-32}	1.4639×10^{-22}	4.9489×10^{-24}	2.6266×10^{-23}	Threshold_3	5.5110×10^{-15}	4.3556×10^{-12}	3.2639×10^{-13}	7.8921×10^{-13}
	Threshold_4	3.3374×10^{-18}	4.3038×10^{-12}	1.4919×10^{-13}	7.7155×10^{-13}	Threshold_4	3.2962×10^{-5}	1.8976×10^{-4}	8.5947×10^{-5}	4.0524×10^{-5}
	Exponential_1	1.5745×10^{-24}	1.0309×10^{-18}	4.3948×10^{-20}	1.8663×10^{-19}	Exponential_1	2.6917×10^{-9}	4.3964×10^{-8}	1.3508×10^{-8}	9.5937×10^{-9}
	Exponential_2	1.9177×10^{-30}	9.0711×10^{-23}	4.0772×10^{-24}	1.6377×10^{-23}	Exponential_2	3.3307×10^{-11}	$2.2886\text{E} \times 10^{-9}$	4.9657×10^{-10}	5.7327×10^{-10}
	Exponential_3	1.4165×10^{-30}	2.0068×10^{-23}	9.8918×10^{-25}	3.6255×10^{-24}	Exponential_3	4.3579×10^{-12}	9.8204×10^{-11}	3.2054×10^{-11}	2.5181×10^{-11}
	Exponential_4	6.4644×10^{-35}	4.5295×10^{-25}	1.7018×10^{-26}	8.1269×10^{-26}	Exponential_4	9.8540×10^{-14}	7.9844×10^{-12}	1.7156×10^{-12}	1.8932×10^{-12}
	Exponential_5	2.5044×10^{-34}	1.9318×10^{-25}	6.4846×10^{-27}	3.4669×10^{-26}	Exponential_5	1.0612×10^{-14}	6.4803×10^{-12}	3.7018×10^{-13}	1.1689×10^{-12}
	Exponential_6	2.3735×10^{-38}	**3.7601×10^{-30}**	**1.8344×10^{-31}**	**7.0604×10^{-31}**	**Exponential_6**	**1.6615×10^{-16}**	**8.0332×10^{-14}**	**1.2209×10^{-14}**	**1.9706×10^{-14}**
	Cosine_1	1.0929×10^{-25}	2.6839×10^{-19}	1.2194×10^{-20}	4.8268×10^{-20}	Cosine_1	1.3660×10^{-9}	8.8086×10^{-8}	1.7253×10^{-8}	2.0154×10^{-8}
	Cosine_2	4.4254×10^{-21}	3.5110×10^{-16}	2.1719×10^{-17}	7.1432×10^{-17}	Cosine_2	5.6587×10^{-8}	2.9214×10^{-6}	5.8827×10^{-7}	7.8914×10^{-7}
	Cosine_3	1.2834×10^{-22}	1.3530×10^{-17}	7.1008×10^{-19}	2.4622×10^{-18}	Cosine_3	1.9881×10^{-9}	2.8788×10^{-7}	5.6503×10^{-8}	6.6569×10^{-8}
	Cosine_4	2.2000×10^{-22}	6.3880×10^{-16}	3.4309×10^{-17}	1.1748×10^{-16}	Cosine_4	1.7535×10^{-8}	2.7122×10^{-6}	2.0647×10^{-7}	4.7558×10^{-7}
f_2	**Straight_1**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Straight_1**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Straight_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Straight_2**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Threshold_1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Threshold_1**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Threshold_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Threshold_2**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Threshold_3	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Threshold_3**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Threshold_4	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Threshold_4**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Exponential_1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Exponential_1**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Exponential_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Exponential_2**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Exponential_3	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Exponential_3**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Exponential_4	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Exponential_4**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Exponential_5	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Exponential_5**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Exponential_6	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Exponential_6**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Cosine_1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Cosine_1**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Cosine_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Cosine_2**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Cosine_3	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Cosine_3**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Cosine_4	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**Cosine_4**	**0.0000**	**0.0000**	**0.0000**	**0.0000**

Table 3. *Cont.*

No.	Dimension (D) = 30					Dimension (D) = 100				
	Adjustment strategy	**Min**	**Max**	**Mean**	**SD**	**Adjustment strategy**	**Min**	**Max**	**Mean**	**SD**
f_3	Straight_1	7.6101×10^{-11}	2.1913×10^{-8}	1.4051×10^{-9}	3.9001×10^{-9}	Straight_1	8.8188×10^{-4}	2.8473×10^{-3}	1.4467×10^{-3}	4.5882×10^{-4}
	Straight_2	6.3744×10^{-14}	8.2891×10^{-10}	5.6932×10^{-11}	1.5070×10^{-10}	Straight_2	1.0465×10^{-4}	3.6867×10^{-4}	2.0451×10^{-4}	7.2735×10^{-5}
	Threshold_1	4.6938×10^{-8}	1.3461×10^{-6}	2.1964×10^{-7}	2.5585×10^{-7}	Threshold_1	1.4060×10^{-2}	3.5936×10^{-2}	1.9850×10^{-2}	4.3707×10^{-3}
	Threshold_2	5.6623×10^{-23}	8.2711×10^{-17}	2.9440×10^{-18}	1.4815×10^{-17}	Threshold_2	2.5513×10^{-9}	4.2922×10^{-8}	1.0152×10^{-8}	8.0648×10^{-9}
	Threshold_3	4.7815×10^{-20}	5.0811×10^{-11}	1.7085×10^{-12}	9.1183×10^{-12}	Threshold_3	7.0365×10^{-9}	1.3972×10^{-7}	4.1610×10^{-8}	3.3889×10^{-8}
	Threshold_4	1.7455×10^{-10}	1.3321×10^{-7}	1.7945×10^{-8}	3.2820×10^{-8}	Threshold_4	1.3864×10^{-3}	6.5341×10^{-3}	3.0170×10^{-3}	1.0805×10^{-3}
	Exponential_1	9.8893×10^{-15}	8.8305×10^{-11}	6.3308×10^{-12}	1.8435×10^{-11}	Exponential_1	8.0177×10^{-6}	7.4683×10^{-5}	2.0822×10^{-5}	1.1771×10^{-5}
	Exponential_2	1.7782×10^{-17}	6.9813×10^{-12}	3.7142×10^{-13}	1.4030×10^{-12}	Exponential_2	7.3854×10^{-7}	6.7228×10^{-6}	3.3092×10^{-6}	1.5847×10^{-6}
	Exponential_3	7.5286×10^{-18}	4.4195×10^{-13}	2.0483×10^{-14}	7.9637×10^{-14}	Exponential_3	2.6791×10^{-7}	1.9304×10^{-6}	6.8633×10^{-7}	3.3018×10^{-7}
	Exponential_4	2.5827×10^{-20}	1.1413×10^{-14}	4.3303×10^{-16}	2.0473×10^{-15}	Exponential_4	3.1645×10^{-8}	1.2139×10^{-6}	2.0931×10^{-7}	2.3506×10^{-7}
	Exponential_5	1.5957×10^{-20}	3.0055×10^{-15}	1.1154×10^{-16}	5.3828×10^{-16}	Exponential_5	9.3641×10^{-9}	1.3374×10^{-7}	3.2879×10^{-8}	2.5806×10^{-8}
	Exponential_6	$\mathbf{5.1270 \times 10^{-23}}$	$\mathbf{2.5548 \times 10^{-17}}$	$\mathbf{1.9511 \times 10^{-18}}$	$\mathbf{5.5869 \times 10^{-18}}$	Exponential_6	$\mathbf{1.7326 \times 10^{-9}}$	$\mathbf{2.2346 \times 10^{-8}}$	$\mathbf{7.9778 \times 10^{-9}}$	$\mathbf{5.9706 \times 10^{-9}}$
	Cosine_1	8.2615×10^{-15}	4.1648×10^{-10}	1.9989×10^{-11}	7.5562×10^{-11}	Cosine_1	7.5681×10^{-6}	9.0058×10^{-5}	2.7007×10^{-5}	1.7485×10^{-5}
	Cosine_2	2.2087×10^{-12}	1.3549×10^{-9}	1.5788×10^{-10}	2.9202×10^{-10}	Cosine_2	5.0087×10^{-5}	2.4876×10^{-4}	1.1495×10^{-4}	4.8700×10^{-5}
	Cosine_3	8.7106×10^{-14}	4.3784×10^{-11}	6.6558×10^{-12}	1.0409×10^{-11}	Cosine_3	1.1137×10^{-5}	1.2211×10^{-4}	4.6150×10^{-5}	2.7857×10^{-5}
	Cosine_4	1.7511×10^{-13}	4.2184×10^{-10}	3.9035×10^{-11}	8.5768×10^{-11}	Cosine_4	1.6561×10^{-5}	4.1825×10^{-4}	8.5160×10^{-5}	7.5082×10^{-5}
f_4	Straight_1	3.8707×10^{1}	2.0746×10^{2}	9.2469×10^{1}	4.2467×10^{1}	Straight_1	7.4364×10^{3}	1.5798×10^{4}	1.2838×10^{4}	2.0788×10^{3}
	Straight_2	2.9107×10^{1}	2.9786×10^{2}	7.7546×10^{1}	5.2474×10^{1}	Straight_2	6.1981×10^{3}	$\mathbf{1.2256 \times 10^{4}}$	9.9557×10^{3}	$\mathbf{1.5833 \times 10^{3}}$
	Threshold_1	2.5223×10^{1}	2.4305×10^{2}	6.8420×10^{1}	4.2239×10^{1}	Threshold_1	1.2769×10^{4}	2.1277×10^{4}	1.6990×10^{4}	2.1766×10^{3}
	Threshold_2	8.2738×10^{1}	4.8897×10^{2}	2.4674×10^{2}	1.0081×10^{2}	Threshold_2	7.0477×10^{3}	1.6115×10^{4}	1.0330×10^{4}	1.9585×10^{3}
	Threshold_3	1.6962×10^{2}	7.4402×10^{2}	3.4890×10^{2}	1.5861×10^{2}	Threshold_3	7.9459×10^{3}	2.0032×10^{4}	1.3965×10^{4}	2.6464×10^{3}
	Threshold_4	$\mathbf{1.5038 \times 10^{1}}$	$\mathbf{1.5980 \times 10^{2}}$	$\mathbf{6.0249 \times 10^{1}}$	$\mathbf{3.5686 \times 10^{1}}$	Threshold_4	8.9389×10^{3}	1.9386×10^{4}	1.3070×10^{4}	2.9327×10^{3}
	Exponential_1	5.2571×10^{1}	3.3140×10^{2}	1.7427×10^{2}	7.7406×10^{1}	Exponential_1	7.8519×10^{3}	1.5283×10^{4}	1.1462×10^{4}	2.1096×10^{3}
	Exponential_2	3.7816×10^{1}	2.9649×10^{2}	1.4459×10^{2}	6.2181×10^{1}	**Exponential_2**	$\mathbf{4.6763 \times 10^{3}}$	1.3135×10^{4}	$\mathbf{8.6301 \times 10^{3}}$	1.9698×10^{3}
	Exponential_3	9.1368×10^{1}	7.9952×10^{2}	3.4719×10^{2}	1.6819×10^{2}	Exponential_3	8.0589×10^{3}	1.7800×10^{4}	1.1645×10^{4}	2.4428×10^{3}
	Exponential_4	5.2605×10^{1}	6.6496×10^{2}	2.5585×10^{2}	1.3827×10^{2}	Exponential_4	6.8390×10^{3}	1.4895×10^{4}	9.8522×10^{3}	1.6440×10^{3}
	Exponential_5	1.9773×10^{2}	1.0629×10^{3}	5.5626×10^{2}	2.1853×10^{2}	Exponential_5	7.2667×10^{3}	1.7819×10^{4}	1.2364×10^{4}	2.4484×10^{3}
	Exponential_6	1.1519×10^{2}	1.1733×10^{3}	5.4639×10^{2}	2.6598×10^{2}	Exponential_6	7.0572×10^{3}	1.4985×10^{4}	1.0313×10^{4}	1.8035×10^{3}
	Cosine_1	2.2677×10^{1}	1.6215×10^{2}	8.4233×10^{1}	3.8116×10^{1}	Cosine_1	8.5285×10^{3}	1.6216×10^{4}	1.1657×10^{4}	2.0568×10^{3}
	Cosine_2	3.5648×10^{1}	2.5791×10^{2}	1.0386×10^{2}	4.7531×10^{1}	Cosine_2	7.3675×10^{3}	1.6256×10^{4}	1.1952×10^{4}	2.1374×10^{3}
	Cosine_3	3.9845×10^{1}	2.5401×10^{2}	1.0903×10^{2}	5.1737×10^{1}	Cosine_3	8.7901×10^{3}	1.5058×10^{4}	1.2176×10^{4}	1.5990×10^{3}
	Cosine_4	4.0827×10^{1}	1.9696×10^{2}	9.1923×10^{1}	4.1878×10^{1}	Cosine_4	8.2399×10^{3}	1.4967×10^{4}	1.1576×10^{4}	1.6474×10^{3}

