J Intell Manuf (2019) 30:405–428
https://doi.org/10.1007/s10845-016-1255-5

Parallel chaotic local search enhanced harmony search algorithm for engineering design optimization

Jin Yi[1] · **Xinyu Li**[1] · **Chih-Hsing Chu**[2] · **Liang Gao**[1]

Received: 21 April 2016 / Accepted: 4 August 2016 / Published online: 18 August 2016

Abstract In this paper, we present a parallel chaotic local search enhanced harmony search algorithm (MHS–PCLS) for solving engineering design optimization problems. The concept of chaos has been previously successfully applied in metaheuristics. However, chaos sequences are sensitive to their initial conditions and cause unstable performance in algorithms. The proposed parallel chaotic local search method searches from several different initial points and diminishes the sensitivity of the initial condition, thereby increasing the robustness of the harmony search method. Numerical benchmark problems are tested to validate the effectiveness of MHS–PCLS. The simulation results confirm that MHS–PCLS obtains superior results for mathematical examples compared to other harmony search variants. Several well-known constrained engineering design problems are also tested using the new approach. The computational results demonstrate that the proposed MHS–PCLS algorithm requires a smaller number of function evaluations and in the majority of cases delivers improved and more robust results compare to other algorithms.

Keywords Harmony search · Parallel chaotic local search · Intersect mutation operator · Engineering design optimization

✉ Xinyu Li
lixinyu@mail.hust.edu.cn

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

[2] Department of Industrial Engineering and Engineering Management, National Tsing-Hua University, Hsinchu, Taiwan

Introduction

Nonlinearity widely exists in real world optimization problems and seriously complicates the search space, posing significant challenges in obtaining the global optimality of interest. Metaheuristic techniques frequently provide satisfactory solutions to these problems; hence, they have been a major focus in recent decades (Yang 2010). These techniques usually mimic some natural or social phenomenon, such as the biological evolutionary process [e.g., genetic algorithm (GA) (Sivaraj and Ravichandran 2011), differential evolution (DE) (Das and Suganthan 2011) and biogeography-based optimization (BBO) (Simon 2008)] or animal behavior [e.g., particle swarm optimization (PSO) (Eberhart et al. 1995), artificial bee colony (ABC) (Karaboga and Basturk 2007), ant colony optimization (ACO) (Dorigo et al. 2006) and cuckoo search (CS) (Yang and Deb 2009)].

The harmony search (HS) algorithm is a recently developed meta-heuristic algorithm (Geem et al. 2001). It imitates the music improvisation process during which musicians continuously adjust the pitch of their instruments to obtain improved harmony. The optimization process is similar to the music improvisation process. Each decision variable continuously changes its value during the search process to converge to the global best. HS has received significant attention and has been applied in many areas such as renewable energy systems (Askarzadeh and Zebarjadi 2014; Maleki and Pourfayaz 2015), image registration (García-Torres et al. 2014), robotics (Koceski et al. 2014; Kundu and Parhi 2016), job shop scheduling (Gao et al. 2014b), transportation (Zheng et al. 2016; Hosseini et al. 2014; Yassen et al. 2015), assembly sequence planning (Li et al. 2016), computer experiment designs(Yi et al. 2016b) and wireless sensor networks (Zeng and Dong 2016). A more detailed survey of HS applications was summarized by Manjarres et al. (2013). Several studies

have been proposed for obtaining improved HS performance and gaining a superior understanding of its concept, including exploratory power (Das et al. 2011), selection strategies (Al-Betar et al. 2013), stochastic analysis (Sarvari and Zamanifar 2012) and iterative convergence (Gao et al. 2014a).

With the development of nonlinear dynamics, the concept of chaos has been successfully applied in the area of optimization methods. Previously, chaotic sequences have been embedded into metaheuristic methods such as particle swarm optimization (Gao et al. 2012; Gandomi et al. 2013c), memetic differential evolution algorithm (Jia et al. 2011), artificial bee colony algorithm (Alatas 2010), firefly algorithm (Gandomi et al. 2013b), imperialist competitive algorithm (Talatahari et al. 2012), bat swarm optimization (Jordehi 2015), krill herd algorithm (Wang et al. 2014) and harmony search (Alatas 2010). In the above methods, sequences generated from different chaotic systems are used to substitute random numbers for different parameters. One of the advantages of chaotic sequences is that chaos can promote order from disorder. Similarly, many metaheuristic methods are inspired from biological systems where order can be realized from disorder. Owing to the similarities between chaos and metaheuristic methods, the use of chaos seems to improve the performance of metaheuristics (Kaveh 2014).

However, chaos is sensitive to its initial conditions, and even small changes in the parameters or the starting values can lead to vastly different future behaviors. Consequently, the performance of optimization algorithms is not stable. To increase the robustness of the optimization algorithm, in this paper, a parallel chaotic local search method is proposed and embedded into a modified harmony search with an intersect mutation operation. The proposed new harmony search variant is called parallel chaotic local search enhanced harmony search (MHS–PCLS). MHS–PCLS maintains the basic structure of HS and adopts an intersect mutation operation in the process of improving new harmonies, which helps to increase the diversity of harmony memory. The parallel chaotic local search is used to detracts the sensitivity of initial condition of chaotic maps and enhance the exploitation ability of MHS–PCLS. Several well-known numerical benchmark problems are tested to demonstrate the efficiency, accuracy and robustness of the new approach. The MHS–PCLS method is further combined with the improved Deb's constrained handling method for constrained optimization problems, which frequently occur in real world engineering design. The performance of the extended method is validated by several constrained engineering design problems and a complex case study of car side impact design. The test results confirm that the MHS–PCLS outperforms other algorithms.

The remander of this paper is organized as follows. In "Previous work" section, the previous work on harmony search is introduced. The new MHS–PCLS algorithm is proposed in "Proposed approach" section. "Numerical simulations and analysis" section provides a numerical simulation of MHS–PCLS on several benchmark functions. The performance of MHS–PCLS combined with a modified Deb's constraint handling technique is tested on several constrained engineering design problems in "Constrained engineering design problems" section. A case study is investigated in "Case study: car side impact design" section. Finally, "Conclusions" section offers the concluding remarks.

Previous work

The harmony search algorithm is simple in concept and easy in implementation with only a small number of parameters (Zarei et al. 2009). However, the performance of the basic HS is not satisfactory; hence, many previous studies have been conducted to improve its performance.

Mahdavi et al. (2007) proposed an improved harmony search (IHS) algorithm. IHS applies the same memory consideration, pitch adjustment and random selection as the basic HS algorithm; however, it dynamically updates the values of the parameters such as the pitch adjustment rate (PAR) and the bandwidth (BW).

Inspired by the concept of swarm intelligence as proposed in PSO (Eberhart et al. 1995), Omran and Mahdavi (2008) proposed another variant of the HS algorithm, called the Global-best harmony search (GHS) algorithm. In GHS, the improvised new vector directly adopts the current best pitch from the harmony memory to simplify the pitch adjustment step. Moreover, the parameters in GHS are also dynamically adapted in the same procedure as IHS. However, this variant has a serious deficiency in that it may generate infeasible solutions. The deficiency is induced by ignoring the differences between the decision variables of the different dimensions.

To overcome the deficiency of GHS, Pan et al. (2010) proposed a self-adaptive global best harmony search (SGHS) algorithm. The SGHS algorithm employs a new improvisation scheme. The memory consideration procedure was modified by adding additional adjustment to the original memory consideration procedure in HS. The pitch adjustment rule was also modified by adopting the corresponding value from the current best pitch. The SGHS algorithm does not require a precise setting of specific values for the critical parameters such as the harmony memory consideration rate ($HMCR$), PAR and BW, in accordance with the problem's characteristics and complexity. The $HMCR$ and PAR parameters are dynamically updated to a suitable range by recording their historical values corresponding to generated harmonies entering the $\boldsymbol{HM}$. The value of the BW parameter is decreased with increasing generations by a dynamic method.

Table 1 Information of the considered chaotic maps

Name	Mathematical expression	Range
Logistic map	$x_{n+1} = \mu \cdot x_n(1 - x_n)$, $x_0 \in (0, 1)$ *and* $x_0 \notin \{0, 0.25, 0.5, 0.75, 1.0\}$	(0, 1)
Tent map	$x_{n+1} = \begin{cases} \frac{x_n}{0.7}, x_n < 0.7 \\ (\frac{10}{3}) \cdot x_n(1 - x_n), else \end{cases}$	(0, 1)
Chebyshev map	$x_{n+1} = cos(\varphi cos^{-1} x_n), \varphi > 0, x_n \in [-1, 1]$	$(-1, 1)$
Circle map	$x_{n+1} = x_n + \theta - (\frac{\tau}{2\pi}) sin(2\pi x_n) mod(1), x_n \in (0, 1), \theta = 0.2$ *and* $\tau = 0.5$	(0, 1)
Cubic map	$x_{n+1} = \rho x_n(1 - x_n^2), x_n \in (0, 1), \rho = 2.59$	(0, 1)
Gauss map	$x_{n+1} = \begin{cases} 0, & x_n = 0 \\ \frac{1}{x_n} - [\frac{1}{x_n}], & x_n \neq 0 \end{cases}, x_n \in (0, 1)$	(0, 1)
ICMIC map	$x_{n+1} = sin(\frac{\alpha}{x_n}), \alpha \in (0, 1), x_n \in (-1, 1)$	$(-1, 1)$
Sinusodial map	$x_{n+1} = sin(\pi x_n), x_n \in (0, 1)$	(0, 1)

Enayatifar et al. (2013) proposed a novel HS algorithm based on learning automata called LAHS. In the LAHS algorithm, learning ability is employed in the HS to select the parameters based on spontaneous reactions. To begin, all the parameter values are randomly selected based on a specific probability distribution, e.g., uniform distribution. Then, a reinforcement signal is returned to the parameters after each fitness evaluation. If the fitness value was improved, the signal is a reward signal; otherwise, it is a punish signal. The automaton learning mechanism employs this feedback to update the existing parameter probability distributions. Repeating this action increases the probability distribution of the better parameters and the most favorable parameter values are eventually determined.

In other research works, selection strategies have been introduced into the HS algorithm. Al-Betar et al. (2012) proposed several novel selection schemes to replace the random selection scheme in the memory consideration operation of HS. The novel selection schemes employed the natural selection principle of "survival of the fittest" to generate the new harmony by focusing on the better solutions stored in $\boldsymbol{HM}$. Computational results verified that the novel selection schemes directly influenced the performance of HS. In another research work, Castelli et al. (2014) proposed a geometric selective HS (GSHS). In GSHS, tournament selection is used to choose two parents for generating new harmonies by the newly defined mutation operation.