Table 3. *Cont.*

No.	Dimension (D) = 30					Dimension (D) = 100				
	Adjustment strategy	**Min**	**Max**	**Mean**	**SD**	**Adjustment strategy**	**Min**	**Max**	**Mean**	**SD**
f_5	Straight_1	1.2321×10^{-2}	2.7805×10^{-1}	1.0051×10^{-1}	6.7510×10^{-2}	Straight_1	4.1360×10^{-6}	3.5617×10^{-1}	1.0453×10^{-1}	9.3265×10^{-2}
	Straight_2	**0.0000**	2.0030×10^{-1}	4.1983×10^{-2}	4.2581×10^{-2}	Straight_2	5.1016×10^{-8}	6.3390×10^{-2}	1.1625×10^{-2}	1.5055×10^{-2}
	Threshold_1	1.7855×10^{-8}	2.6377×10^{-1}	9.8336×10^{-2}	6.8880×10^{-2}	Threshold_1	8.8364×10^{-4}	2.8092×10^{-1}	7.3770×10^{-2}	6.8845×10^{-2}
	Threshold_2	**0.0000**	1.8867×10^{-1}	4.0923×10^{-2}	4.6115×10^{-2}	Threshold_2	1.4433×10^{-15}	9.4723×10^{-2}	1.7078×10^{-2}	2.3582×10^{-2}
	Threshold_3	3.6320×10^{-6}	2.2197×10^{-1}	1.1814×10^{-1}	6.0107×10^{-2}	Threshold_3	1.3545×10^{-14}	3.6928×10^{-1}	1.0457×10^{-1}	9.8452×10^{-2}
	Threshold_4	6.6613×10^{-16}	$\mathbf{8.0817 \times 10^{-2}}$	$\mathbf{3.1209 \times 10^{-2}}$	$\mathbf{2.3482 \times 10^{-2}}$	Threshold_4	1.7774×10^{-5}	$\mathbf{3.6827 \times 10^{-2}}$	9.0876×10^{-3}	$\mathbf{9.4618 \times 10^{-3}}$
	Exponential_1	**0.0000**	2.4379×10^{-1}	7.9307×10^{-2}	5.8015×10^{-2}	Exponential_1	2.4888×10^{-9}	4.4986×10^{-1}	1.2518×10^{-1}	9.4176×10^{-2}
	Exponential_2	**0.0000**	1.7609×10^{-1}	4.5556×10^{-2}	4.1760×10^{-2}	**Exponential_2**	3.5352×10^{-11}	4.8906×10^{-2}	$\mathbf{6.4754 \times 10^{-3}}$	9.8313×10^{-3}
	Exponential_3	4.2099×10^{-10}	3.2955×10^{-1}	1.1643×10^{-1}	8.0421×10^{-2}	Exponential_3	3.6280×10^{-12}	3.2945×10^{-1}	1.0033×10^{-1}	9.5541×10^{-2}
	Exponential_4	**0.0000**	2.0253×10^{-1}	4.8723×10^{-2}	4.6463×10^{-2}	Exponential_4	8.5154×10^{-14}	1.6488×10^{-1}	1.2543×10^{-2}	3.0277×10^{-2}
	Exponential_5	8.8818×10^{-16}	2.5708×10^{-1}	8.4472×10^{-2}	6.4674×10^{-2}	Exponential_5	1.2990×10^{-14}	4.9613×10^{-1}	1.2331×10^{-1}	1.0636×10^{-1}
	Exponential_6	**0.0000**	1.6595×10^{-1}	4.9137×10^{-2}	3.9865×10^{-2}	Exponential_6	$\mathbf{2.2204 \times 10^{-16}}$	8.3124×10^{-2}	1.2513×10^{-2}	1.8702×10^{-2}
	Cosine_1	**0.0000**	1.4149×10^{-1}	4.2637×10^{-2}	4.3674×10^{-2}	Cosine_1	8.3748×10^{-10}	5.4050×10^{-2}	1.0821×10^{-2}	1.5102×10^{-2}
	Cosine_2	1.6653×10^{-15}	3.3647×10^{-1}	1.1985×10^{-1}	8.0014×10^{-2}	Cosine_2	6.3283×10^{-8}	4.0323×10^{-1}	9.7344×10^{-2}	9.0742×10^{-2}
	Cosine_3	**0.0000**	1.3191×10^{-1}	5.2162×10^{-2}	3.7768×10^{-2}	Cosine_3	2.6663×10^{-9}	1.7983×10^{-1}	3.1360×10^{-2}	4.5413×10^{-2}
	Cosine_4	1.1102×10^{-16}	2.7598×10^{-1}	9.9773×10^{-2}	6.7164×10^{-2}	Cosine_4	1.6227×10^{-8}	2.4487×10^{-1}	5.5685×10^{-2}	6.5963×10^{-2}
f_6	Straight_1	4.4632×10^{-9}	2.1372×10^{-7}	3.9100×10^{-8}	4.0940×10^{-8}	Straight_1	9.1269×10^{-4}	3.1804×10^{-3}	1.6518×10^{-3}	5.3523×10^{-4}
	Straight_2	3.7761×10^{-12}	8.6966×10^{-10}	1.0236×10^{-10}	2.0891×10^{-10}	Straight_2	4.2112×10^{-5}	3.8837×10^{-4}	1.1886×10^{-4}	7.9706×10^{-5}
	Threshold_1	9.1029×10^{-7}	8.0166×10^{-6}	2.4129×10^{-6}	1.5101×10^{-6}	Threshold_1	1.2113×10^{-2}	2.5585×10^{-2}	1.9902×10^{-2}	3.4669×10^{-3}
	Threshold_2	7.4163×10^{-14}	6.3549×10^{-13}	1.6938×10^{-13}	1.0610×10^{-13}	Threshold_2	1.7921×10^{-9}	6.5055×10^{-8}	1.2631×10^{-8}	1.2902×10^{-8}
	Threshold_3	1.0014×10^{-12}	5.2655×10^{-10}	6.8025×10^{-11}	1.2657×10^{-10}	Threshold_3	2.5582×10^{-8}	1.5023×10^{-6}	2.2886×10^{-7}	2.7137×10^{-7}
	Threshold_4	1.4622×10^{-9}	2.7494×10^{-7}	3.1953×10^{-8}	4.9739×10^{-8}	Threshold_4	5.3312×10^{-4}	3.8584×10^{-3}	1.5894×10^{-3}	7.9124×10^{-4}
	Exponential_1	1.8744×10^{-11}	4.9544×10^{-9}	4.4463×10^{-10}	8.7649×10^{-10}	Exponential_1	1.7624×10^{-5}	9.1723×10^{-5}	4.3472×10^{-5}	1.7983×10^{-5}
	Exponential_2	1.1324×10^{-13}	4.9805×10^{-12}	9.3889×10^{-13}	1.1256×10^{-12}	Exponential_2	6.4716×10^{-7}	6.8428×10^{-6}	2.1846×10^{-6}	1.3670×10^{-6}
	Exponential_3	4.9338×10^{-13}	5.5745×10^{-11}	6.6129×10^{-12}	1.1049×10^{-11}	Exponential_3	6.0042×10^{-7}	6.0591×10^{-6}	2.4126×10^{-6}	1.5002×10^{-6}
	Exponential_4	7.4163×10^{-14}	1.1862×10^{-12}	1.7826×10^{-13}	1.9564×10^{-13}	Exponential_4	3.8729×10^{-8}	3.1068×10^{-7}	1.2811×10^{-7}	6.7689×10^{-8}
	Exponential_5	1.5588×10^{-13}	4.0887×10^{-12}	8.2758×10^{-13}	8.0931×10^{-13}	Exponential_5	3.9803×10^{-8}	5.7158×10^{-7}	1.4832×10^{-7}	1.2402×10^{-7}
	Exponential_6	$\mathbf{4.9294 \times 10^{-14}}$	$\mathbf{2.2338 \times 10^{-13}}$	$\mathbf{9.6308 \times 10^{-14}}$	$\mathbf{3.4392 \times 10^{-14}}$	**Exponential_6**	$\mathbf{1.0897 \times 10^{-9}}$	$\mathbf{2.9882 \times 10^{-8}}$	$\mathbf{9.3030 \times 10^{-9}}$	$\mathbf{7.9441 \times 10^{-9}}$
	Cosine_1	1.1506×10^{-12}	9.2267×10^{-11}	2.2030×10^{-11}	2.6803×10^{-11}	Cosine_1	2.3571×10^{-6}	1.0218×10^{-4}	1.8274×10^{-5}	1.7509×10^{-5}
	Cosine_2	1.7620×10^{-10}	5.8307×10^{-8}	6.5754×10^{-9}	1.2324×10^{-8}	Cosine_2	6.1209×10^{-5}	5.0676×10^{-4}	1.8226×10^{-4}	1.1078×10^{-4}
	Cosine_3	5.0302×10^{-12}	7.4329×10^{-10}	1.1653×10^{-10}	1.6314×10^{-10}	Cosine_3	8.4419×10^{-6}	8.5649×10^{-5}	2.7943×10^{-5}	1.8587×10^{-5}
	Cosine_4	7.7018×10^{-12}	5.7757×10^{-9}	5.7647×10^{-10}	1.2234×10^{-9}	Cosine_4	2.9068×10^{-5}	1.9156×10^{-4}	6.7917×10^{-5}	4.2305×10^{-5}