Other researchers have attempted to improve the performance of HS by modifying the algorithm structure. Related research work such as Al-Betar et al. (2015) proposed an island-based HS (iHS) algorithm. iHS learns from the concept of the island model, which has been successfully applied in other evolutionary algorithms (EAs). According to the island model, iHS divides the HM into several subpopulations (islands) and each island evolves independently for a numbers of iterations. Periodically, the islands interact using a migration process, which is responsible for sending and receiving certain individuals across islands controlled by migration rate and migration frequency.

Hybridization of different metaheuristics is also an efficient method to enhance the performance of the algorithms. It can integrate their advantages and shield their shortcomings. Along with this research line, the HS algorithm was successfully hybridized with PSO and DE to yield improved performance (Zhao et al. 2015; Abedinpourshotorban et al. 2016).

Proposed approach

Chaotic sequence

In meta-heuristic optimization, random sequences with a long period and good uniformity are always required (Schuster and Just 2006). Chaos is the external complexity performance of identifying a nonlinear system with inherent randomness (Wang and Yao 2009). It has the characteristics of long-term unpredictability, non-periodicity, non-convergence and boundedness. Further, it has a sensitive dependence upon its initial condition and parameter (Schuster and Just 2006). Phatak and Rao (1995) proved that a chaotic sequence could be used as a possible approach to obtain random numbers. In this study, eight well-known one-dimensional chaotic maps in Baykasoglu (2012) are considered. The detailed information of these chaotic maps is presented in Table 1.

To investigate the distribution and ergodic properties of these different chaotic maps, we run these maps for 10,000 iterations, and obtain the result for the distribution in the range of (0, 1). The chaotic sequences of the Chebyshev map and the ICMIC map are normalized. The distribution properties of the eight different maps are displayed in Fig. 1, which indicates that the distribution or the ergodic properties of these chaotic maps are different. Previous studies have

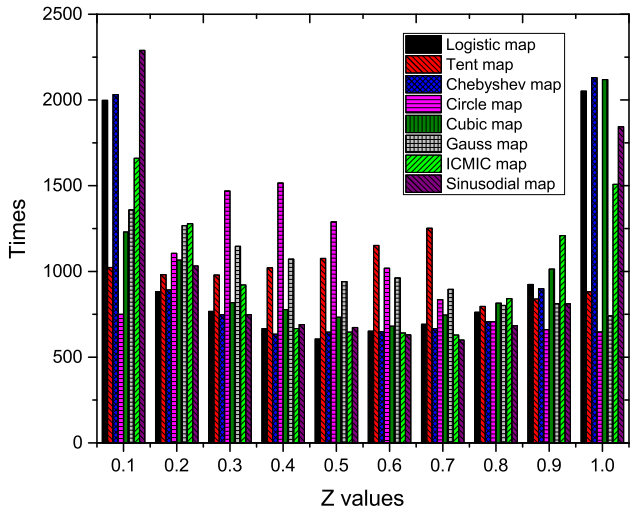


Fig. 1 Distribution property comparisons of eight different chaotic maps

demonstrated that the probability distribution property of different chaotic maps significantly affect the global searching capability and optimization efficiency (Yang et al. 2014; Yuan et al. 2014). Therefore, it is advantageous to employ the chaotic map contributing to a higher convergence rate and accuracy. This part will be discussed in "Performance of MHS–PCLS with different chaotic maps" section.

Basic HS algorithm

In the basic HS algorithm, each solution is called a harmony and is represented by an N-dimension real vector. An initial population of harmony vectors is randomly generated and stored in the harmony memory (HM). Then, a new candidate harmony is improvised from all of the solutions in the HM using a memory consideration rule, a pitch adjustment rule and a random re-initialization. Finally, the HM is updated by comparing the fitness between the new candidate harmony and the worst harmony in the current HM. The worst harmony vector is replaced by the new candidate harmony vector if it is better than the worst harmony vector in the HM. The improvisation and updating process repeat until a predefined termination criterion is reached. The pseudo code of HS is illustrated in Algorithm 1. A more detailed description of HS can be found in (Mahdavi et al. 2007).

Algorithm 1 Pseudo code of the Harmony Search algorithm

Begin HS
Define fitness function $fitness(\boldsymbol{x}) = f(\boldsymbol{x}), \boldsymbol{x} = (x^1, x^2, \ldots, x^N)$
Define the lower and upper boundaries: $\boldsymbol{LB}, \boldsymbol{UB}$
Set algorithm parameters: harmony memory size(HMS),
harmony memory consideration rate($HMCR$),
pitch adjustment rate(PAR) and the bandwidth(BW).
Set maximum number of iterations NI.
$\boldsymbol{HM} \leftarrow$ Generate initial population
Set $t = 0$
while $t < NI$ **do**
  for $j = 1$ to N /*N denotes the number of variables*/ **do**
    if $r_1 < HMCR$ /*r_1, r_2, r_3 and r_4 are uniformly distributed continuous random numbers between [0, 1]*/ **then**
      $x_{new}^j = \boldsymbol{HM}(a, j), a \in \{1, 2, \ldots, HMS\}$ /*Choose a value from $\boldsymbol{HM}$ for j*/
      if $r_2 < PAR$ **then**
        $x_{new}^j = x_{new}^j \pm r_3 \cdot BW$ /*Pitch adjustment */
      else
        $x_{new}^j = LB^j + r_4 \cdot (UB^j - LB^j)$ /*Randomly generate a value */
      end if
    end if
  end for
  if $fitness(\boldsymbol{x}_{new}) \leq worst(fitness(\boldsymbol{HM}))$ **then**
    $\boldsymbol{HM} \leftarrow \boldsymbol{x}_{new}$ /*Update the $\boldsymbol{HM}$ */
  end if
  $t = t + 1$
end while
end

Proposed MHS–PCLS algorithm

HS algorithm with intersect mutation operator

In our previous study, a modified HS variant (MHS) with an intersect mutation operator was proposed to solve the continuous function optimization problems (Yi et al. 2016a). The core idea of MHS is to utilize the intersect mutation operation between the better part and the worse part to maintain the diversity of the harmony memory. The intersect mutation operation was proven to be efficient and is adopted in this study. Figure 2 is the schematic drawing of the intersect mutation operation. To begin, all of the harmony vectors are divided into two parts based on their fitness evaluations. We introduce a constant coefficient $M(0 < M < 1)$, which stands for the proportion of better harmonies in the harmony memory pool. In this research, we set $M = 0.4$. For the better part, we mutate the vectors with one harmony (wr_1) chosen from the worse part and two harmonies (br_1 and br_2) chosen from the better part, as the formula below indicates:

$$x_i = x_{wr_1} + F \cdot (x_{br_1} - x_{br_2}), br_1 \neq br_2 \neq wr_1 \neq i \quad (1)$$

where $F(0 < F < 1)$ is the mutation parameter. F is set to 0.5 in this study.

For the worse part, we mutate the vectors with one harmony (br_1) chosen from the better part and two harmonies (wr_1 and wr_2) chosen from the worse part, as the formula below indicates:

$$x_i = x_{br_1} + F \cdot (x_{wr_1} - x_{wr_2}), wr_1 \neq wr_2 \neq br_1 \neq i \quad (2)$$

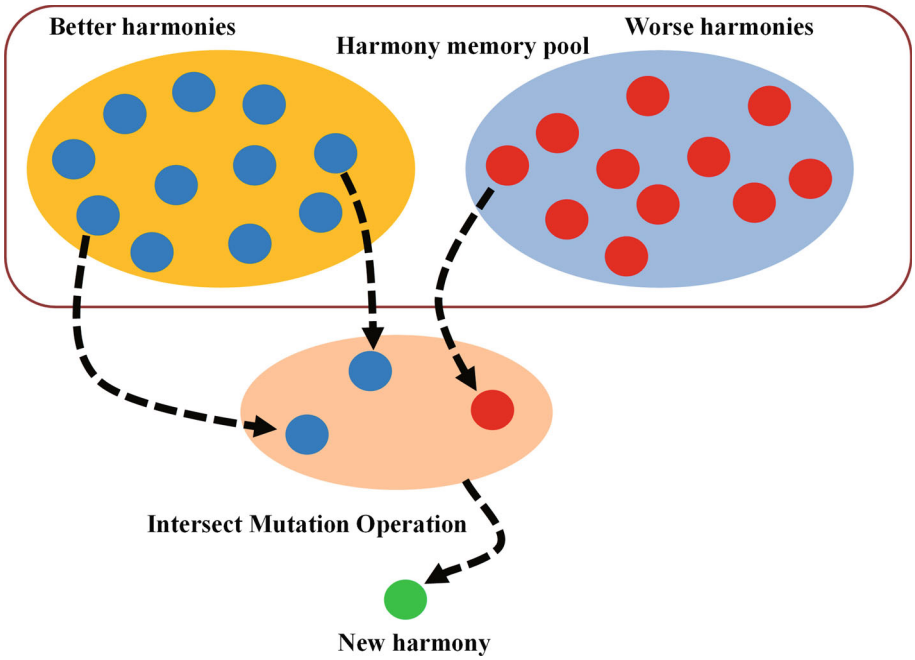


Fig. 2 Schematic drawing of the proposed intersect mutation operation

The detailed pseudo code explaining the various steps of improvising new harmonies is exhibited in Algorithm 2 for easy implementation and understanding of the proposed approach.

Algorithm 2 Pseudo code of improvising new harmonies

Begin
for $i = 1 : HMS$ **do**
  for $j = 1 : N$ **do**
    if $r_1 < HMCR$ **then**
      $x_i^j = \boldsymbol{HM}(a, j), a \in \{1, 2, \ldots, HMS\}$ /*Randomly choose a value from $\boldsymbol{HM}$ */
      if $r_2 < PAR$ **then**
        if $i < M * HMS$ **then**
          $x_i^j = x_{wr_1}^j + F \cdot (x_{br_1}^j - x_{br_2}^j), wr_1 \neq br_1 \neq br_2 \neq i$ /*Intersect mutation operation 1 */
        else
          $x_i^j = x_{br_1}^j + F \cdot (x_{wr_1}^j - x_{wr_2}^j), br_1 \neq wr_1 \neq wr_2 \neq i$ /*Intersect mutation operation 2 */
        end if
      end if
    else
      $x_i^j = LB^j + r_3 \cdot (UB^j - LB^j)$ /*Randomly generate a variable */
    end if
  end for
end for
End

Parallel chaotic local search

As mentioned in "Chaotic sequence" section, chaos is sensitive to its initial conditions, which leads to an algorithm with low robustness. To increase the robustness of the algorithm, a parallel chaotic local search (PCLS) is proposed in this paper. The PCLS method searches from several different initial points and hence can diminish the sensitivity of the initial conditions.

For an optimization problem with decision variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, N represents the number of decision variables. In the proposed method, each decision variable is mapped by m chaotic maps simultaneously. In the PCLS, a $m \times N$ matrix $\boldsymbol{CMS}$ of chaotic maps is generated by,

$$\boldsymbol{CMS}_i = \begin{bmatrix} \delta_{11}^i & \delta_{12}^i & \cdots & \delta_{1N}^i \\ \delta_{21}^i & \delta_{22}^i & \cdots & \delta_{2N}^i \\ \vdots & \cdots & \cdots & \vdots \\ \delta_{m1}^i & \delta_{m2}^i & \cdots & \delta_{mN}^i \end{bmatrix}_{m \times N} \tag{3}$$

where δ_{jk}^i is a random number in the range of (0, 1) generated by the chaotic maps. i is the ith generation, j is the jth chaotic map and k is the kth variable. δ_{jk}^i is updated in every generation by the chaotic maps mentioned in "Chaotic sequence" section.

The candidate solutions in the PCLS method are generated by the following formulas:

$$\boldsymbol{CM}_i = \lambda \boldsymbol{CB}_i + (1 - \lambda)\boldsymbol{RS}_i \tag{4}$$

$$\boldsymbol{CB}_i = \begin{bmatrix} \boldsymbol{X}_i \\ \boldsymbol{X}_i \\ \vdots \\ \boldsymbol{X}_i \end{bmatrix}_{m \times N} \tag{5}$$

$$\boldsymbol{X}_i = \left[x_i^1 \; x_i^2 \; \ldots, x_i^N \right] \tag{6}$$

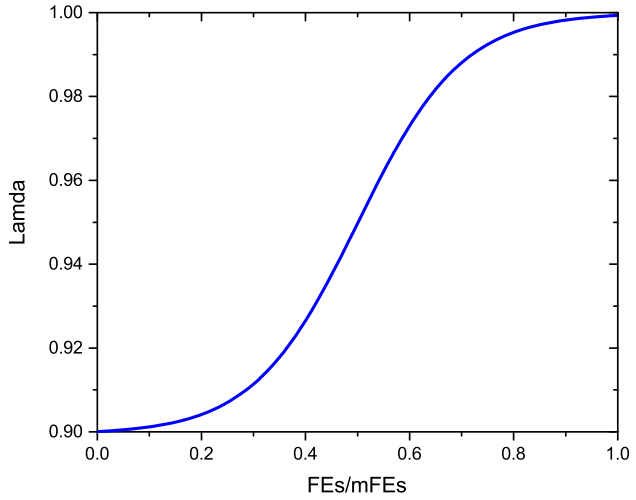


Fig. 3 Trend of λ with increasing FEs

where $\boldsymbol{CM}_i$ is the candidates matrix of PCLS in the ith generation, $\boldsymbol{CB}_i$ is the matrix consisting of m individuals $\boldsymbol{X}_i$, and $\boldsymbol{RS}_i$ is generated by the equation $\boldsymbol{RS}_i = \boldsymbol{LB} + \boldsymbol{CMS}_i \cdot (\boldsymbol{UB} - \boldsymbol{LB})$, where $[LB, UB]$ is the search space of individual $\boldsymbol{X}_i$. λ is the weighting factor.

Equation (4) is based on the geometric crossover operation (Moraglio et al. 2013), where the offspring $\boldsymbol{CM}$ is the linear combination of the two parents $\boldsymbol{CB}$ and $\boldsymbol{RS}$. The generated offspring always stand between the segments delimited by the two parents and the greater the value of λ, the closer the offspring is to the current best. A local search is performed to ensure that the offspring will be produced around the current best; hence, a greater value of λ is preferred. Moreover, in case the local search has been trapped into local optima, a dynamic adjustment strategy on λ according to the logistic curve (Pearl and Reed 1920) is adopted in this paper:

$$\lambda = 0.9 + 0.1 \cdot \frac{1 + e^{-5}}{1 + e^{10\left(0.5 - \frac{FEs}{mFEs}\right)}} \tag{7}$$

where FEs is the current number of function evaluations, and $mFEs$ is the maximum number of function evaluations. Figure 3 presents the trend of λ with increasing FEs and assumes the shape of "S". Because λ continues to increase during the entire optimization process, the step length of PCLS will be decreased; hence, PCLS will enhance the exploration abilities at the beginning phase of the optimization and refine the solutions in the later phase.

Although the PCLS method can enhance the search ability of the MHS algorithm, it also increases the computational cost. To avoid premature convergence and save FEs, PCLS terminates when a superior fitness is obtained. The procedure for the method is presented as follows and the flowchart is provided in Fig. 4.

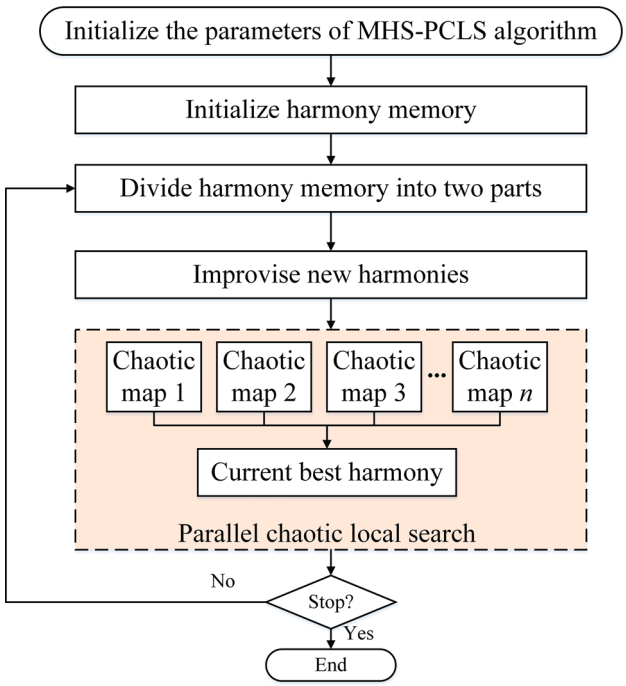


Fig. 4 Flowchart of the proposed MHS–PCLS

Step 1 Initialization

1.1. Set parameters HMS, $HMCR$, PAR, and the max iteration number (NI), proportion of the better part in harmony memories $M(0 < M < 1)$ and mutation parameter F.
1.2. Initialize all harmonies in $\boldsymbol{HM}$ within the search space and evaluate their fitness.
1.3. Divide all the harmonies into two parts based on their fitness.

Step 2 Iteration

2.1. Improvise new harmonies with intersect mutation as indicated in Algorithm 2.
2.2. Divide all the harmonies into two parts again and identify the current best harmony in the harmony memory pool.

Step 3 Apply PCLS on the current best harmony.
Step 4 Check the termination criterion. If the termination criterion is met, output the best solution. Otherwise, return to Step 2.

Numerical simulations and analysis

Experimental setup

In this section, the performance of the proposed MHS–PCLS algorithm is compared with the original HS algorithm and

Table 2 Test functions

Name	Function expression	Range	Global optimum
Sphere	$f_1(x) = \sum_{i=1}^{N} x_i^2$	$[-100, 100]^N$	0
Schwefel's problem 2.22	$f_2(x) = \sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	$[-10, 10]^N$	0
Rosenbrock	$f_3(x) = \sum_{i=1}^{N-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30, 30]^N$	0
Step	$f_4(x) = \sum_{i=1}^{N} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^N$	0
Rotated hyper-ellipsoid	$f_5(x) = \sum_{i=1}^{N} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^N$	0
Schwefel's problem 2.26	$f_6(x) = 418.9829 \cdot N - \sum_{i=1}^{N} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	$[-500, 500]^N$	0
Rastrigin	$f_7(x) = \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^N$	0
Ackley	$f_8(x) = 20 + e - 20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{N} x_i^2}\right) - exp\left(\frac{1}{n}\cos(2\pi x_i)\right)$	$[-32, 32]^N$	0
Griewank	$f_9(x) = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]^N$	0

its prominent variants, including HS, IHS, GHS, SGHS and LAHS. All of the parameter settings were the same as in (Enayatifar et al. 2013). Because MHS–PCLS contains a local search operation, it increased the number of FEs (FEs indicates the number of function evaluations). To obtain an unbiased comparison, the termination criterion was set to 50,000 FEs for all compared algorithms. Several well-studied benchmark problems were used as test functions; these are indicated in Table 2. For all of the benchmark functions, the dimensions N was set to 30. Based on our empirical experience (Yi et al. 2016a), the parameters of MHS–PCLS were recommended as follows: the size of harmony memory $HMS = 100$, $HMCR = 0.995$, $PAR = 0.90$, $M = 0.4$, and $F = 0.5$.

A proper search length for the local search is important for PCLS. A small length may be inefficient in exploring the best solution and may therefore be unsuccessful at improving the search quality. Conversely, with a longer length, the PCLS may consume additional function evaluations unnecessarily. Based on the recommendations in the literature (Jia et al. 2011), the maximum search length of PCLS in each iteration was set to $S_l = N/5$.

The number of parallel chaotic maps is also important for PCLS. More parallel chaotic maps can increase the robustness of the algorithm; however, they may consume additional function evaluations unnecessarily. As a compromise between the above two factors, the number of parallel chaotic maps was to set to $N_p = 5$.

Performance of MHS–PCLS with different chaotic maps

As discussed in "Chaotic sequence" section, the search capability with different chaotic maps can differ in view of the convergence rate and accuracy. In this section, the performance of MHS–PCLS with different chaotic maps was compared to identify the best option. To perform a fair comparison, all parameter settings of the MHS–PCLS remained the same. Each benchmark function was tested for 100 runs and the maximum number of function evaluations $mFEs$ was set to 10,000. The performance of MHS–PCLS with different chaotic maps was ranked based on their mean value of the 100 runs; the results are presented in Table 3. The performance of MHS–PCLS with an ICMIC map provided the best performance, which means an ICMIC map has the potential to generate superior solutions. Hence, the ICMIC map is selected for the experiments.