Table 3. *Cont.*

No.	Dimension (D) = 30					Dimension (D) = 100				
	Adjustment strategy	**Min**	**Max**	**Mean**	**SD**	**Adjustment strategy**	**Min**	**Max**	**Mean**	**SD**
f_7	Straight_1	9.3515×10^{-3}	$\mathbf{2.2089 \times 10^{1}}$	$\mathbf{1.0089 \times 10^{1}}$	**8.7452**	Straight_1	1.1325	5.6386×10^{2}	1.0620×10^{2}	1.2570×10^{2}
	Straight_2	8.6741×10^{-3}	5.1406×10^{2}	4.4419×10^{1}	9.3454×10^{1}	Straight_2	1.0018×10^{2}	6.9565×10^{2}	2.6307×10^{2}	1.3802×10^{2}
	Threshold_1	7.9026×10^{-2}	4.7147×10^{2}	2.5678×10^{1}	8.3161×10^{1}	Threshold_1	3.4344×10^{1}	2.3308×10^{3}	3.2695×10^{2}	5.9498×10^{2}
	Threshold_2	1.4559×10^{-3}	6.1627×10^{2}	8.4192×10^{1}	1.6666×10^{2}	Threshold_2	1.1503×10^{2}	5.1947×10^{2}	2.3270×10^{2}	9.2809×10^{1}
	Threshold_3	4.4533×10^{-2}	3.9917×10^{2}	3.9657×10^{1}	9.6881×10^{1}	**Threshold_3**	5.6804×10^{-2}	1.1562×10^{3}	$\mathbf{6.1559 \times 10^{1}}$	2.0554×10^{2}
	Threshold_4	1.5343×10^{-3}	1.6611×10^{2}	3.2721×10^{1}	4.3985×10^{1}	Threshold_4	1.8240×10^{2}	7.1746×10^{2}	2.8865×10^{2}	1.1583×10^{2}
	Exponential_1	8.0795×10^{-4}	2.2094×10^{1}	1.2282×10^{1}	8.8145	Exponential_1	2.3784×10^{-1}	1.2901×10^{3}	1.0700×10^{2}	2.5169×10^{2}
	Exponential_2	1.2988×10^{-2}	2.2802×10^{2}	4.6231×10^{1}	5.4244×10^{1}	Exponential_2	1.1174×10^{2}	6.4584×10^{2}	2.2362×10^{2}	9.2963×10^{1}
	Exponential_3	5.6146×10^{-3}	9.7058×10^{2}	5.8068×10^{1}	1.8542×10^{2}	Exponential_3	4.0212×10^{-2}	1.0768×10^{3}	1.0022×10^{2}	2.1852×10^{2}
	Exponential_4	5.5288×10^{-3}	9.3064×10^{1}	2.6240×10^{1}	3.2270×10^{1}	Exponential_4	7.6558×10^{1}	$\mathbf{2.8626 \times 10^{2}}$	2.0961×10^{2}	$\mathbf{5.1587 \times 10^{1}}$
	Exponential_5	2.7747×10^{-3}	1.6422×10^{3}	1.3893×10^{2}	3.7163×10^{2}	Exponential_5	$\mathbf{1.2275 \times 10^{-3}}$	1.5623×10^{3}	9.0305×10^{1}	2.8689×10^{2}
	Exponential_6	$\mathbf{1.2831 \times 10^{-5}}$	4.9205×10^{2}	4.0422×10^{1}	9.0413×10^{1}	Exponential_6	5.8594×10^{1}	6.3056×10^{2}	2.1990×10^{2}	9.4453×10^{1}
	Cosine_1	1.8015×10^{-3}	1.3748×10^{2}	2.9047×10^{1}	3.8603×10^{1}	Cosine_1	1.0997×10^{2}	1.2606×10^{3}	2.6362×10^{2}	2.0716×10^{2}
	Cosine_2	4.4398×10^{-3}	5.3878×10^{2}	3.6644×10^{1}	1.0520×10^{2}	Cosine_2	8.8583×10^{-1}	1.6674×10^{3}	1.5580×10^{2}	3.2284×10^{2}
	Cosine_3	1.1943×10^{-1}	3.6476×10^{2}	3.5174×10^{1}	6.8477×10^{1}	Cosine_3	1.4076×10^{2}	1.7107×10^{3}	3.5751×10^{2}	3.8108×10^{2}
	Cosine_4	8.0777×10^{-3}	8.0590×10^{1}	1.4814×10^{1}	1.8135×10^{1}	Cosine_4	1.0177	1.7477×10^{3}	1.9765×10^{2}	3.2461×10^{2}
f_8	Straight_1	3.1974×10^{-14}	3.3089×10^{-7}	1.1716×10^{-8}	5.9285×10^{-8}	Straight_1	2.0388×10^{-3}	3.3216	6.6242×10^{-1}	8.8988×10^{-1}
	Straight_2	**0.0000**	3.5527×10^{-15}	5.9212×10^{-16}	1.1543×10^{-15}	Straight_2	1.9115×10^{-7}	2.9849	3.9804×10^{-1}	7.9594×10^{-1}
	Threshold_1	3.6512×10^{-10}	2.1702×10^{-7}	4.1154×10^{-8}	6.3766×10^{-8}	Threshold_1	2.2994	8.4160	5.4959	1.4385
	Threshold_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	Threshold_2	1.4211×10^{-14}	1.9899	1.9909×10^{-1}	4.7366×10^{-1}
	Threshold_3	**0.0000**	9.9496×10^{-1}	6.6407×10^{-2}	2.4817×10^{-1}	Threshold_3	3.2097×10^{-10}	1.9918	4.5307×10^{-1}	6.0607×10^{-1}
	Threshold_4	**0.0000**	8.8285×10^{-13}	1.0646×10^{-13}	2.0045×10^{-13}	Threshold_4	9.9507×10^{-1}	7.9637	3.2612	1.7811
	Exponential_1	5.3291×10^{-15}	7.1937×10^{-8}	9.0557×10^{-9}	2.0970×10^{-8}	Exponential_1	6.3110×10^{-6}	9.9714×10^{-1}	1.4900×10^{-1}	3.3483×10^{-1}
	Exponential_2	**0.0000**	1.7764×10^{-15}	1.7764×10^{-16}	5.3291×10^{-16}	Exponential_2	1.1186×10^{-10}	9.9496×10^{-1}	9.9549×10^{-2}	2.9847×10^{-1}
	Exponential_3	**0.0000**	6.4209×10^{-6}	3.3419×10^{-7}	1.2242×10^{-6}	**Exponential_3**	3.5170×10^{-8}	$\mathbf{5.9362 \times 10^{-1}}$	$\mathbf{2.7729 \times 10^{-2}}$	$\mathbf{1.0943 \times 10^{-1}}$
	Exponential_4	**0.0000**	1.9899	9.9496×10^{-2}	3.9382×10^{-1}	Exponential_4	1.3145×10^{-13}	9.9496×10^{-1}	9.9497×10^{-2}	2.9849×10^{-1}
	Exponential_5	**0.0000**	9.9496×10^{-1}	9.9534×10^{-2}	2.9848×10^{-1}	Exponential_5	1.9582×10^{-10}	1.0029	1.3296×10^{-1}	3.3892×10^{-1}
	Exponential_6	**0.0000**	2.6645×10^{-14}	1.0066×10^{-15}	4.7815×10^{-15}	Exponential_6	$\mathbf{1.0658 \times 10^{-14}}$	1.9899	6.6332×10^{-2}	3.5720×10^{-1}
	Cosine_1	**0.0000**	4.7731×10^{-10}	1.5911×10^{-11}	8.5680×10^{-11}	Cosine_1	4.0101×10^{-7}	2.9978	6.8374×10^{-1}	8.4558×10^{-1}
	Cosine_2	**0.0000**	8.7041×10^{-14}	1.0836×10^{-14}	1.6641×10^{-14}	Cosine_2	7.9506×10^{-7}	1.9900	9.2880×10^{-1}	7.2337×10^{-1}
	Cosine_3	**0.0000**	4.4409×10^{-13}	2.6053×10^{-14}	8.0975×10^{-14}	Cosine_3	9.6632×10^{-6}	2.0318	6.9709×10^{-1}	6.1734×10^{-1}
	Cosine_4	**0.0000**	7.4181×10^{-9}	3.8545×10^{-10}	1.4870×10^{-9}	Cosine_4	3.2724×10^{-5}	2.9858	7.9728×10^{-1}	8.2825×10^{-1}

Table 3. *Cont.*

No.	Dimension (D) = 30					Dimension (D) = 100				
	Adjustment strategy	**Min**	**Max**	**Mean**	**SD**	**Adjustment strategy**	**Min**	**Max**	**Mean**	**SD**
f_9	Straight_1	$\mathbf{3.8183 \times 10^{-4}}$	3.8184×10^{-4}	3.8184×10^{-4}	2.3807×10^{-9}	Straight_1	8.5055×10^{-3}	2.7732×10^{1}	1.5810	5.5711
	Straight_2	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	1.5953×10^{-13}	Straight_2	1.2731×10^{-3}	1.3293×10^{-3}	1.2773×10^{-3}	1.0150×10^{-5}
	Threshold_1	$\mathbf{3.8183 \times 10^{-4}}$	3.8229×10^{-4}	3.8190×10^{-4}	9.3441×10^{-8}	Threshold_1	6.0293×10^{-1}	1.3343×10^{2}	1.6668×10^{1}	3.6662×10^{1}
	Threshold_2	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{1.3763 \times 10^{-13}}$	**Threshold_2**	$\mathbf{1.2728 \times 10^{-3}}$	$\mathbf{1.2728 \times 10^{-3}}$	$\mathbf{1.2728 \times 10^{-3}}$	$\mathbf{1.4537 \times 10^{-9}}$
	Threshold_3	$\mathbf{3.8183 \times 10^{-4}}$	4.0474×10^{-4}	3.8314×10^{-4}	4.2676×10^{-6}	Threshold_3	$\mathbf{1.2728 \times 10^{-3}}$	1.1844×10^{2}	4.6076	2.1276×10^{1}
	Threshold_4	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	2.0629×10^{-12}	Threshold_4	1.3933×10^{-3}	1.1844×10^{2}	7.9095	2.9541×10^{1}
	Exponential_1	$\mathbf{3.8183 \times 10^{-4}}$	1.5644×10^{-4}	4.2241×10^{-4}	2.1212×10^{-4}	Exponential_1	1.2781×10^{-3}	1.7045×10^{-2}	2.7552×10^{-3}	3.2908×10^{-3}
	Exponential_2	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	1.4555×10^{-13}	Exponential_2	$\mathbf{1.2728 \times 10^{-3}}$	1.2728×10^{-3}	1.2728×10^{-3}	4.9514×10^{-9}
	Exponential_3	$\mathbf{3.8183 \times 10^{-4}}$	3.9497×10^{-4}	3.8261×10^{-4}	2.8259×10^{-6}	Exponential_3	$\mathbf{1.2728 \times 10^{-3}}$	4.6522×10^{-3}	1.4855×10^{-3}	6.7891×10^{-4}
	Exponential_4	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	1.6171×10^{-13}	Exponential_4	$\mathbf{1.2728 \times 10^{-3}}$	1.3827×10^{-3}	1.2764×10^{-3}	1.9734×10^{-5}
	Exponential_5	$\mathbf{3.8183 \times 10^{-4}}$	4.2990×10^{-4}	3.8366×10^{-4}	8.6277×10^{-6}	Exponential_5	$\mathbf{1.2728 \times 10^{-3}}$	1.1844×10^{2}	6.9547	2.6270×10^{1}
	Exponential_6	$\mathbf{3.8183 \times 10^{-4}}$	1.1844×10^{2}	3.9483	2.1260×10^{1}	Exponential_6	$\mathbf{1.2728 \times 10^{-3}}$	1.2730×10^{-3}	$\mathbf{1.2728 \times 10^{-3}}$	4.1256×10^{-8}
	Cosine_1	$\mathbf{3.8183 \times 10^{-4}}$	3.8231×10^{-4}	3.8184×10^{-4}	8.6835×10^{-8}	Cosine_1	$\mathbf{1.2728 \times 10^{-3}}$	1.1844×10^{2}	4.1934	2.1251×10^{1}
	Cosine_2	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	5.5438×10^{-13}	Cosine_2	1.2765×10^{-3}	1.4798×10^{-3}	1.3277×10^{-3}	5.8337×10^{-5}
	Cosine_3	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	8.8412×10^{-13}	Cosine_3	1.2731×10^{-3}	1.5879×10^{-3}	1.3274×10^{-3}	8.8845×10^{-5}
	Cosine_4	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	$\mathbf{3.8183 \times 10^{-4}}$	3.6890×10^{-13}	Cosine_4	1.2732×10^{-3}	4.8779×10^{-2}	3.7520×10^{-3}	8.8272×10^{-3}
f_{10}	Straight_1	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.0008×10^{-13}	Straight_1	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.1952×10^{-5}
	Straight_2	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	7.3385×10^{-14}	Straight_2	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.4478×10^{-6}
	Threshold_1	$\mathbf{-4.5000 \times 10^{2}}$	-4.4999×10^{2}	-4.4999×10^{2}	8.3459×10^{-12}	Threshold_1	$\mathbf{-4.5000 \times 10^{2}}$	-4.4999×10^{2}	-4.4999×10^{2}	1.7050×10^{-3}
	Threshold_2	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	6.3128×10^{-14}	**Threshold_2**	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{2.4117 \times 10^{-13}}$
	Threshold_3	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	9.7356×10^{-14}	Threshold_3	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	4.6932×10^{-13}
	Threshold_4	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.9443×10^{-13}	Threshold_4	$\mathbf{-4.5000 \times 10^{2}}$	-4.4999×10^{2}	-4.4999×10^{2}	9.0431×10^{-5}
	Exponential_1	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	7.4115×10^{-14}	Exponential_1	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.1789×10^{-8}
	Exponential_2	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{5.4916 \times 10^{-14}}$	Exponential_2	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	4.2493×10^{-10}
	Exponential_3	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	7.4838×10^{-14}	Exponential_3	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	2.2269×10^{-10}
	Exponential_4	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	6.8841×10^{-14}	Exponential_4	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.5991×10^{-12}
	Exponential_5	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	7.3385×10^{-14}	Exponential_5	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	5.9672×10^{-13}
	Exponential_6	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	7.1149×10^{-14}	Exponential_6	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	2.8686×10^{-13}
	Cosine_1	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	9.0475×10^{-14}	Cosine_1	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.9650×10^{-8}
	Cosine_2	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	9.0475×10^{-14}	Cosine_2	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.1307×10^{-6}
	Cosine_3	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	8.3025×10^{-14}	Cosine_3	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	2.3531×10^{-7}
	Cosine_4	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	8.5580×10^{-14}	Cosine_4	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	$\mathbf{-4.5000 \times 10^{2}}$	1.5911×10^{-7}

Table 3. *Cont.*

No.	Dimension (D) = 30					Dimension (D) = 100				
	Adjustment strategy	**Min**	**Max**	**Mean**	**SD**	**Adjustment strategy**	**Min**	**Max**	**Mean**	**SD**
f_{11}	Straight_1	-4.2165×10^{2}	-1.3296×10^{2}	-3.0000×10^{2}	7.4536×10^{1}	Straight_1	1.1767×10^{4}	2.3516×10^{4}	1.6891×10^{4}	3.1744×10^{3}
	Straight_2	-4.3444×10^{2}	-1.0722×10^{2}	-3.2701×10^{2}	8.4417×10^{1}	Straight_2	8.4226×10^{3}	1.8619×10^{4}	1.2552×10^{4}	**2.2028×10^{3}**
	Threshold_1	-4.0106×10^{2}	-1.4896×10^{2}	-3.3088×10^{2}	5.8709×10^{1}	Threshold_1	1.5199×10^{4}	2.8379×10^{4}	2.1552×10^{4}	3.2507×10^{3}
	Threshold_2	-3.4901×10^{2}	2.5368×10^{2}	-6.2125×10^{1}	1.5696×10^{2}	Threshold_2	8.7977×10^{3}	1.7870×10^{4}	1.3695×10^{4}	2.2976×10^{3}
	Threshold_3	-3.3390×10^{2}	9.3989×10^{2}	1.2263×10^{2}	3.1892×10^{2}	Threshold_3	1.2026×10^{4}	3.1616×10^{4}	2.1121×10^{4}	4.5467×10^{3}
	Threshold_4	**-4.4289×10^{2}**	**-2.5392×10^{2}**	**-3.7419×10^{2}**	**4.4269×10^{1}**	Threshold_4	9.0435×10^{3}	2.5277×10^{4}	1.6287×10^{4}	3.1871×10^{3}
	Exponential_1	-3.1846×10^{2}	2.1908×10^{2}	-9.9376×10^{1}	1.4315×10^{2}	Exponential_1	1.0395×10^{4}	2.1639×10^{4}	1.5861×10^{4}	3.1550×10^{3}
	Exponential_2	-3.8982×10^{2}	3.1632×10^{2}	-1.6371×10^{2}	1.6523×10^{2}	**Exponential_2**	**6.9718×10^{3}**	**1.7181×10^{4}**	**1.1471×10^{4}**	2.4981×10^{3}
	Exponential_3	-3.0139×10^{2}	2.1120×10^{3}	1.5585×10^{2}	4.5585×10^{2}	Exponential_3	1.2034×10^{4}	2.4924×10^{4}	1.7356×10^{4}	2.6542×10^{3}
	Exponential_4	-3.3706×10^{2}	2.7787×10^{2}	-5.1714×10^{1}	1.6504×10^{2}	Exponential_4	8.1868×10^{3}	1.8387×10^{4}	1.2757×10^{4}	2.9393×10^{3}
	Exponential_5	-2.5225×10^{2}	1.5514×10^{3}	5.6252×10^{2}	4.1866×10^{2}	Exponential_5	1.2930×10^{4}	2.3952×10^{4}	1.8058×10^{4}	2.9540×10^{3}
	Exponential_6	-2.8287×10^{2}	9.9251×10^{2}	2.6238×10^{2}	3.3327×10^{2}	Exponential_6	9.7956×10^{3}	2.0311×10^{4}	1.3953×10^{4}	2.4149×10^{3}
	Cosine_1	-4.1764×10^{2}	-1.8561×10^{1}	-2.6591×10^{2}	9.4610×10^{1}	Cosine_1	9.0379×10^{3}	2.5832×10^{4}	1.5303×10^{4}	3.5867×10^{3}
	Cosine_2	-4.2366×10^{2}	-4.9167×10^{1}	-2.7333×10^{2}	9.0290×10^{1}	Cosine_2	1.2007×10^{4}	2.5142×10^{4}	1.7464×10^{4}	3.3184×10^{3}
	Cosine_3	-4.2193×10^{2}	8.5517×10^{1}	-2.2015×10^{2}	1.2019×10^{2}	Cosine_3	1.0444×10^{4}	2.0965×10^{4}	1.4526×10^{4}	2.5147×10^{3}
	Cosine_4	-4.0935×10^{2}	4.8281×10^{1}	-2.5462×10^{2}	1.2357×10^{2}	Cosine_4	1.0484×10^{4}	2.3679×10^{4}	1.6049×10^{4}	3.3138×10^{3}
f_{12}	Straight_1	-1.7895×10^{2}	-1.7527×10^{2}	-1.7807×10^{2}	9.8257×10^{-1}	Straight_1	-1.5655×10^{2}	-9.8500×10^{1}	-1.3288×10^{2}	1.5417×10^{1}
	Straight_2	-1.7894×10^{2}	-1.7562×10^{2}	**-1.7821×10^{2}**	7.6813×10^{-1}	Straight_2	-1.6750×10^{2}	-1.2621×10^{2}	-1.4904×10^{2}	1.1365×10^{1}
	Threshold_1	-1.7886×10^{2}	-1.7403×10^{2}	-1.7782×10^{2}	1.0687	Threshold_1	-1.3312×10^{2}	-5.8638×10^{1}	-9.9192×10^{1}	2.1166×10^{1}
	Threshold_2	-1.7891×10^{2}	-1.7361×10^{2}	-1.7769×10^{2}	1.0980	Threshold_2	-1.6995×10^{2}	-1.2583×10^{2}	-1.5366×10^{2}	1.0233×10^{1}
	Threshold_3	-1.7883×10^{2}	-1.7150×10^{2}	-1.7711×10^{2}	1.4020	Threshold_3	-1.6505×10^{2}	-1.2822×10^{2}	-1.4608×10^{2}	1.0842×10^{1}
	Threshold_4	-1.7888×10^{2}	-1.7563×10^{2}	-1.7812×10^{2}	7.7099×10^{-1}	Threshold_4	-1.6802×10^{2}	-1.0750×10^{2}	-1.3614×10^{2}	1.5041×10^{1}
	Exponential_1	-1.7881×10^{2}	**-1.7636×10^{2}**	-1.7781×10^{2}	7.2018×10^{-1}	Exponential_1	-1.6142×10^{2}	-1.1356×10^{2}	-1.4376×10^{2}	1.2858×10^{1}
	Exponential_2	-1.7883×10^{2}	-1.7565×10^{2}	-1.7788×10^{2}	9.6280×10^{-1}	Exponential_2	-1.6950×10^{2}	-1.4032×10^{2}	-1.5798×10^{2}	**6.6526**
	Exponential_3	**-1.7909×10^{2}**	-1.7556×10^{2}	-1.7747×10^{2}	9.0208×10^{-1}	Exponential_3	-1.6515×10^{2}	-1.2713×10^{2}	-1.5275×10^{2}	9.8056
	Exponential_4	-1.7890×10^{2}	-1.7505×10^{2}	-1.7780×10^{2}	8.8838×10^{-1}	Exponential_4	-1.7101×10^{2}	-1.4042×10^{2}	-1.5931×10^{2}	7.7148
	Exponential_5	-1.7882×10^{2}	-1.7400×10^{2}	-1.7725×10^{2}	1.2914	Exponential_5	**-1.7364×10^{2}**	-1.2751×10^{2}	-1.5552×10^{2}	1.0491×10^{1}
	Exponential_6	-1.7892×10^{2}	-1.7468×10^{2}	-1.7757×10^{2}	1.0984	**Exponential_6**	-1.7066×10^{2}	**-1.4082×10^{2}**	**-1.6037×10^{2}**	6.8764
	Cosine_1	-1.7889×10^{2}	-1.7626×10^{2}	-1.7804×10^{2}	6.7941×10^{-1}	Cosine_1	-1.7102×10^{2}	-9.7261×10^{1}	-1.4361×10^{2}	1.5506×10^{1}
	Cosine_2	-1.7888×10^{2}	-1.7610×10^{2}	-1.7807×10^{2}	**6.4727×10^{-1}**	Cosine_2	-1.6466×10^{2}	-1.1821×10^{2}	-1.4021×10^{2}	1.3072×10^{1}
	Cosine_3	-1.7891×10^{2}	-1.7571×10^{2}	-1.7791×10^{2}	8.9066×10^{-1}	Cosine_3	-1.6130×10^{2}	-1.1206×10^{2}	-1.4392×10^{2}	1.3861×10^{1}
	Cosine_4	-1.7873×10^{2}	-1.7465×10^{2}	-1.7780×10^{2}	8.3139×10^{-1}	Cosine_4	-1.6327×10^{2}	-1.0337×10^{2}	-1.3840×10^{2}	1.5854×10^{1}