Comparison of the optimization results and computation time among different methods

Table 4 displays the results of all test functions obtained by HS, IHS, GHS, SGHS, LAHS and the proposed MHS–PCLS. Each benchmark problem performed 30 independent replications for all of the compared algorithms. In Table 4, AE and SD are the average value over 30 runs and the corresponding standard deviations, respectively. MHS–PCLS generated the best results for seven out of nine functions. For function f_4, both SGHS and MHS–PCLS obtained the global optimum. For functions f_6 and f_7, the results obtained by MHS–PCLS were inferior to SGHS. To demonstrate the evolution process of the compared algorithms, the convergence curves of the competitive algorithms are given in Fig. 5. It can be observed from Fig. 5 that for the majority of the functions, MHS–PCLS converged marginally more slowly than the other algorithms during the initial stage of the evolution process; however it always overtaken during the later iterations. The average computation times of the compared methods are presented in Table 5. It can be seen in Table 5 that although MHS–PCLS expends extra computation cost on the local search, the total computation times have no significant difference compared

Table 3 Ranking of the performance of MHS–PCLS with different chaotic maps

Chaotic maps	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	Average rank
Logistic	4	8	7	1	7	7	4	3	2	4.78(4)
Tent	3	3	5	3	8	6	6	7	3	4.89(7)
Chebyshev	2	2	2	8	4	3	8	8	6	4.78(4)
Circle	6	6	4	4	3	2	2	2	1	3.33(2)
Cubic	5	1	8	7	2	1	7	5	5	4.56(3)
Gauss	8	5	3	2	5	4	3	6	7	4.78(4)
ICMIC	1	4	1	5	1	5	5	1	4	3(1)
Sinusodial	7	7	6	6	6	8	1	4	8	5.89(8)

Table 4 Mean and standard deviation ($\pm SD$) of the benchmark functions optimization results ($N = 30$)

Functions	HS	IHS	GHS	SGHS	LAHS	MHS–PCLS
Sphere						
AE	5.3069e+00	1.0468e+01	2.6845e+00	1.7795e−08	5.9674e−04	**5.4654e−14**
SD	7.4936e−01	7.6135e+00	2.1878e+00	5.1326e−09	5.0773e−04	**6.2150e−14**
Schwefel's2.22						
AE	8.1786e−01	3.1314e−01	1.9975e−01	4.1256e−04	9.9346e−02	**2.8306e−08**
SD	7.3159e−02	2.2349e−01	1.5791e−01	8.0259e−05	7.7078e−02	**1.0236e−08**
Rosenbrock						
AE	1.4510e+03	2.7902e+04	3.5423e+03	1.9681e+02	5.1163e+02	**2.5882e+01**
SD	5.3936e+02	1.0400e+05	1.1570e+03	5.1644e+02	1.1242e+03	**3.7270e−01**
Step						
AE	4.5333e+00	3.4466e+01	5.2333e+00	**0.0000e+00**	2.3600e+01	**0.0000e+00**
SD	1.4772e+00	1.8396e+01	4.0470e+00	**0.0000e+00**	9.3152e+00	**0.0000e+00**
Rotated hyper-ellipsoid						
AE	4.8224e+02	2.1861e+03	8.5693e+02	1.0278e+01	1.3843e+03	**4.9226e+00**
SD	2.8087e+02	1.2219e+03	5.3388e+02	5.3209e+00	1.1275e+03	**2.5088e+00**
Schwefel's2.26						
AE	1.7123e+01	2.5574e+01	1.1254e+01	**2.5526e−03**	3.6695e+01	1.1844e+01
SD	3.6494e+00	1.2954e+01	7.2627e+00	**3.6296e−03**	2.1410e+01	2.4275e+01
Rastrigin						
AE	1.8478e+01	5.2389e+00	2.9195e+00	**2.5526e−01**	7.8715e+00	6.3579e+00
SD	2.6731e+00	1.9833e+00	1.7135e+00	**4.2642e−01**	2.0172e+00	4.0371e+00
Ackley						
AE	1.1678e+00	1.3712e+00	7.1344e−01	8.0726e−05	1.4502e+00	**1.6906e−07**
SD	1.8113e−01	3.4720e−01	4.1747e−01	1.3596e−05	3.0848e−01	**7.8367e−08**
Griewank						
AE	1.0480e+00	1.1249e+00	8.8290e−01	7.7119e−02	9.7241e−01	**9.2931e−13**
SD	6.9077e−03	1.3627e−01	1.9029e−01	4.6151e−02	2.2036e−01	**6.6629e−13**

Bold values indicate the best results

to the other methods. The possible reason is that the number of FEs for all methods are the same.

Wilcoxon's sum rank test

To judge whether the results obtained with the MHS–PCLS algorithm differ from the results of the other algorithms in a statistically significant manner, a nonparametric statistical test called Wilcoxon's rank sum test for independent samples was conducted. The significance level was set to 5 %. The P-values for the compared HS variants over all nine benchmark functions are provided in Table 6. In Table 6, the P-values in each row are calculated through the rank sum between the algorithm that obtained the best solutions and the remaining algorithms; hence, NA appears when the algorithms obtained the best solutions for the corresponding

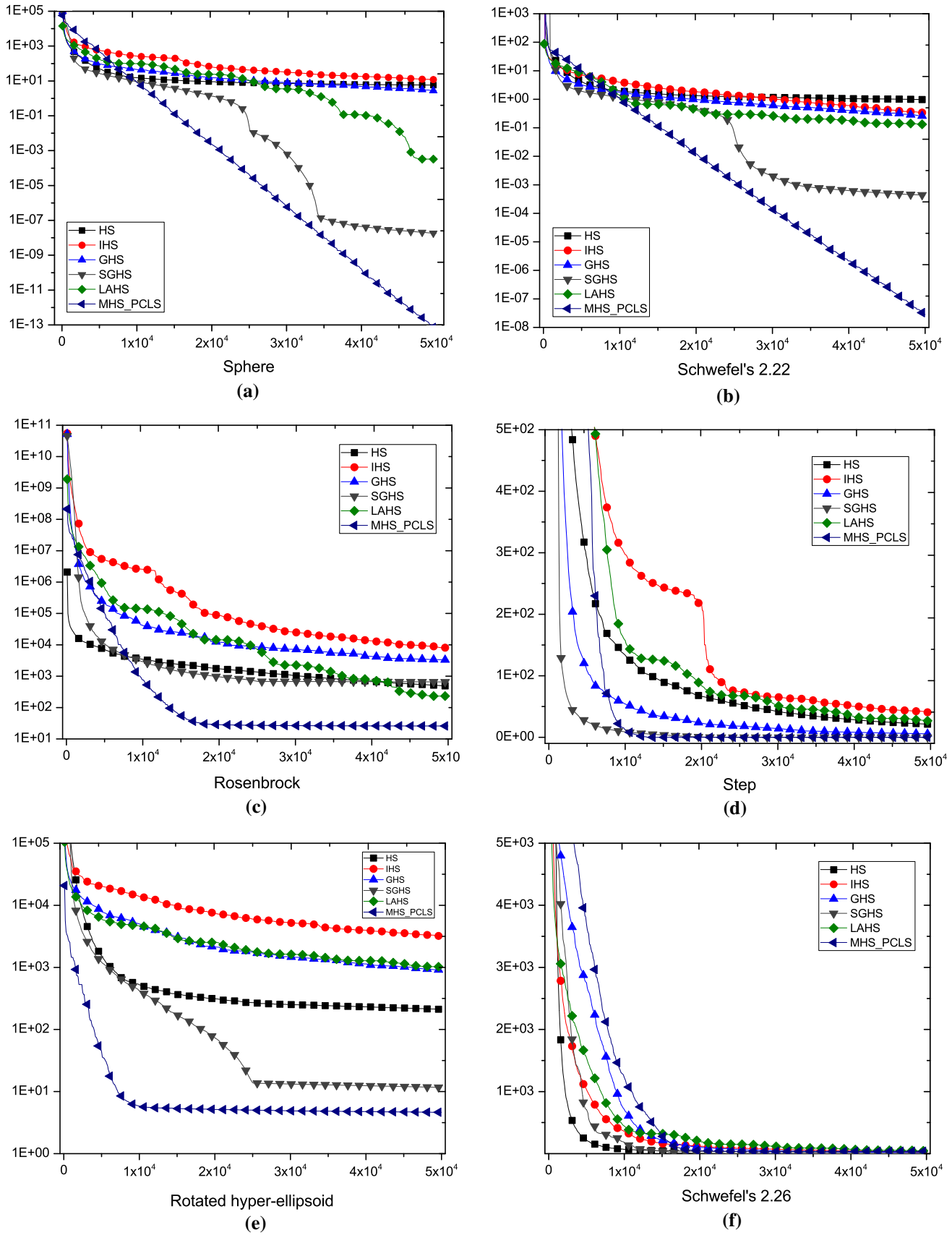


Fig. 5 Convergence graphs of the six compared algorithms on the nine benchmark functions. ($f_1 - f_9$)

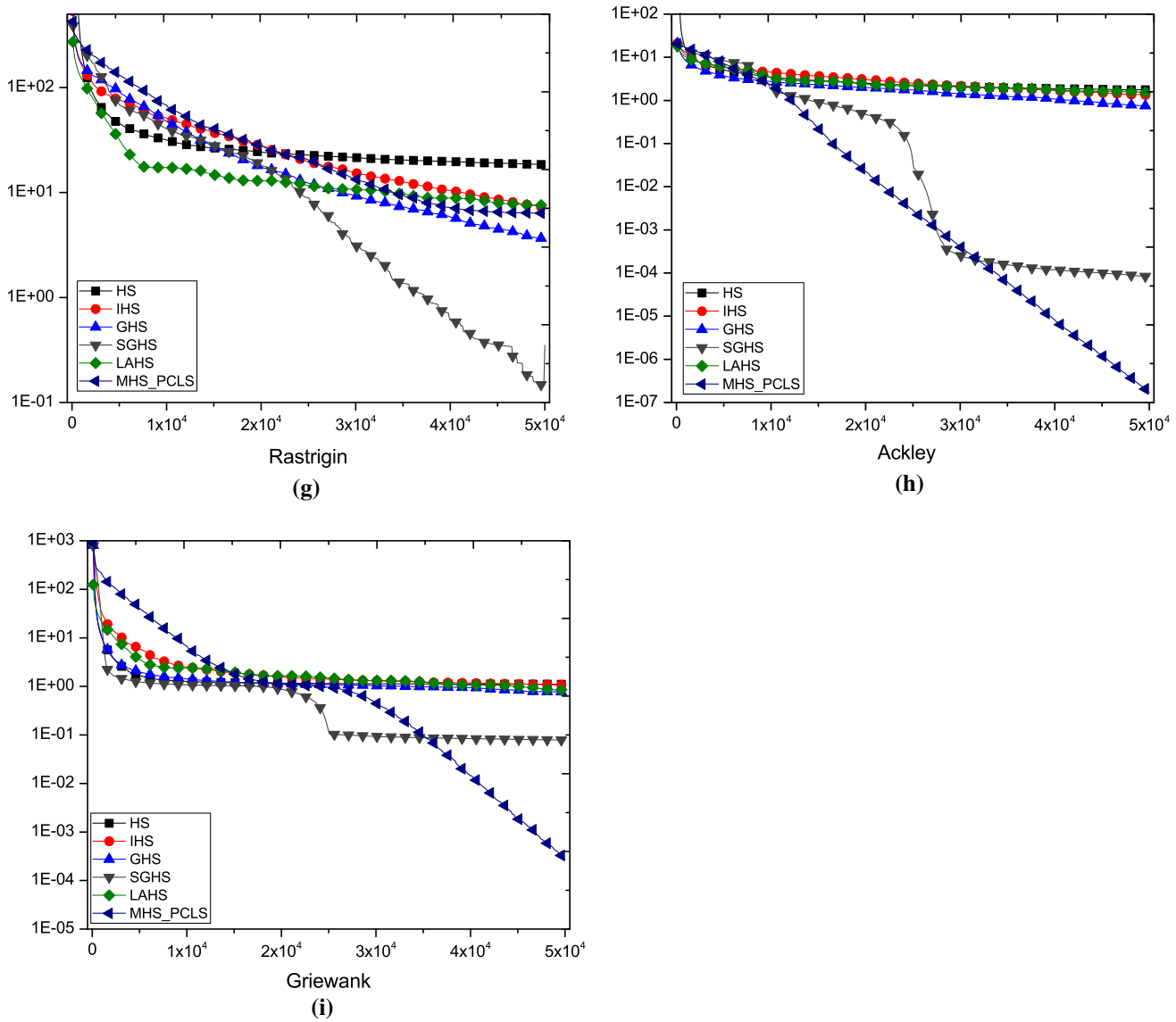


Fig. 5 continued

Table 5 The mean computation time of different methods (in s)

Functions	HS	IHS	GHS	SGHS	LAHS	MHS–PCLS
Sphere	0.996	1.166	0.945	1.190	1.159	1.207
Schwefel's2.22	1.023	1.258	0.998	1.381	1.268	1.179
Rosenbrock	1.346	1.697	1.320	1.583	1.670	1.187
Step	1.019	1.589	0.960	1.263	1.318	1.185
Rotated hyper-ellipsoid	1.121	1.666	1.029	1.286	1.409	1.315
Schwefel's2.26	1.099	1.614	1.047	1.302	2.190	1.306
Rastrigin	1.041	1.217	0.973	1.242	1.740	1.254
Ackley	0.984	1.219	0.983	1.252	1.809	1.302
Griewank	1.018	1.258	1.003	1.266	2.266	1.321

Table 6 P-values calculated for Wilcoxon's Rank Sum Test for all of the benchmark problems

P-values	HS	IHS	GHS	SGHS	LAHS	MHS–PCLS
f_1	3.01797e−11	3.01986e−11	5.57265e−10	3.01986e−11	3.01986e−11	NA
f_2	3.00287e−11	3.01986e−11	3.01986e−11	3.01986e−11	3.01986e−11	NA
f_3	3.01986e−11	3.01986e−11	3.01986e−11	4.9426e−5	3.01986e−11	NA
f_4	1.0239e−12	1.20192e−12	5.57275e−11	NA	1.10099e−12	NA
f_5	3.01986e−11	3.01986e−11	3.01986e−11	1.47332e−7	3.01986e−11	NA
f_6	3.01986e−11	3.01986e−11	3.01986e−11	NA	3.01986e−11	1.01761e−05
f_7	2.98034e−11	2.98034e−11	2.83766e−10	NA	2.98034e−11	2.98034e−11
f_8	3.01986e−11	3.01986e−11	3.01986e−11	3.01986e−11	3.01986e−11	NA
f_9	3.01986e−11	3.01986e−11	3.01986e−11	3.01986e−11	3.01986e−11	NA

Table 7 The effect of HMS on the mean and standard deviation ($\pm SD$) of the benchmark functions optimization results

Functions	$HMS = 10$	$HMS = 30$	$HMS = 50$	$HMS = 100$
Sphere				
AE	3.2874e−10	1.6606e−10	3.9858e−12	**5.4654e−14**
SD	2.7470e−10	9.8355e−11	1.7388e−12	**6.2150e−14**
Schwefel's2.22				
AE	5.8130e−06	5.5731e−06	7.3681e−07	**2.8306e−08**
SD	1.3821e−06	6.8044e−07	2.4299e−07	**1.0236e−08**
Rosenbrock				
AE	2.5879e+01	2.5062e+01	**2.4279e+01**	2.5882e+01
SD	7.0459e−01	4.6574e−01	**8.5616e−01**	3.7270e−01
Step				
AE	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
SD	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
Rotated hyper-ellipsoid				
AE	5.6082e+00	5.4184e+00	5.2482e+00	**4.9226e+00**
SD	8.0463e−01	1.0505e+00	2.2993e+00	**2.5088e+00**
Schwefel's2.26				
AE	3.4860e+01	6.8675e+00	**5.1823e+00**	1.1844e+01
SD	4.4215e+01	5.5398e+00	**1.0182e+00**	2.4275e+01
Rastrigin				
AE	1.1584e+01	8.2604e+00	**5.9857e+00**	6.3579e+00
SD	3.5925e+00	1.5998e+00	**3.7882e+00**	4.0371e+00
Ackley				
AE	3.3428e−06	4.3931e−06	7.1521e−07	**1.6906e−07**
SD	1.1532e−06	1.0233e−06	2.6728e−07	**7.8367e−08**
Griewank				
AE	4.5470e−10	6.4505e−10	1.3215e−11	**9.2931e−13**
SD	1.7224e−10	4.0088e−10	5.4922e−12	**6.6629e−13**

Bold values indicate the best results

benchmark function. The differences between the algorithms that obtained the best solutions and the other algorithms and are considered as significant if the P-values are less than 0.05. It can be observed from Table 6 that MHS–PCLS achieved statistically superior performance compared to all other HS variants for six out of nine benchmark functions.

Effects of changing the parameters on the performance of the MHS–PCLS

In this subsection, the effect of changing HMS, $HMCR$, PAR, N_p and S_l on the performance of MHS–PCLS is investigated. Tables 7, 8, 9, 10 and 11 presents the effects of these parameters on the mean and standard deviation ($\pm SD$) of the benchmark function optimization results with 30 indepen-

Table 8 The effect of $HMCR$ on the mean and standard deviation ($\pm SD$) of the benchmark functions optimization results

Functions	$HMCR = 0.8$	$HMCR = 0.9$	$HMCR = 0.99$	$HMCR = 0.995$
Sphere				
AE	9.8543e+03	2.3980e+01	1.0076e−12	**5.4654e−14**
SD	5.4926e+02	3.2123e+01	4.5427e−13	**6.2150e−14**
Schwefel's2.22				
AE	4.0852e+01	1.8232e+00	7.7842e−07	**2.8306e−08**
SD	6.5575e+00	1.4830e+00	2.9447e−07	**1.0236e−08**
Rosenbrock				
AE	5.2437e+06	2.3805e+04	**2.5050e+01**	2.5882e+01
SD	3.9355e+05	1.6231e+04	**4.9070e−01**	3.7270e−01
Step				
AE	9.1123e+03	3.1667e+01	**0.0000e+00**	**0.0000e+00**
SD	5.2181e+02	2.7813e+01	**0.0000e+00**	**0.0000e+00**
Rotated hyper-ellipsoid				
AE	8.7276e+02	3.6876e+02	6.6023e+00	**4.9226e+00**
SD	1.1953e+02	2.2313e+02	1.3591e+00	**2.5088e+00**
Schwefel's2.26				
AE	6.9624e+03	5.4438e+03	4.7375e+01	**1.1844e+01**
SD	2.3512e+02	5.2434e+02	7.8563e+01	**2.4275e+01**
Rastrigin				
AE	2.2770e+02	1.8915e+02	9.8764e+00	**6.3579e+00**
SD	1.4419e+01	8.1905e+00	4.1369e+00	**4.0371e+00**
Ackley				
AE	1.4833e+01	2.1449e+00	3.8469e−07	**1.6906e−07**
SD	5.1367e−01	1.0296e+00	1.8205e−07	**7.8367e−08**
Griewank				
AE	6.0339e+01	1.1837e+00	9.8573e−04	**9.2931e−13**
SD	7.4048e+00	4.2234e−01	2.9572e−03	**6.6629e−13**

Bold values indicate the best results

dent runs. It can be observed in Table 7 that the performance of MHS–PCLS with a smaller HMS is inferior to that with a larger HMS. This is because MHS–PCLS with a smaller HMS has poor solutions diversity and can be easily trapped into local minima. Thus, large value for HMS are suggested. As indicated in Table 8, the performance difference of MHS–PCLS with different values of $HMCR$ is significant, and the greater the $HMCR$, the greater the performance. Therefore, a large value of $HMCR$ is recommended. Table 9 indicates that there is no single choice for PAR, however, it seems that using a relatively large value (such as PAR = 0.5, 0.7 and 0.9) improves the performance of MHS–PCLS. As illustrated in Tables 10 and 11, N_p and S_l have similar effects on the performance of MHS–PCLS. It seems that moderate values of N_p and S_l are more suitable.

Constrained engineering design problems

In this section, the proposed MHS–PCLS algorithm is applied to solve several constrained engineering problems. These well-known examples have been previously solved based on a variety of optimization techniques; hence, they can be compared with the proposed approach and facilitate validating the effectiveness and efficiency of this approach. The parameter settings of MHS–PCLS are the same as in "Numerical simulations and analysis" section.

Modified Deb's heuristic constrained handling method

Before applying MHS–PCLS algorithm to solve constrained engineering problems, the constraint handling method must first be determined first. In general, constrained optimization problems with m variables and n constraints can be stated as follows:

$$Minimize: \quad f(\boldsymbol{x}), \tag{8}$$

$$s.t. \begin{cases} g_j(\boldsymbol{x}) \le 0, j = 1, 2 \ldots, q \\ h_j(\boldsymbol{x}) = 0, j = q+1, q+2 \ldots, n \\ LB_i \le x_i \le UB_i, i = 1, 2, \ldots, m \end{cases} \tag{9}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ is the solution vector, $f(\boldsymbol{x})$ is the objective function, and $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are the inequality

Table 9 The effect of PAR on the mean and standard deviation ($\pm SD$) of the benchmark functions optimization results

Functions	$PAR = 0.3$	$PAR = 0.5$	$PAR = 0.7$	$PAR = 0.9$
Sphere				
AE	2.6711e−01	**1.2648e−28**	2.6085e−20	5.4654e−14
SD	7.5112e−01	**8.5432e−29**	2.1217e−20	6.2150e−14
Schwefel's2.22				
AE	3.6107e−03	5.5647e−05	**5.4847e−13**	2.8306e−08
SD	6.7126e−03	1.6694e−04	**2.2936e−13**	1.0236e−08
Rosenbrock				
AE	1.3792e+02	8.3970e+01	4.3455e+01	**2.5882e+01**
SD	1.4120e+02	2.4346e+01	2.6051e+01	**3.7270e−01**
Step				
AE	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
SD	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
Rotated hyper-ellipsoid				
AE	6.5971e+00	4.9902e+00	5.0320e+00	**4.9226e+00**
SD	2.4016e+00	2.6189e+00	2.6154e+00	**2.5088e+00**
Schwefel's2.26				
AE	2.6332e+02	2.3606e+02	1.4117e+02	**1.1844e+01**
SD	1.6245e+02	9.3257e+01	7.4929e+01	**2.4275e+01**
Rastrigin				
AE	2.2418e+01	2.0286e+01	1.4847e+01	**6.3579e+00**
SD	6.1212e+00	5.0753e+00	3.5205e+00	**4.0371e+00**
Ackley				
AE	4.4757e−01	**4.0856e−15**	3.2138e−11	1.6906e−07
SD	3.4602e−01	**1.7405e−15**	8.4277e−12	7.8367e−08
Griewank				
AE	2.5916e−03	7.3960e−04	4.3565e−06	**9.2931e−13**
SD	5.9860e−03	2.2188e−03	3.5425e−06	**6.6629e−13**

Bold values indicate the best results

and equality constraints, respectively. The values of LB_i and UB_i are the lower and upper bounds of x_i, respectively.