Table 3. *Cont.*

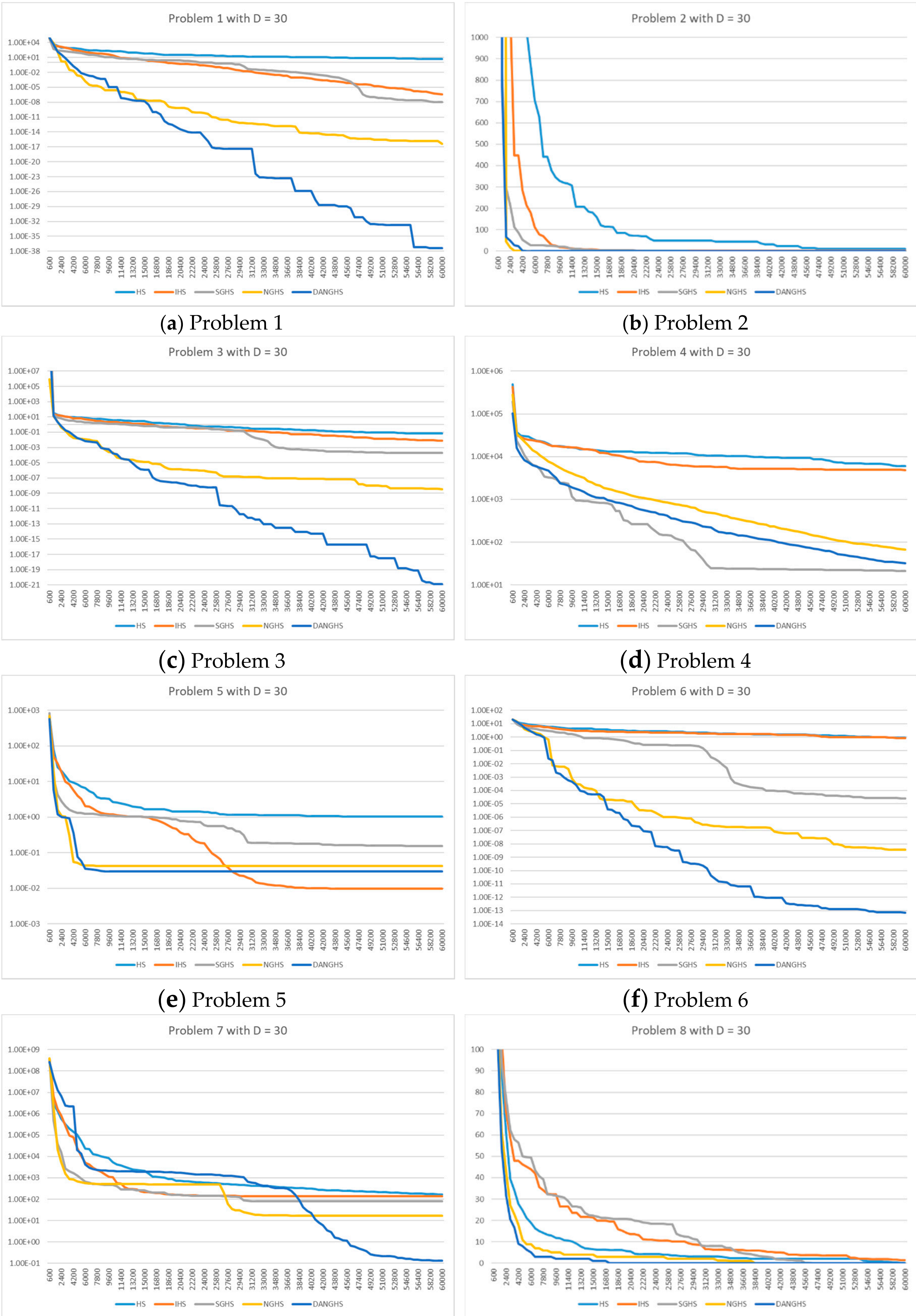
No.	Dimension (D) = 30					Dimension (D) = 100				
	Adjustment strategy	**Min**	**Max**	**Mean**	**SD**	**Adjustment strategy**	**Min**	**Max**	**Mean**	**SD**
f_{13}	Straight_1	**3.9000×10^{2}**	4.5949×10^{2}	3.9958×10^{2}	1.3088×10^{1}	Straight_1	3.9280×10^{2}	3.6620×10^{3}	5.7871×10^{2}	5.8225×10^{2}
	Straight_2	3.9002×10^{2}	4.8656×10^{2}	4.0653×10^{2}	2.5520×10^{1}	Straight_2	5.5729×10^{2}	2.9468×10^{3}	8.3659×10^{2}	5.4759×10^{2}
	Threshold_1	3.9007×10^{2}	4.0876×10^{2}	3.9979×10^{2}	**7.3298**	Threshold_1	4.2763×10^{2}	3.9501×10^{3}	8.1099×10^{2}	8.3480×10^{2}
	Threshold_2	**3.9000×10^{2}**	7.9400×10^{2}	4.2759×10^{2}	7.5937×10^{1}	Threshold_2	4.9760×10^{2}	2.2597×10^{3}	6.9290×10^{2}	3.1747×10^{2}
	Threshold_3	3.9012×10^{2}	6.7882×10^{2}	4.0955×10^{2}	5.0771×10^{1}	Threshold_3	3.9002×10^{2}	2.8846×10^{3}	8.5966×10^{2}	7.8665×10^{2}
	Threshold_4	**3.9000×10^{2}**	4.7366×10^{2}	4.1236×10^{2}	3.1291×10^{1}	Threshold_4	4.8872×10^{2}	3.5299×10^{3}	9.3974×10^{2}	7.2954×10^{2}
	Exponential_1	3.9001×10^{2}	7.8965×10^{2}	4.1481×10^{2}	7.3227×10^{1}	Exponential_1	3.9200×10^{2}	2.8106×10^{3}	6.1515×10^{2}	4.9085×10^{2}
	Exponential_2	3.9007×10^{2}	7.3063×10^{2}	4.1744×10^{2}	6.3802×10^{1}	Exponential_2	4.7908×10^{2}	**8.1095×10^{2}**	6.1817×10^{2}	**7.2382×10^{1}**
	Exponential_3	**3.9000×10^{2}**	9.0772×10^{2}	4.2429×10^{2}	9.6564×10^{1}	Exponential_3	3.9006×10^{2}	2.9396×10^{3}	6.9200×10^{2}	6.6627×10^{2}
	Exponential_4	3.9013×10^{2}	5.4184×10^{2}	4.1655×10^{2}	3.5768×10^{1}	Exponential_4	4.8828×10^{2}	1.1518×10^{3}	6.4499×10^{2}	1.5060×10^{2}
	Exponential_5	**3.9000×10^{2}**	1.0517×10^{3}	4.4257×10^{2}	1.4997×10^{2}	Exponential_5	**3.9001×10^{2}**	2.7678×10^{3}	6.7424×10^{2}	6.4280×10^{2}
	Exponential_6	3.9004×10^{2}	4.7407×10^{2}	4.0836×10^{2}	2.6488×10^{1}	Exponential_6	4.6519×10^{2}	2.0147×10^{3}	6.5145×10^{2}	2.8388×10^{2}
	Cosine_1	**3.9000×10^{2}**	6.1903×10^{2}	4.2205×10^{2}	4.8758×10^{1}	Cosine_1	4.5850×10^{2}	2.3449×10^{3}	7.1282×10^{2}	3.4454×10^{2}
	Cosine_2	3.9002×10^{2}	8.5583×10^{2}	4.1202×10^{2}	8.2736×10^{1}	Cosine_2	3.9079×10^{2}	3.2281×10^{3}	7.6770×10^{2}	8.2332×10^{2}
	Cosine_3	3.9010×10^{2}	8.8341×10^{2}	4.2415×10^{2}	8.8994×10^{1}	Cosine_3	5.1242×10^{2}	1.8003×10^{3}	7.3318×10^{2}	2.8816×10^{2}
	Cosine_4	3.9001×10^{2}	**4.0870×10^{2}**	**3.9875×10^{2}**	7.7194	**Cosine_4**	3.9074×10^{2}	1.1216×10^{3}	**5.3644×10^{2}**	1.8263×10^{2}
f_{14}	Straight_1	**-3.3000×10^{2}**	-3.2999×10^{2}	-3.2999×10^{2}	2.3198×10^{-8}	Straight_1	-3.2999×10^{2}	-3.2784×10^{2}	-3.2924×10^{2}	7.3617×10^{-1}
	Straight_2	**-3.3000×10^{2}**	**-3.3000×10^{2}**	**-3.3000×10^{2}**	**6.0514×10^{-14}**	Straight_2	**-3.3000×10^{2}**	-3.2801×10^{2}	-3.2962×10^{2}	5.9806×10^{-1}
	Threshold_1	**-3.3000×10^{2}**	-3.2999×10^{2}	-3.2999×10^{2}	1.1729×10^{-7}	Threshold_1	-3.2877×10^{2}	-3.2057×10^{2}	-3.2511×10^{2}	1.7985
	Threshold_2	**-3.3000×10^{2}**	-3.2901×10^{2}	-3.2997×10^{2}	1.7860×10^{-1}	Threshold_2	**-3.3000×10^{2}**	**-3.2901×10^{2}**	-3.2977×10^{2}	4.2082×10^{-1}
	Threshold_3	**-3.3000×10^{2}**	-3.2901×10^{2}	-3.2997×10^{2}	1.7859×10^{-1}	Threshold_3	**-3.3000×10^{2}**	-3.2801×10^{2}	-3.2974×10^{2}	4.9350×10^{-1}
	Threshold_4	**-3.3000×10^{2}**	**-3.3000×10^{2}**	**-3.3000×10^{2}**	1.4154×10^{-13}	Threshold_4	**-3.3000×10^{2}**	-3.2303×10^{2}	-3.2718×10^{2}	1.5641
	Exponential_1	**-3.3000×10^{2}**	-3.2999×10^{2}	-3.2999×10^{2}	3.0646×10^{-8}	Exponential_1	**-3.3000×10^{2}**	-3.2897×10^{2}	-3.2988×10^{2}	3.1104×10^{-1}
	Exponential_2	**-3.3000×10^{2}**	**-3.3000×10^{2}**	**-3.3000×10^{2}**	1.0062×10^{-13}	Exponential_2	**-3.3000×10^{2}**	**-3.2901×10^{2}**	-3.2983×10^{2}	3.7070×10^{-1}
	Exponential_3	**-3.3000×10^{2}**	-3.2999×10^{2}	-3.2999×10^{2}	1.8858×10^{-6}	Exponential_3	**-3.3000×10^{2}**	-3.2900×10^{2}	-3.2990×10^{2}	2.9728×10^{-1}
	Exponential_4	**-3.3000×10^{2}**	**-3.3000×10^{2}**	**-3.3000×10^{2}**	1.0934×10^{-13}	**Exponential_4**	**-3.3000×10^{2}**	**-3.2901×10^{2}**	**-3.2997×10^{2}**	**1.7860×10^{-1}**
	Exponential_5	**-3.3000×10^{2}**	-3.2901×10^{2}	-3.2997×10^{2}	1.7852×10^{-1}	Exponential_5	**-3.3000×10^{2}**	-3.2891×10^{2}	-3.2980×10^{2}	3.8883×10^{-1}
	Exponential_6	**-3.3000×10^{2}**	-3.2901×10^{2}	-3.2997×10^{2}	1.7860×10^{-1}	Exponential_6	**-3.3000×10^{2}**	**-3.2901×10^{2}**	-3.2987×10^{2}	3.3822×10^{-1}
	Cosine_1	**-3.3000×10^{2}**	-3.2999×10^{2}	-3.2999×10^{2}	4.0641×10^{-9}	Cosine_1	**-3.3000×10^{2}**	-3.2777×10^{2}	-3.2947×10^{2}	6.9220×10^{-1}
	Cosine_2	**-3.3000×10^{2}**	**-3.3000×10^{2}**	**-3.3000×10^{2}**	2.9464×10^{-13}	Cosine_2	**-3.3000×10^{2}**	-3.2702×10^{2}	-3.2920×10^{2}	1.0082
	Cosine_3	**-3.3000×10^{2}**	-3.2999×10^{2}	-3.2999×10^{2}	1.5012×10^{-11}	Cosine_3	**-3.3000×10^{2}**	-3.2747×10^{2}	-3.2933×10^{2}	7.1582×10^{-1}
	Cosine_4	**-3.3000×10^{2}**	**-3.3000×10^{2}**	**-3.3000×10^{2}**	2.0231×10^{-13}	Cosine_4	**-3.3000×10^{2}**	-3.2701×10^{2}	-3.2924×10^{2}	9.1373×10^{-1}

Table 4. Experimental results of difference HS algorithms.

No.	Dimension (D) = 30						Dimension (D) = 100					
	Algorithm	**Strategy**	**Min**	**Max**	**Mean**	**SD**	**Algorithm**	**Strategy**	**Min**	**Max**	**Mean**	**SD**
1	HS	–	9.5152×10^{-1}	7.2900	3.9526	1.8888	HS	–	9.8221×10^{3}	1.4318×10^{4}	1.2247×10^{4}	1.1304×10^{3}
	IHS	–	1.8017×10^{-7}	4.5253×10^{-7}	3.4508×10^{-7}	6.9069×10^{-8}	IHS	–	9.3279×10^{3}	1.5282×10^{4}	1.2496×10^{4}	1.2772×10^{3}
	SGHS	–	7.6930×10^{-10}	1.5045×10^{-8}	5.0535×10^{-9}	3.1296×10^{-9}	SGHS	–	6.6607×10^{-1}	2.7569	1.5343	4.7765×10^{-1}
	NGHS	–	1.7413×10^{-17}	2.3048×10^{-15}	3.4620×10^{-16}	4.7004×10^{-16}	NGHS	–	3.0447×10^{-4}	1.3603×10^{-3}	7.4741×10^{-4}	2.1074×10^{-4}
	DANGHS	Exponential_6	**2.3735×10^{-38}**	**3.7601×10^{-30}**	**1.8344×10^{-31}**	**7.0604×10^{-31}**	**DANGHS**	Exponential_6	**1.6615×10^{-16}**	**8.0332×10^{-14}**	**1.2209×10^{-14}**	**1.9706×10^{-14}**
2	HS	–	3.0000	1.7000×10^{1}	9.3000	3.7162	HS	–	8.4840×10^{3}	1.6381×10^{4}	1.2242×10^{4}	1.6586×10^{3}
	IHS	–	**0.0000**	3.0000	9.3333×10^{-1}	1.0306	IHS	–	1.0060×10^{4}	1.5588×10^{4}	1.2560×10^{4}	1.3116×10^{3}
	SGHS	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**	SGHS	–	3.0000	1.8000×10^{1}	8.7667	3.1271
	NGHS	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**NGHS**	–	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	DANGHS	Exponential_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**DANGHS**	Exponential_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**
3	HS	–	3.8826×10^{-2}	2.1547×10^{-1}	8.3000×10^{-2}	3.9484×10^{-2}	HS	–	5.2475×10^{1}	6.6253×10^{1}	6.0705×10^{1}	4.1892
	IHS	–	1.8454×10^{-3}	2.7586×10^{-2}	3.1832×10^{-3}	4.5541×10^{-3}	IHS	–	5.1429×10^{1}	6.9346×10^{1}	6.0238×10^{1}	4.2859
	SGHS	–	1.2406×10^{-4}	2.3354×10^{-4}	1.6844×10^{-4}	2.7009×10^{-5}	SGHS	–	6.9687×10^{-2}	4.0539×10^{-1}	2.2004×10^{-1}	7.5524×10^{-2}
	NGHS	–	2.8122×10^{-10}	4.8894×10^{-9}	1.3786×10^{-9}	9.1666×10^{-10}	NGHS	–	8.0120×10^{-3}	1.8302×10^{-2}	1.4477×10^{-2}	2.3050×10^{-3}
	DANGHS	Exponential_6	**5.1270×10^{-23}**	**2.5548×10^{-17}**	**1.9511×10^{-18}**	**5.5869×10^{-18}**	**DANGHS**	Exponential_6	**1.7326×10^{-9}**	**2.2346×10^{-8}**	**7.9778×10^{-9}**	**5.9706×10^{-9}**
4	HS	–	1.3615×10^{3}	8.1756×10^{3}	3.7966×10^{3}	1.4524×10^{3}	HS	–	1.2355×10^{5}	2.2504×10^{5}	1.8030×10^{5}	2.0587×10^{4}
	IHS	–	1.5474×10^{3}	6.0226×10^{3}	3.8475×10^{3}	1.1754×10^{3}	IHS	–	1.2992×10^{5}	2.3481×10^{5}	1.7522×10^{5}	2.7139×10^{4}
	SGHS	–	2.0150×10^{1}	**1.0642×10^{2}**	**5.2245×10^{1}**	**2.2107×10^{1}**	SGHS	–	1.7856×10^{4}	3.1133×10^{4}	2.2834×10^{4}	2.8349×10^{3}
	NGHS	-	2.8355×10^{1}	1.4013×10^{2}	6.5269×10^{1}	3.3421×10^{1}	NGHS	–	7.4976×10^{3}	**1.2945×10^{4}**	9.7007×10^{3}	**1.6021×10^{3}**
	DANGHS	Threshold_4	**1.5038×10^{1}**	1.5980×10^{2}	6.0249×10^{1}	3.5686×10^{1}	**DANGHS**	Exponential_2	**4.6763×10^{3}**	1.3135×10^{4}	**8.6301×10^{3}**	1.9698×10^{3}
5	HS	–	1.0212	1.1106	1.0591	2.2096×10^{-2}	HS	–	9.5506×10^{1}	1.4758×10^{2}	1.1631×10^{2}	1.1240×10^{1}
	IHS	–	1.2959×10^{-7}	**3.4241×10^{-2}**	**7.5274×10^{-3}**	**9.2294×10^{-3}**	IHS	–	7.5548×10^{1}	1.4827×10^{2}	1.0997×10^{2}	1.4826×10^{1}
	SGHS	–	1.7833×10^{-2}	2.3440×10^{-1}	1.0043×10^{-1}	5.1304×10^{-2}	SGHS	–	4.4296×10^{-1}	8.8847×10^{-1}	6.8599×10^{-1}	9.9379×10^{-2}
	NGHS	–	**3.3307×10^{-16}**	2.5387×10^{-1}	6.1311×10^{-2}	4.9633×10^{-2}	NGHS	–	1.5343×10^{-4}	9.9663×10^{-2}	1.7168×10^{-2}	2.2003×10^{-2}
	DANGHS	Threshold_4	6.6613×10^{-16}	8.0817×10^{-2}	3.1209×10^{-2}	2.3482×10^{-2}	**DANGHS**	Exponential_2	**3.5352×10^{-11}**	**4.8906×10^{-2}**	**6.4754×10^{-3}**	**9.8313×10^{-3}**
6	HS	-	1.9421×10^{-2}	1.3050	4.9617×10^{-1}	4.2318×10^{-1}	HS	–	1.0882×10^{1}	1.2567×10^{1}	1.1743×10^{1}	3.8517×10^{-1}
	IHS	–	3.4980×10^{-4}	1.3915	2.2199×10^{-1}	3.4543×10^{-1}	IHS	–	1.0987×10^{1}	1.2722×10^{1}	1.1852×10^{1}	4.3446×10^{-1}
	SGHS	–	1.7703×10^{-5}	4.5526×10^{-5}	3.0830×10^{-5}	6.1683×10^{-6}	SGHS	–	6.3791×10^{-2}	4.5729×10^{-1}	2.4057×10^{-1}	1.2018×10^{-1}
	NGHS	–	7.7839×10^{-10}	2.0025×10^{-8}	5.7085×10^{-9}	5.2959×10^{-9}	NGHS	–	2.6973×10^{-3}	5.3184×10^{-3}	3.6500×10^{-3}	5.4706×10^{-4}
	DANGHS	Exponential_6	**4.9294×10^{-14}**	**2.2338×10^{-13}**	**9.6308×10^{-14}**	**3.4392×10^{-14}**	**DANGHS**	Exponential_6	**1.0897×10^{-9}**	**2.9882×10^{-8}**	**9.3030×10^{-9}**	**7.9441×10^{-9}**
7	HS	–	9.6358×10^{1}	3.9298×10^{2}	1.8204×10^{2}	5.9631×10^{1}	HS	–	3.2565×10^{6}	9.1894×10^{6}	5.9320×10^{6}	1.2941×10^{6}
	IHS	-	1.7586×10^{1}	2.1565×10^{3}	3.6705×10^{2}	5.5299×10^{2}	IHS	–	4.1100×10^{6}	8.2424×10^{6}	5.7186×10^{6}	1.0494×10^{6}
	SGHS	–	9.0932	2.0293×10^{3}	1.7534×10^{2}	3.7957×10^{2}	SGHS	–	1.0832×10^{2}	2.8592×10^{3}	5.1645×10^{2}	4.7866×10^{2}
	NGHS	–	**6.6756×10^{-4}**	2.3003×10^{2}	1.4971×10^{1}	4.0757×10^{1}	NGHS	–	2.1179×10^{1}	1.4411×10^{3}	2.8501×10^{2}	2.8532×10^{2}
	DANGHS	Straight_1	9.3515×10^{-3}	**2.2089×10^{1}**	**1.0089×10^{1}**	**8.7452**	**DANGHS**	Threshold_3	**5.6804×10^{-2}**	**1.1562×10^{3}**	**6.1559×10^{1}**	**2.0554×10^{2}**