There is a variety of constraint-handling techniques for constrained optimization problems (Mezura-Montes and Coello 2011; Kramer 2010). The marjority of these are based on the penalty function method because of its simplicity. However, even though the penalty function method is simple and competitive in some numerical optimization problems, defining the appropriate penalty parameters is significant challenge, and it influences the feasible and infeasible solutions severely. Deb (2000) proposed a parameter-less penalty strategy. The primary idea of Deb's method is to distinguish infeasible and feasible solutions and to select a feasible solution or the relatively best infeasible solution. However, the main drawback of this method is that it is prone to cause premature convergence, probably owing to its strong preference for feasible solutions. Mohamed and Sabry (2012) thus proposed a modified Deb's method to address this problem. According to the new comparison rules, new improvised harmonies can replace the harmonies stored in the $\boldsymbol{HM}$ if any of the following rules are true:

1. The new improvised harmony is feasible and the corresponding harmony in $\boldsymbol{HM}$ is infeasible.
2. The new improvised harmony and the corresponding harmony in $\boldsymbol{HM}$ are both feasible; however, the new improvised harmony has a smaller or equal fitness value compare to the corresponding harmony in $\boldsymbol{HM}$.
3. The new improvised harmony and the corresponding harmony in $\boldsymbol{HM}$ are both infeasible; however, the new improvised harmony has a smaller or equal overall constraint violation to the corresponding harmony in $\boldsymbol{HM}$.

The overall constraint violation is calculated by the following steps: first, a tolerance ε is allowed for equality constraints; the constraint violation of a decision vector or an individual $\boldsymbol{x}$ on the jth constraint is calculated by

$$cv_j(\boldsymbol{x}) = \begin{cases} max(0, g_j(\boldsymbol{x})), \ j = 1, 2, \ldots, q \\ max(0, |h_j(\boldsymbol{x})| - \varepsilon), \ j = q+1, q+2, \ldots, n \end{cases} \tag{10}$$

Then, the overall violation of n constraints can be calculated by:

Table 10 The effect of N_p on the mean and standard deviation ($\pm SD$) of the benchmark functions optimization results

Functions	$N_p = 2$	$N_p = 5$	$N_p = 10$	$N_p = 20$
Sphere				
AE	1.1272$e-$08	**5.4654$e-$14**	3.7705$e-$13	8.7785$e-$11
SD	6.0015$e-$09	**6.2150$e-$14**	1.7537$e-$13	5.1068$e-$11
Schwefel's2.22				
AE	4.5824$e-$05	**2.8306$e-$08**	2.2555$e-$07	3.8521$e-$06
SD	1.5918$e-$05	**1.0236$e-$08**	9.8111$e-$08	1.3540$e-$06
Rosenbrock				
AE	2.6173e+01	2.5882e+01	**2.4697e+01**	2.4935e+01
SD	9.8704$e-$01	3.7270$e-$01	**1.0583e+00**	1.6318e+00
Step				
AE	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
SD	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
Rotated hyper-ellipsoid				
AE	**6.4372e+00**	4.9226e+00	3.0013e+00	1.0124e+01
SD	**2.1531e+00**	2.5088e+00	1.1954e+00	5.3263e+00
Schwefel's2.26				
AE	7.1063e+01	**1.1844e+01**	1.3805e+01	4.0673e+01
SD	5.8023e+01	**2.4275e+01**	2.7610e+01	5.1757e+01
Rastrigin				
AE	8.8811e+00	**6.3579e+00**	7.8926e+00	1.2078e+01
SD	1.0540e+00	**4.0371e+00**	4.3271e+00	6.4664e+00
Ackley				
AE	3.2621$e-$05	1.6906$e-$07	**1.9604$e-$07**	2.4986$e-$06
SD	1.0191$e-$05	7.8367$e-$08	**2.7534$e-$08**	8.0640$e-$07
Griewank				
AE	1.7325$e-$08	**9.2931$e-$13**	1.0932$e-$12	1.5296$e-$10
SD	1.1555$e-$08	**6.6629$e-$13**	5.0678$e-$13	1.1771$e-$10

Bold values indicate the best results

$$cv(\boldsymbol{x}) = \sum_{j=1}^{n} cv_j(\boldsymbol{x}) \tag{11}$$

This simple modification can reduce the probability of stagnation and help the algorithm to spread out and search through the entire solution space. So this method is adopted in MHS–PCLS to handle the constraints. To validate the performance of the combined method for solving constrained problems, several engineering design problems were tested and the results are shown in the next sections.

Tension/compression string design problem

The tension/compression string design problem was first introduced by Arora (2004), as illustrated in Fig. 6. In this problem, the weight of a string is subject to constraints on minimum deflection, shear stress, surge frequency, and limits on the outside diameter. The design variables are: the wire diameter $d(x_1)$, the mean coil diameter $D(x_2)$ and the number of active coils $P(x_3)$. The mathematical model of this problem can be found in "Appendix".

This problem has been previously solved by GA through the use of dominance-based tournament selection (DGA) proposed by Coello and Montes (2002), an effective co-evolutionary particle swarm optimization approach (CPSO) proposed by He and Wang (2007a), hybrid particle swarm optimization (HPSO) proposed by He and Wang (2007b), co-evolutionary differential evolution (CDE) proposed by Huang et al. (2007), improved harmony search (IHS) proposed by Mahdavi et al. (2007),hybrid Nelder-Mead simplex search and particle swarm optimization (NM-PSO) proposed by Zahara and Kao (2009), another improved harmony search (IPHS) variant proposed by Jaberipour and Khorram (2010), teaching and learning based optimization (TLBO) proposed by Rao et al. (2011), articial bee colony (ABC) algorithm proposed by Akay and Karaboga (2012), upgraded articial bee colony (UABC) algorithm proposed by Brajevic and Tuba (2013), mine blast algorithm (MBA) proposed by Sadollah et al. (2013), hybrid cuckoo search algorithm based on Solis and Wets local search technique that relies on an augmented

Table 11 The effect of S_l on the mean and standard deviation ($\pm SD$) of the benchmark functions optimization results

Functions	$S_l = 2$	$S_l = 6$	$S_l = 10$	$S_l = 20$
Sphere				
AE	3.4108$e-10$	**5.4654$e-14$**	4.1557$e-11$	5.5604$e-10$
SD	2.0180$e-10$	**6.2150$e-14$**	4.2439$e-11$	2.2712$e-10$
Schwefel's				
AE	7.8994$e-06$	**2.8306$e-08$**	1.5502$e-06$	1.3837$e-05$
SD	4.2182$e-06$	**1.0236$e-08$**	3.7182$e-07$	5.3213$e-06$
Rosenbrock				
AE	2.5589$e+01$	2.5882$e+01$	2.5274$e+01$	**2.5092$e+01$**
SD	1.1031$e+00$	3.7270$e-01$	9.1516$e-01$	**3.9348$e-01$**
Step				
AE	**0.0000$e+00$**	**0.0000$e+00$**	**0.0000$e+00$**	**0.0000$e+00$**
SD	**0.0000$e+00$**	**0.0000$e+00$**	**0.0000$e+00$**	**0.0000$e+00$**
Rotated hyper-ellipsoid				
AE	5.6852$e+00$	**4.9226$e+00$**	6.3221$e+00$	5.4973$e+00$
SD	3.3562$e+00$	**2.5088$e+00$**	9.5187$e-01$	1.6518$e+00$
Schwefel's2.26				
AE	2.3689$e+01$	**1.1844$e+01$**	5.1917$e+01$	5.5950$e+01$
SD	4.7375$e+01$	**2.4275$e+01$**	5.4944$e+01$	5.3369$e+01$
Rastrigin				
AE	1.0791$e+01$	**6.3579$e+00$**	1.5703$e+01$	1.2607$e+01$
SD	5.8709$e+00$	**4.0371$e+00$**	5.4659$e+00$	5.2183$e+00$
Ackley				
AE	4.1419$e-06$	**1.6906$e-07$**	1.3810$e-06$	8.4684$e-06$
SD	1.3733$e-06$	**7.8367$e-08$**	6.0305$e-07$	4.1550$e-06$
Griewank				
AE	1.0549$e-09$	**9.2931$e-13$**	5.3670$e-11$	2.9475$e-09$
SD	6.4970$e-10$	**6.6629$e-13$**	2.3278$e-11$	1.2390$e-09$

Bold values indicate the best results

Lagrangian function for constraint handling (HCS-LSAL) proposed by Long et al. (2014), social spider optimization (SSO-C) proposed by Cuevas and Cienfuegos (2014), and adaptive firefly algorithm (AFA) proposed by Baykasoğlu and Ozsoydan (2015). These algorithms are also used to discuss the welded beam design problem and the pressure vessel design problem. A comparison of the best solution among these algorithms is given in Table 12. The results obtained by the compared methods are taken from the previous literature. It should be noted that the results obtained by NM-PSO are infeasible because the first two constraints were violated.

The statistical results of 30 independent runs obtained by the considered methods and MHS–PCLS are presented in Table 13. As can be observed in Table 13, MHS–PCLS obtained the function value 0.0126652 in every independent run with 10,000 function evaluations. MBA required only 7650 function evaluations to obtain the function value of 0.012665; however, its performance was less robust than MHS–PCLS from the perspective of statistics. The performance of TLBO is virtually the same as MHS–PCLS, whereas the performances of other algorithms such as ABC, CPSO, HPSO, CDE, and SSO-C are less robust than MHS–PCLS with more function evaluations.