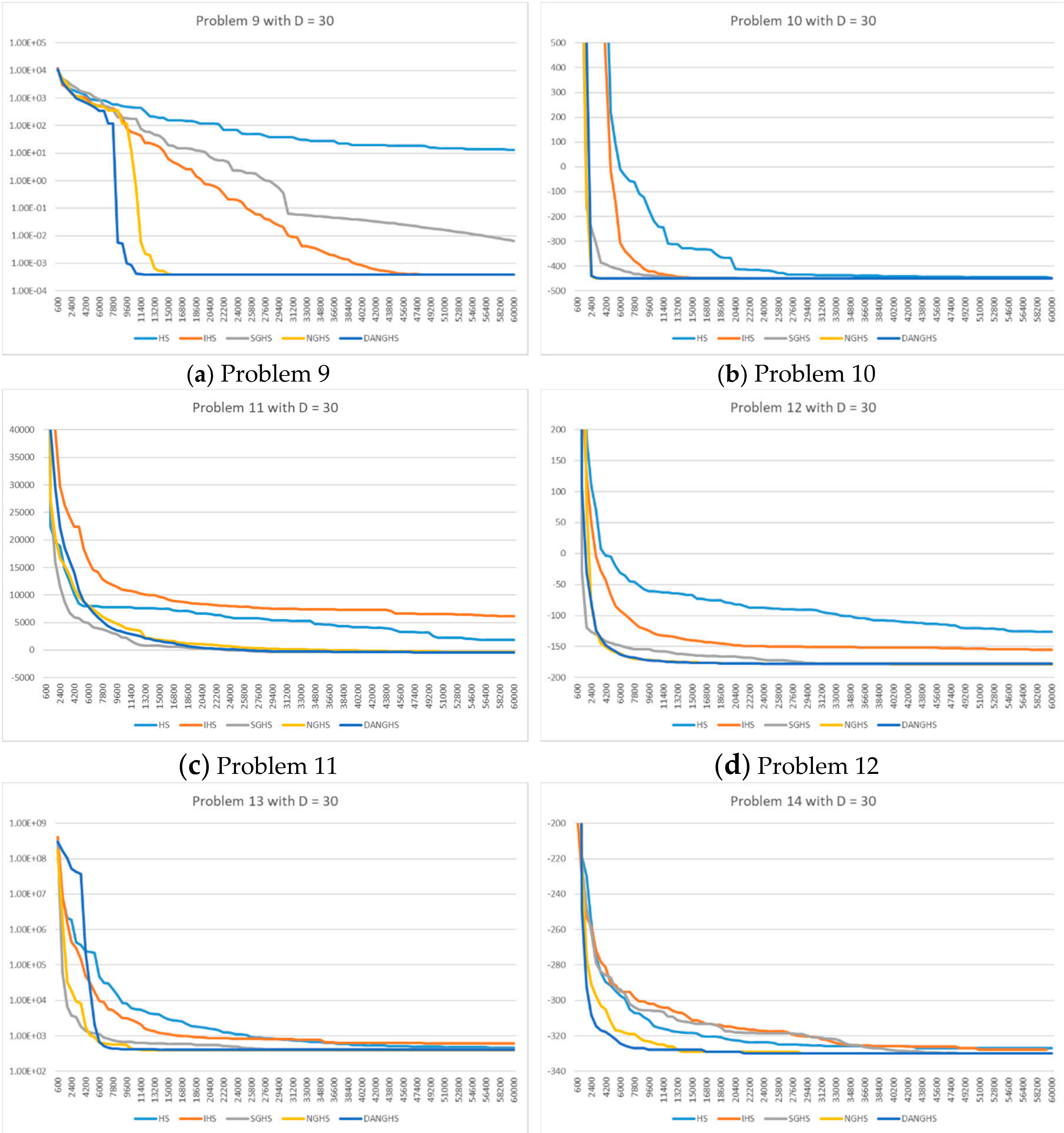
Table 4. *Cont.*

No.	Dimension (D) = 30						Dimension (D) = 100					
	Algorithm	**Strategy**	**Min**	**Max**	**Mean**	**SD**	**Algorithm**	**Strategy**	**Min**	**Max**	**Mean**	**SD**
8	HS	–	3.0572×10^{-2}	2.0546	4.6448×10^{-1}	6.5390×10^{-1}	HS	–	2.1874×10^{2}	2.8758×10^{2}	2.5192×10^{2}	1.6481×10^{1}
	IHS	–	4.1948×10^{-5}	4.5484	1.2420	9.8291×10^{-1}	IHS	–	2.0838×10^{2}	2.8193×10^{2}	2.4294×10^{2}	1.8844×10^{1}
	SGHS	–	3.7300×10^{-7}	9.9498×10^{-1}	1.3267×10^{-1}	3.3822×10^{-1}	SGHS	–	3.2260×10^{-2}	9.1200	4.5553	2.2588
	NGHS	-	**0.0000**	1.6069×10^{-11}	9.3241×10^{-13}	3.2209×10^{-12}	NGHS	–	1.2729×10^{-3}	1.0102	2.1542×10^{-1}	3.9474×10^{-1}
	DANGHS	Threshold_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**DANGHS**	Exponential_3	**3.5170×10^{-8}**	**5.9362×10^{-1}**	**2.7729×10^{-2}**	**1.0943×10^{-1}**
9	HS	–	6.9691	3.8058×10^{1}	1.8422×10^{1}	6.8471	HS	–	4.5568×10^{3}	6.9091×10^{3}	5.7964×10^{3}	5.5601×10^{2}
	IHS	–	3.8186×10^{-4}	5.2695×10^{-1}	1.7934×10^{-2}	9.4522×10^{-2}	IHS	–	4.2297×10^{3}	6.4098×10^{3}	5.4659×10^{3}	5.5784×10^{2}
	SGHS	–	2.3563×10^{-3}	3.6545×10^{-2}	1.3771×10^{-2}	7.5711×10^{-3}	SGHS	–	7.2936	3.8640×10^{1}	1.5981×10^{1}	7.4744
	NGHS	–	**3.8183×10^{-4}**	**3.8183×10^{-4}**	**3.8183×10^{-4}**	5.5493×10^{-13}	NGHS	–	3.3819×10^{-3}	6.8492×10^{-2}	1.1069×10^{-2}	1.3684×10^{-2}
	DANGHS	Threshold_2	**3.8183×10^{-4}**	**3.8183×10^{-4}**	**3.8183×10^{-4}**	**1.3763×10^{-13}**	**DANGHS**	Threshold_2	**1.2728×10^{-3}**	**1.2728×10^{-3}**	**1.2728×10^{-3}**	**1.4537×10^{-9}**
10	HS	–	-4.4898×10^{2}	-4.3989×10^{2}	-4.4573×10^{2}	2.2745	HS	–	1.0055×10^{4}	1.6736×10^{4}	1.2963×10^{4}	1.7748×10^{3}
	IHS	–	**-4.5000×10^{2}**	-4.4999×10^{2}	-4.4999×10^{2}	1.1657×10^{-7}	IHS	–	1.0425×10^{4}	1.4910×10^{4}	1.2856×10^{4}	1.2084×10^{3}
	SGHS	–	**-4.5000×10^{2}**	-4.4999×10^{2}	-4.4999×10^{2}	2.7913×10^{-9}	SGHS	–	-4.4918×10^{2}	-4.4701×10^{2}	-4.4841×10^{2}	5.8573×10^{-1}
	NGHS	–	**-4.5000×10^{2}**	**-4.5000×10^{2}**	**-4.5000×10^{2}**	6.9619×10^{-14}	NGHS	–	**-4.5000×10^{2}**	-4.4999×10^{2}	-4.4999×10^{2}	3.0551×10^{-4}
	DANGHS	Exponential_2	**-4.5000×10^{2}**	**-4.5000×10^{2}**	**-4.5000×10^{2}**	**5.4916×10^{-14}**	**DANGHS**	Threshold_2	**-4.5000×10^{2}**	**-4.5000×10^{2}**	**-4.5000×10^{2}**	**2.4117×10^{-13}**
11	HS	–	1.5142×10^{3}	7.6068×10^{3}	3.4157×10^{3}	1.2671×10^{3}	HS	–	1.4921×10^{5}	2.5525×10^{5}	1.9430×10^{5}	2.3256×10^{4}
	IHS	–	1.0412×10^{3}	7.3312×10^{3}	3.3180×10^{3}	1.3292×10^{3}	IHS	–	1.4941×10^{5}	2.7307×10^{5}	2.0189×10^{5}	2.7108×10^{4}
	SGHS	–	-4.3194×10^{2}	**-3.2048×10^{2}**	**-3.9763×10^{2}**	**2.7006×10^{1}**	SGHS	–	1.5646×10^{4}	3.5936×10^{4}	2.6350×10^{4}	4.0064×10^{3}
	NGHS	–	-4.2591×10^{2}	-1.9340×10^{2}	-3.3680×10^{2}	6.4820×10^{1}	NGHS	–	9.0685×10^{3}	1.7976×10^{4}	1.1950×10^{4}	**2.0823×10^{3}**
	DANGHS	Threshold_4	**-4.4289×10^{2}**	-2.5392×10^{2}	-3.7419×10^{2}	4.4269×10^{1}	**DANGHS**	Exponential_2	**6.9718×10^{3}**	**1.7181×10^{4}**	**1.1471×10^{4}**	2.4981×10^{3}
12	HS	–	-1.7016×10^{2}	-1.3324×10^{2}	-1.5876×10^{2}	9.4348	HS	–	3.2343×10^{3}	5.9684×10^{3}	4.7002×10^{3}	6.7476×10^{2}
	IHS	–	-1.7607×10^{2}	-1.3892×10^{2}	-1.5831×10^{2}	7.7527	IHS	–	3.4934×10^{3}	6.5826×10^{3}	4.8855×10^{3}	8.4393×10^{2}
	SGHS	–	-1.7830×10^{2}	-1.7149×10^{2}	-1.7583×10^{2}	1.6385	SGHS	–	-1.3057×10^{2}	-1.5099×10^{1}	-7.2944×10^{1}	2.8794×10^{1}
	NGHS	–	**-1.7913×10^{2}**	-1.7532×10^{2}	**-1.7829×10^{2}**	**6.4615×10^{-1}**	NGHS	–	-1.5784×10^{2}	-1.1939×10^{2}	-1.4231×10^{2}	1.0814×10^{1}
	DANGHS	Straight_2	-1.7894×10^{2}	**-1.7562×10^{2}**	-1.7821×10^{2}	7.6813×10^{-1}	**DANGHS**	Exponential_6	**-1.7066×10^{2}**	**-1.4082×10^{2}**	**-1.6037×10^{2}**	**6.8764**
13	HS	–	4.7650×10^{2}	2.5184×10^{3}	6.6765×10^{2}	3.6290×10^{2}	HS	–	4.0732×10^{6}	8.7806×10^{6}	6.0785×10^{6}	1.0919×10^{6}
	IHS	–	4.1262×10^{2}	1.7487×10^{3}	5.7845×10^{2}	2.3203×10^{2}	IHS	–	4.5479×10^{6}	8.5517×10^{6}	6.3527×10^{6}	1.2268×10^{6}
	SGHS	–	3.9001×10^{2}	5.6058×10^{2}	4.6408×10^{2}	4.0896×10^{1}	SGHS	–	6.4618×10^{2}	2.5249×10^{3}	9.7963×10^{2}	4.0336×10^{2}
	NGHS	–	**3.9000×10^{2}**	4.0874×10^{2}	**3.9494×10^{2}**	**5.7399**	NGHS	–	4.6424×10^{2}	1.6001×10^{3}	7.1517×10^{2}	2.9163×10^{2}
	DANGHS	Cosine_4	3.9001×10^{2}	**4.0870×10^{2}**	3.9875×10^{2}	7.7194	**DANGHS**	Cosine_4	**3.9074×10^{2}**	**1.1216×10^{3}**	**5.3644×10^{2}**	**1.8263×10^{2}**
14	HS	–	-3.2997×10^{2}	-3.2788×10^{2}	-3.2938×10^{2}	7.4966×10^{-1}	HS	–	-9.3948×10^{1}	-6.2235	-5.1541×10^{1}	2.2413×10^{1}
	IHS	–	-3.2999×10^{2}	-3.2749×10^{2}	-3.2897×10^{2}	6.9697×10^{-1}	IHS	–	-1.1215×10^{2}	-3.3592×10^{1}	-6.5372×10^{1}	1.6227×10^{1}
	SGHS	–	**3.9000×10^{2}**	-3.2901×10^{2}	-3.2993×10^{2}	2.4813×10^{-1}	SGHS	–	-3.2900×10^{2}	-3.2101×10^{2}	-3.2614×10^{2}	2.2502
	NGHS	–	**3.9000×10^{2}**	-3.2999×10^{2}	-3.2999×10^{2}	1.1444×10^{-12}	NGHS	–	**3.9000×10^{2}**	-3.2798×10^{2}	-3.2979×10^{2}	5.4096×10^{-1}
	DANGHS	Straight_2	**3.9000×10^{2}**	**3.9000×10^{2}**	**3.9000×10^{2}**	**6.0514×10^{-14}**	**DANGHS**	Exponential_4	**3.9000×10^{2}**	**-3.2901×10^{2}**	**-3.2997×10^{2}**	**1.7860×10^{-1}**



(**a**) Problem 1 (**b**) Problem 2 (**c**) Problem 3 (**d**) Problem 4 (**e**) Problem 5 (**f**) Problem 6 (**g**) Problem 7 (**h**) Problem 8

Figure 7. Typical convergence graph of five different algorithms for problems 1 to 8 (D = 30). (**a**) Problem 1; (**b**) Problem 2; (**c**) Problem 3; (**d**) Problem 4; (**e**) Problem 5; (**f**) Problem 6; (**g**) Problem 7; (**h**) Problem 8.

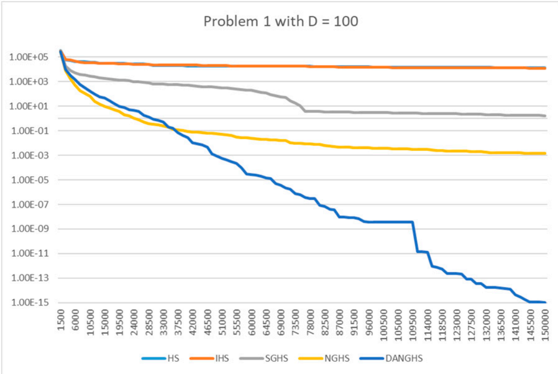


(**a**) Problem 9 (**b**) Problem 10

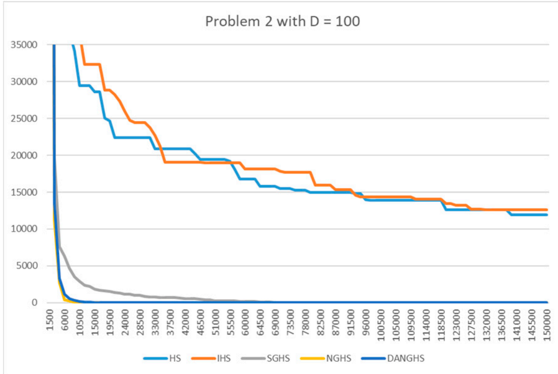
(**c**) Problem 11 (**d**) Problem 12

(**e**) Problem 13 (**f**) Problem 14

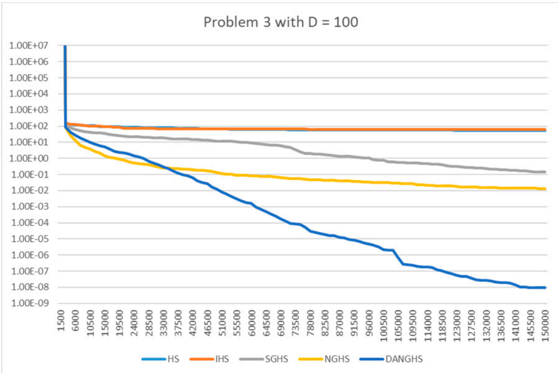
Figure 8. Typical convergence graph of five different algorithms for problems 9 to 14 (D = 30). (**a**) Problem 9; (**b**) Problem 10; (**c**) Problem 11; (**d**) Problem 12; (**e**) Problem 13; (**f**) Problem 14.