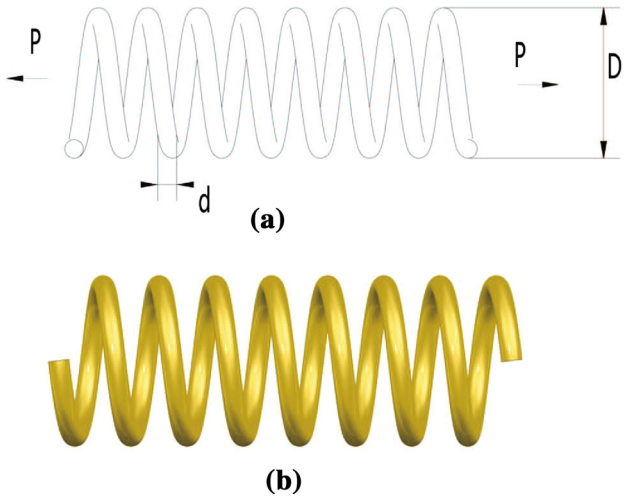


Fig. 6 The tension/compression string design problem

Welded beam design problem

The welded beam design problem was first proposed by Coello (2000). The aim of the problem is to design a welded beam that has minimum cost subject to constraints on shear stress(τ), bending stress in the beam (σ), buckling load on the bar P_c, end deflection of the beam (δ), and side constraints. There are four design variables: $h(x_1)$, $l(x_2)$, $t(x_3)$ and $b(x_4)$, as illustrated in Fig. 7. The mathematical model of this problem can be found in "Appendix".

A comparison of the best solutions given by the mentioned algorithms is presented in Table 14. A comparison of the statistical results is given in Table 15. The best solution was obtained by NM-PSO with an objective function value of 1.724717 after 80,000 function evaluations, whereas the best solution obtained by MHS–PCLS was 1.724852 with only 10,000 function evaluations. From Table 15, it can be observed that the performance of MHS–PCLS was more stable than the other algorithms and the $mFEs$ of MHS–PCLS were less than the other algorithm, except for TLBO.

Pressure vessel design problem

In the pressure vessel design problem, proposed by Kannan and Kramer (1994), the aim is to minimize the total cost, which consists of the material, forming cost and welding cost. A cylindrical vessel is capped at both ends by hemispherical heads, as displayed in Fig. 8. There are four design variables: $T_s(x_1$, thickness of the shell), $T_h(x_2$, thickness of the head),

Table 12 Comparison of the best solution obtained by various algorithms for the tension/compression spring design problem

Methods	$x_1(d)$	$x_2(D)$	$x_3(P_b)$	$g_1(\boldsymbol{x})$	$g_2(\boldsymbol{x})$	$g_3(\boldsymbol{x})$	$g_4(\boldsymbol{x})$	$f(\boldsymbol{x})$
DGA	0.051989	0.363965	10.890522	$-1.3e{-5}$	$-2.1e{-5}$	−4.061	−0.723	0.0126810
CPSO	0.051728	0.357644	11.244543	$-8.45e{-4}$	$-1.3e{-5}$	−4.051	−0.727	0.0126747
HPSO	0.051706	0.357126	11.265083	$-3.07e{-6}$	$1.4e{-6}$	−4.055	−0.727	**0.0126652**
CDE	0.051609	0.354714	11.410831	$-3.9e{-5}$	$-1.8e{-4}$	−4.048	−0.729	0.0126702
NM-PSO	0.051620	0.355498	11.333272	0.001	0.001	−4.061	−0.729	0.0126302
IHS	0.051154	0.349871	12.076432	0.000000	$-7.0e{-6}$	−4.027840	−0.736572	0.0126706
IPHS	0.051861	0.360858	11.050339	$-2.1963e{-6}$	$-2.8408e{-7}$	−4.061873	−0.724854	0.0126658
ABC	0.051749	0.358179	11.203763	−0.000000	−0.000000	−4.056663	−0.726713	**0.012665**
UABC	0.051691	0.356769	11.285988	−0.000000	−0.000000	−4.053886	−0.727694	**0.012665**
MBA	0.051656	0.355940	11.344665	0	0	−4.052248	−0.728268	**0.012665**
HCS-LSAL	0.051689	0.356718	11.28896	$-6.4e{-6}$	$-3.9e{-6}$	−4.054	−0.728	**0.0126652**
SSO-C	0.051689	0.356718	11.28896	$-6.4e{-6}$	$-3.9e{-6}$	−4.054	−0.728	**0.0126652**
AFA	0.051667	0.356198	11.319561	$-3.99e{-5}$	$2.42e{-5}$	−4.054	−0.728	0.0126653
MHS–PCLS	0.05168918	0.35672077	11.288788	$-2.2128e{-10}$	$-4.5078e{-11}$	−4.0538	−0.7277	**0.0126652**

Bold values indicate the best results

Table 13 Comparison of the statistical results of various algorithms for the tension/compression spring design problem

Algorithms	*Best*	*Mean*	*Worst*	*Std*	*NFE*	*Time*(*s*)
DGA	0.0126810	0.0127420	0.0129730	$5.90e{-05}$	80, 000	–
CPSO	0.0126747	0.0127300	0.0129240	$5.20e{-04}$	200, 000	–
HPSO	**0.0126652**	0.0127072	0.0127190	$1.58e{-05}$	81, 000	–
CDE	0.0126702	0.012703	0.012790	$2.70e{-05}$	204, 800	–
NM-PSO	0.0126302	0.0126314	0.0126330	$8.74e{-07}$	80, 000	–
ABC	**0.012665**	0.012709	NA	0.012813	30, 000	–
UABC	**0.012665**	0.012683	NA	$3.31e{-05}$	15, 000	–
TLBO	**0.012665**	0.01266576	NA	NA	10000	–
HCS-LSAL	**0.0126652**	0.0126683	0.0126764	$5.37e{-07}$	150, 000	–
MBA	**0.012665**	0.012713	0.012900	$6.30e{-05}$	**7650**	–
SSO-C	**0.012665233**	0.012764882	0.012867917	$9.29e{-5}$	25, 000	–
AFA	0.0126653	0.0126770	0.0127117	$1.28e{-5}$	50, 000	–
MHS–PCLS	**0.0126652**	**0.0126652**	**0.0126652**	**$3.6008e{-11}$**	10, 000	1.238

Bold values indicate the best results

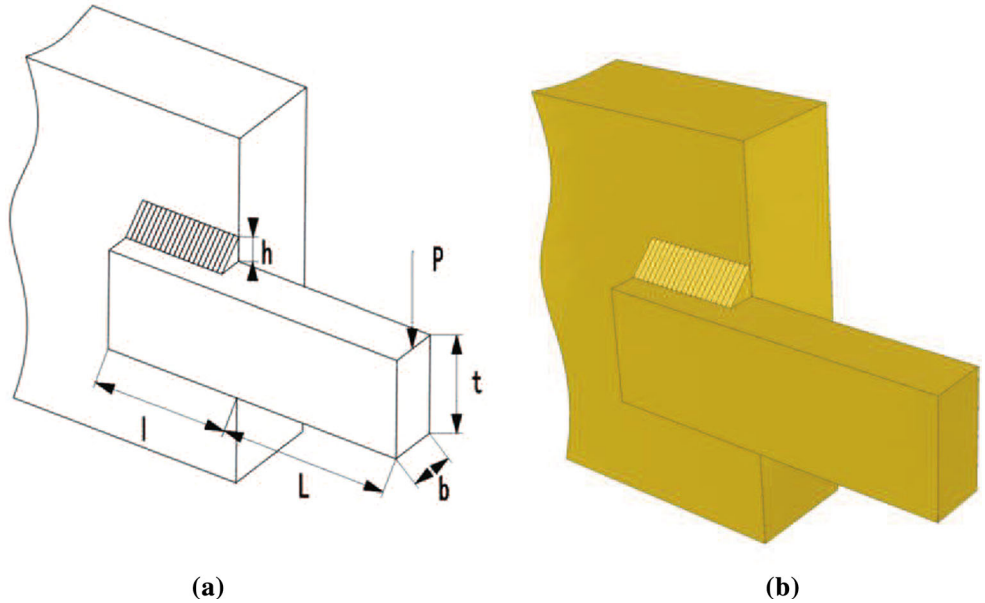


Fig. 7 Welded beam design problem

Table 14 Comparison of the best solution obtained by various algorithms for the welded beam problem

Methods	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	$f(\boldsymbol{x})$
DGA	0.205986	3.471328	9.020224	0.206480	1.728226
CPSO	0.202369	3.544214	9.048210	0.205723	1.728024
HPSO	0.205730	3.470489	9.036624	0.205730	1.724852
CDE	0.203137	3.542998	9.033498	0.206179	1.733462
IHS	0.20573	3.47049	9.03662	0.20573	1.7248
IPHS	0.20573	3.47049	9.03662	0.20573	1.7248
ABC	0.205730	3.470489	9.036624	0.205730	1.724852
UABC	0.205730	3.470489	9.036624	0.205730	1.724852
MBA	0.205729	3.470493	9.036626	0.205729	1.724853
NM-PSO	0.205830	3.468338	9.036624	0.205730	**1.724717**
SSO-C	0.2057296	3.470489	9.036624	0.205729	1.7248523
AFA	0.205730	3.470489	9.036624	0.205730	1.724852
MHS–PCLS	0.205730	3.470489	9.036624	0.205730	1.724852

Bold value indicates the best result

Table 15 Comparison of the statistical results given by various algorithms for the welded beam problem

Methods	*Best*	*Mean*	*Worst*	*Std*	*NFE*	*Time*(*s*)
DGA	1.728226	1.792654	1.993408	$7.47e{-}02$	80, 000	–
CPSO	1.728024	1.748831	1.782143	$1.29e{-}02$	200, 000	–
HPSO	1.724852	1.749040	1.814295	$4.01e{-}02$	81, 000	–
CDE	1.733461	1.768158	1.824105	$2.22e{-}02$	204, 800	–
ABC	1.724852	1.741913	NA	$3.1e{-}02$	30, 000	–
UABC	1.724852	1.724853	NA	$1.7e{-}06$	15, 000	–
TLBO	1.724852	1.72844676	NA	NA	**10,000**	–
NM-PSO	**1.724717**	1.726373	1.733393	$3.50e{-}03$	80, 000	–
MBA	1.724853	1.724853	1.724853	$\mathbf{6.94e{-}19}$	47, 340	–
SSO-C	1.7248523085	1.746461619	1.746461619	0.025729853	25, 000	–
AFA	1.724852	**1.724852**	**1.724852**	0.00	50, 000	–
MHS–PCLS	1.724852	**1.724852**	**1.724852**	$8.11e{-}10$	**10,000**	1.438