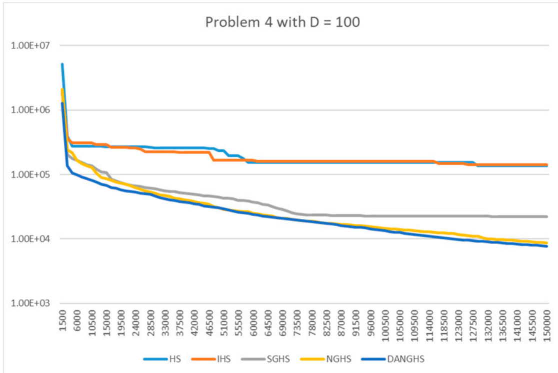
(**a**) Problem 1



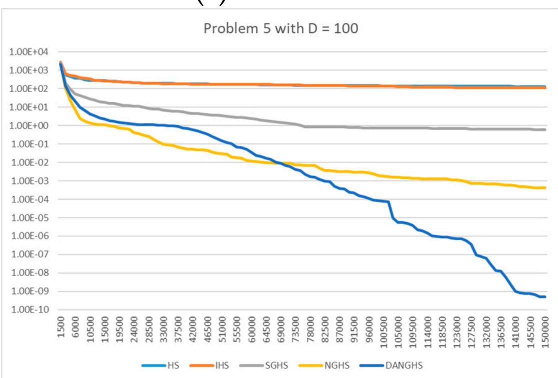
(**b**) Problem 2



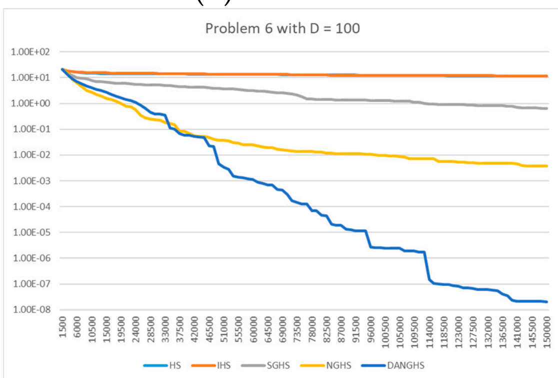
(**c**) Problem 3



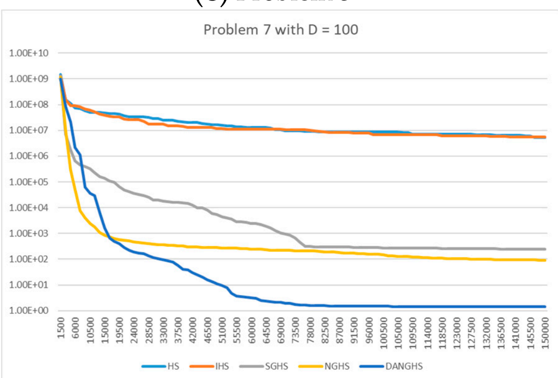
(**d**) Problem 4



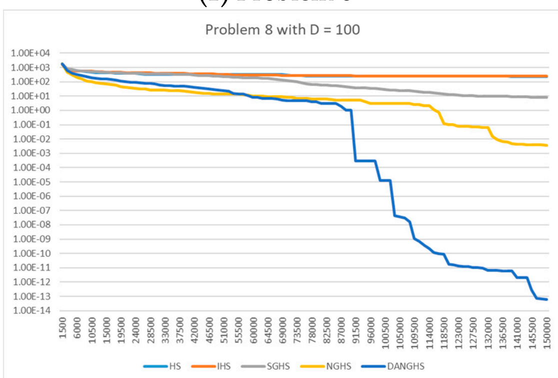
(**e**) Problem 5



(**f**) Problem 6

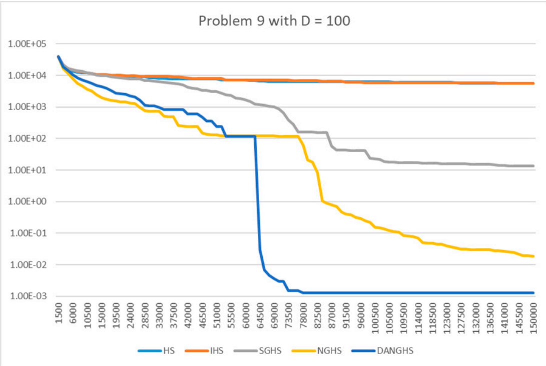


(**g**) Problem 7

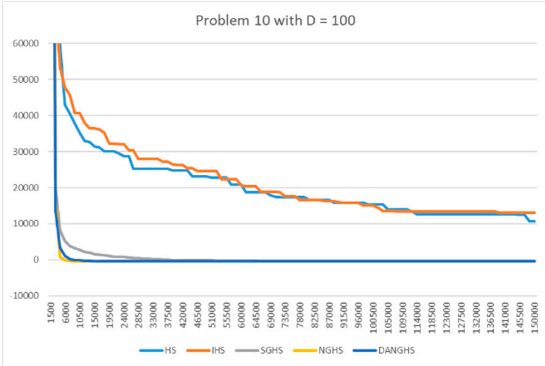


(**h**) Problem 8

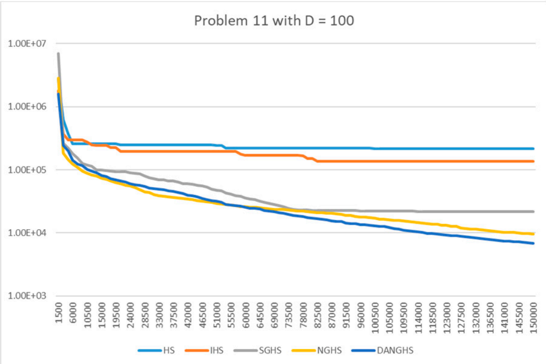
Figure 9. Typical convergence graph of five different algorithms for problems 1 to 8 (D = 100). (**a**) Problem 1; (**b**) Problem 2; (**c**) Problem 3; (**d**) Problem 4; (**e**) Problem 5; (**f**) Problem 6; (**g**) Problem 7; (**h**) Problem 8.



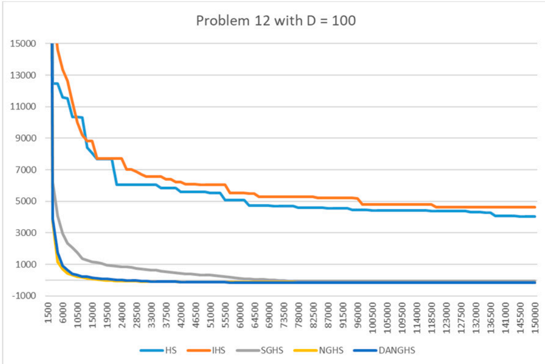
(**a**) Problem 9



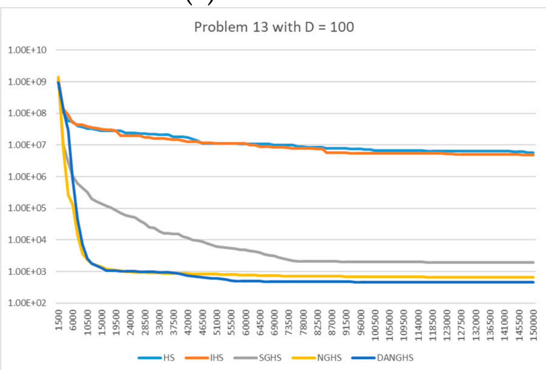
(**b**) Problem 10



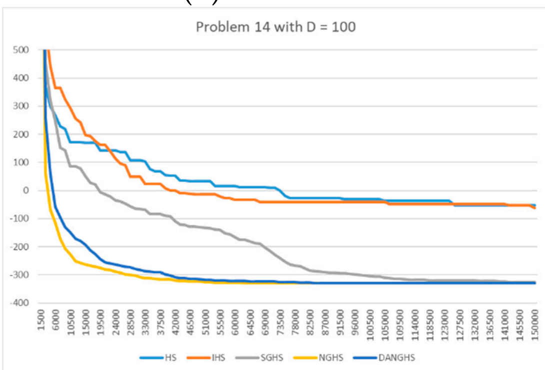
(**c**) Problem 11



(**d**) Problem 12



(**e**) Problem 13



(**f**) Problem 14

Figure 10. Typical convergence graph of five different algorithms for problems 9 to 14 (D = 100). (**a**) Problem 9; (**b**) Problem 10; (**c**) Problem 11; (**d**) Problem 12; (**e**) Problem 13; (**f**) Problem 14.

5. Conclusions

We presented a DANGHS algorithm, which combines NGHS and the dynamic adjustment strategy for genetic mutation probability. Moreover, the extensive computational experiments and comparisons were carried out for 14 benchmark continuous optimization problems. According to the extensive computational results, there are several findings in this paper worth summarizing.

First, different strategies are suitable for different problems.

1. The decreasing dynamic adjustment strategies should be applied to some problems in which the DANGHS algorithm needs a larger, p_m, in the early iterations, in order to have a larger probability of finding a better trial solution around the current one.
2. The increasing dynamic adjustment strategies should be applied to other problems. For these problems, if the current solution is trapped in a local optimum, the DANGHS algorithm requires a larger probability, p_m, in later iterations in order to avoid the local optima.

3. The periodic dynamic adjustment strategy can find the best objective function value for problem 13. These particular results show that there are not only two kinds of adjustment strategies, decreasing and increasing strategies, which are suitable for all problems. This viewpoint is different from the common views: the adjustment strategy is as small as possible or as large as possible with a generation number. For a specific problem, the periodic dynamic adjustment strategy could have better performance in comparison with other decreasing or increasing strategies. Therefore, these results inspire us to further investigate this kind of periodic dynamic adjustment strategy in future experiments.

Second, the extensive experimental results showed that the DANGHS algorithm had better searching performance in comparison with other HS algorithms for D = 30 and 100 in most problems. Particularly for D = 100, the DANGHS algorithm could search the best objective function value in all 14 problems. In other words, the DANGHS had superior searching performance in high-dimensional problems. According to the numerical results, we proofed that algorithms with dynamic parameters, such as the DANGHS algorithm and the IHS algorithm, could have better searching performance than algorithms without dynamic parameters, such as the NGHS algorithm and the HS algorithm. Moreover, according to these results, we proofed that the viewpoint presented in previous studies is suitable for the NGHS algorithm. This viewpoint states that appropriate parameters can enhance the searching ability of a metaheuristic algorithm.

Finally, the DANGHS algorithm using Pseudocode 4 was more efficient than that using Pseudocode 3. In Pseudocode 3, the algorithm generates a new harmony, and then with p_m probability, the algorithm abandoned it to generate a mutated harmony. Obviously, it was redundant and inefficient. Therefore, we modified the procedure in Pseudocode 3 and presented a more efficient Pseudocode 4 in this paper. In conclusion, the DANGHS algorithm is a more efficient and effective algorithm.

Author Contributions: P.-C.S. designed the algorithm, conducted the experiments, analyzed the experimental results, wrote the paper and polished the English; C.-Y.C. and X.L. supervised the research work and provided helpful suggestions.

Funding: This research received no external funding.

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

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

References

1. Chiu, C.Y.; Fan, S.K.S.; Shih, P.C.; Weng, Y.H. Applying HBMO-based SOM in predicting the Taiwan steel price fluctuation. *Int. J. Electron. Bus. Manag.* **2014**, *12*, 1–14.
2. Chen, K.H.; Chen, L.F.; Su, C.T. A new particle swarm feature selection method for classification. *J. Intell. Inf. Syst.* **2014**, *42*, 507–530. [CrossRef]
3. Petrović, M.; Vuković, N.; Mitić, M.; Miljković, Z. Integration of process planning and scheduling using chaotic particle swarm optimization algorithm. *Expert Syst. Appl.* **2016**, *64*, 569–588. [CrossRef]
4. Khaled, N.; Hemayed, E.E.; Fayek, M.B. A GA-based approach for epipolar geometry estimation. *Int. J. Pattern Recognit. Artif. Intell.* **2013**, *27*, 1355014. [CrossRef]
5. Metawaa, N.; Hassana, M.K.; Elhoseny, M. Genetic algorithm based model for optimizing bank lending decisions. *Expert Syst. Appl.* **2017**, *80*, 75–82. [CrossRef]
6. Song, S. Design of distributed database systems: an iterative genetic algorithm. *J. Intell. Inf. Syst.* **2015**, *45*, 29–59. [CrossRef]
7. Gambardella, L.M.; Montemanni, R.; Weyland, D. Coupling ant colony systems with strong local searches. *Eur. J. Oper. Res.* **2012**, *220*, 831–843. [CrossRef]
8. D'Andreagiovanni, F.; Mett, F.; Nardin, A.; Pulaj, J. Integrating LP-guided variable fixing with MIP heuristics in the robust design of hybrid wired-wireless FTTx access networks. *Appl. Soft. Comput.* **2017**, *61*, 1074–1087. [CrossRef]
9. D'Andreagiovanni, F.; Nardin, A. Towards the fast and robust optimal design of wireless body area networks. *Appl. Soft. Comput.* **2015**, *37*, 971–982. [CrossRef]

10. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995.
11. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
12. Ozcan, E.; Mohan, C.K. Analysis of a simple particle swarm optimization system. *Intell. Eng. Syst. Artif. Neural Netw.* **1998**, *8*, 253–258.
13. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
14. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992.
15. Gordini, N. A genetic algorithm approach for SMEs bankruptcy prediction: Empirical evidence from Italy. *Expert Syst. Appl.* **2014**, *41*, 6433–6445. [CrossRef]
16. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
17. Haddad, O.B.; Afshar, A.; Marino, M.A. Honey-bees mating optimization (HBMO) algorithm: A new heuristic approach for water resources optimization. *Water Resour. Manag.* **2006**, *20*, 661–680. [CrossRef]
18. Ouyang, A.; Peng, X.; Liu, Y.; Fan, L.; Li, K. An efficient hybrid algorithm based on HS and SFLA. *Int. J. Pattern Recognit. Artif. Intell.* **2016**, *30*, 1659012. [CrossRef]
19. Tavakoli, S.; Valian, E.; Mohanna, S. Feedforward neural network training using intelligent global harmony search. *Evol. Syst.* **2012**, *3*, 125–131. [CrossRef]
20. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
21. Assad, A.; Deep, K. A Hybrid Harmony search and Simulated Annealing algorithm for continuous optimization. *Inf. Sci.* **2018**, *450*, 246–266. [CrossRef]
22. Assad, A.; Deep, K. A two-phase harmony search algorithm for continuous optimization. *Comput. Intell.* **2017**, *33*, 1038–1075. [CrossRef]
23. Mahdavi, M.; Fesanghary, M.; Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **2007**, *188*, 1567–1579. [CrossRef]
24. Pan, Q.K.; Suganthan, P.N.; Tasgetiren, M.F.; Liang, J.J. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Appl. Math. Comput.* **2010**, *216*, 830–848. [CrossRef]
25. Zou, D.; Gao, L.; Li, S.; Wu, J.; Wang, X. A novel global harmony search algorithm for task assignment problem. *J. Syst. Softw.* **2010**, *83*, 1678–1688. [CrossRef]
26. Zou, D.; Gao, L.; Wu, J.; Li, S.; Li, Y. A novel global harmony search algorithm for reliability problems. *Comput. Ind. Eng.* **2010**, *58*, 307–316. [CrossRef]
27. Zou, D.; Gao, L.; Wu, J.; Li, S. Novel global harmony search algorithm for unconstrained problems. *Neurocomputing* **2010**, *73*, 3308–3318. [CrossRef]
28. Valian, E.; Tavakoli, S.; Mohanna, S. An intelligent global harmony search approach to continuous optimization problems. *Appl. Math. Comput.* **2014**, *232*, 670–684. [CrossRef]
29. Kattan, A.; Abdullah, R. Training of feed-forward neural networks for pattern-classification applications using music inspired algorithm. *Inter. J. Comput. Sci. Inf. Secur.* **2011**, *9*, 44–57.
30. Lee, K.S.; Geem, Z.W. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Comput. Method Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [CrossRef]
31. Omran, M.G.H.; Mahdavi, M. Global-best harmony search. *Appl. Math. Comput.* **2008**, *198*, 643–656. [CrossRef]