Bold values indicate the best results

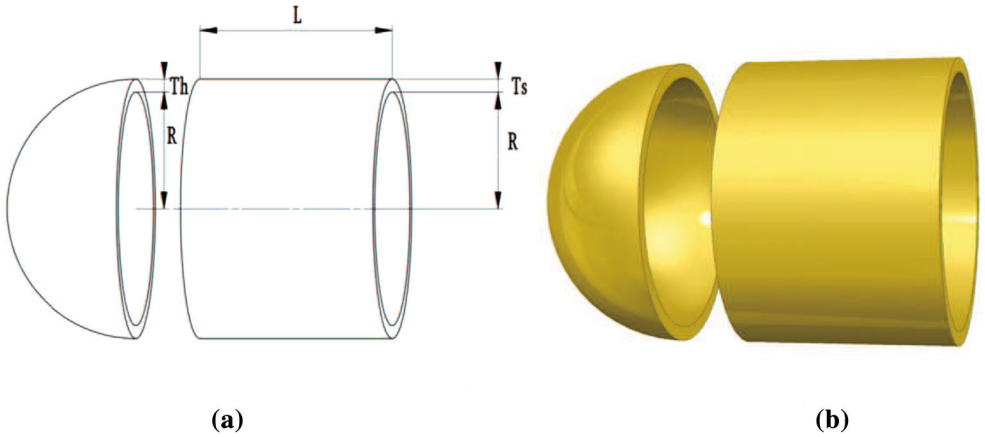


Fig. 8 Pressure vessel design problem

Table 16 Comparison of the best solution obtained by various algorithms for the pressure vessel design problem

Methods	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	$g_1(\boldsymbol{x})$	$g_2(\boldsymbol{x})$	$g_3(\boldsymbol{x})$	$g_4(\boldsymbol{x})$	$f(\boldsymbol{x})$
DGA	0.8125	0.4375	42.0974	176.6540	$-2.01e{-}02$	$-3.58e{-}02$	-24.7593	-63.3460	6059.9463
CPSO	0.8125	0.4375	42.0974	176.7465	$-1.37e{-}06$	$-3.59e{-}04$	-118.7687	-63.2535	6061.0777
HPSO	0.8125	0.4375	42.0984	176.6366	$-8.80e{-}02$	$-3.58e{-}02$	3.1226	-63.3634	6059.7143
CDE	0.8125	0.4375	42.0984	176.6376	$-6.67e{-}07$	$-3.58e{-}02$	-3.7051	-63.3623	6059.7340
NM-PSO	0.8036	0.3972	41.6392	182.4120	$3.65e{-}05$	$3.79e{-}05$	-1.5914	-57.5879	5930.3137
ABC	0.8125	0.4375	42.098446	176.636596	0.000000	-0.035881	-0.000226	-63.3634	**6059.71433**
UABC	0.8125	0.4375	42.098446	176.636596	0.000000	-0.035881	-0.00000	-63.3634	**6059.71433**
MBA	0.7802	0.3856	40.4292	198.4964	0	0	-86.3645	-41.5035	5889.3216
AFA	0.8125	0.4375	42.0984	176.636589	$-8.8e{-}07$	$-3.58e{-}02$	3.1839	-63.363	6059.71427
MHS–PCLS	0.8125	0.4375	42.098446	176.636596	$-1.6928e{-}11$	$-3.59e{-}02$	$-1.7664e{-}05$	-63.3634	**6059.71433**

Bold values indicate the best results

$R(x_3$, inner radius) and $L(x_4$, length of the cylindrical section of the vessel, not including the head). Among the four variables, T_s and T_h are integer multiples of 0.0625in; the available thicknesses of rolled steel plates. R and L are continuous variables. The mathematical model of this problem can be found in "Appendix".

A comparison of the best solutions given by the mentioned algorithms is presented in Table 16. A comparison of the statistical results is given in Table 17. The best solutions obtained by NM-PSO and MBA are infeasible because the first two design variables of NM-PSO and MBA are not integer multiples of 0.0625. From Table 17, it can be observed that MHS–PCLS and TLBO obtained more stable results with less $mFEs$ compared to the other algorithms.

Speed reducer design problem

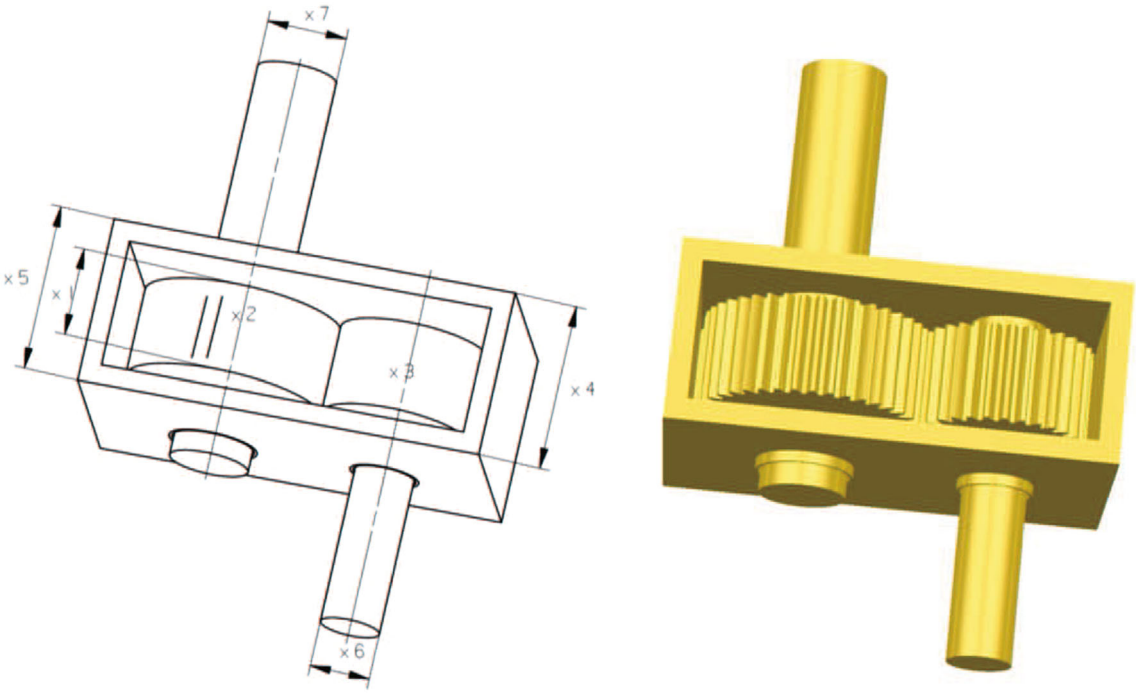
A speed reducer is part of the gear box of mechanical systems; it is also used for many other applications (Cuevas and Cienfuegos 2014). The design of a speed reducer is considered a challenging optimization problem in mechanical engineering (Jaberipour and Khorram 2010). The aim of this problem is to minimize the weight of the speed reducer while considering constraints on the bending stress of the gear teeth, surface stress, transverse deflections of the shafts, and stresses in the shafts (see Fig. 9). The problem contains seven variables: $b(x_1$, face width), $m(x_2$, module of teeth), $z(x_3$, number of teeth in the pinion), $l_1(x_4$, length of the first shaft between bearings), $l_2(x_5$, length of the second shaft between bearings),$d_1(x_6$, diameter of the first shaft), and $d_2(x_7$, diameter of the second shaft).This is a mixed integer programming problem. The third variable x_3(number of teeth) is an integer value and all other variables are continuous. The mathematical model of this problem is given in "Appendix".

A comparison of the best solutions compared to previous methods is presented in Table 18. Table 19 presents a comparison of the statistical results obtained by the different algorithms. The best solution of this problem was obtained by DSS-DE, DELC and MAL-DE with an objective function value of 2994.471066. As can be seen in Table 19, MHS–PCLS obtained similar results to DSS-DE, DELC and MAL-DE with significant fewer $mFEs$.

Table 17 Comparison of the statistical results given by various algorithms for the pressure vessel design problem

Methods	*Best*	*Mean*	*Worst*	*Std*	*NFE*	*Time*(*s*)
DGA	6059.9463	6177.2533	6469.3220	130.9297	80, 000	–
CPSO	6061.0777	6147.1332	6363.8041	86.45	240, 000	–
HPSO	**6059.7143**	6099.9323	6288.6770	86.20	81, 000	–
CDE	6059.7340	6085.2303	6371.0455	43.0130	204, 800	–
NM-PSO	5930.3137	5946.7901	5960.0557	9.161	80, 000	–
ABC	**6059.714**	6245.308	NA	205	30, 000	–
UABC	**6059.71433**	6192.116211	NA	204	15, 000	–
TLBO	**6059.71433**	**6059.71434**	NA	NA	**10, 000**	–
MBA	5889.3216	6200.64765	6392.5062	160.34	70, 650	–
AFA	6059.71427	6064.33605	6090.52614	11.28785	50, 000	–
MHS–PCLS	**6059.71433**	**6059.71434**	**6059.71439**	**1.28120*e*−05**	**10, 000**	1.301

Bold values indicate the best results



(a) **(b)**

Fig. 9 Speed reducer design problem

Case study: car side impact design

In this section, a case study of car side-impact design is investigated. A car side is a weak part of the entire car body. According to a Chinese accident statistical report for the year 2007 (Zhou 2015), the proportion of casualties caused by car side impact represented 36 % of the total accident casualties. Many countries have established their test standard of car side impact. In this paper, the European Enhanced Vehicle-Safety Committee (EEVC) side impact procedure is used. The EEVC side impact procedure offers the lowest safety standard for the dummy, including head injuries, load in abdomen, pubic force and rib deflection. The main objective of the design is to maintain the total weight minimized while additional constraints are addressed. These constraints include load in the abdomen(g_1), dummy upper chest (g_2), dummy middle chest (g_3), dummy lower chest (g_4), upper rib deflection(g_5), middle rib deflection (g_6), lower rib deflection (g_7), pubic force (g_8), velocity of V-Pillar at middle point (g_9), and velocity of front door at V-Pillar (g_{10}). There are 11 decision variables. These are the thicknesses of B-Pillar inner(x_1), B-Pillar reinforcement(x_2), floor side inner(x_3), cross members(x_4), door beam(x_5), door beltline reinforcement(x_6), roof rail (x_7), material of B-Pillar inner(x_8), floor side inner (x_9), barrier height(x_{10}), and hitting position (x_{11}). The finite element model of a test case is

Table 18 Comparison of the best solutions obtained by various algorithms for the speed reducer design problem

Methods	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(\boldsymbol{x})$
DSS-DE	3.50000000	0.700000	17.0000000	7.300000	7.715319	3.350214	5.286654	**2994.471066**
DELC	3.50000000	0.700000	17.0000000	7.300000	7.715319	3.350214	5.286654	**2994.471066**
HEA-ACT	3.500022	0.700000	17.00001	7.300427	7.715377	3.350230	5.286663	2994.499107
MDE	3.500010	0.700000	17	7.300156	7.800027	3.350221	5.286685	2996.356689
MAL-DE	3.50000000	0.700000	17.0000000	7.300000	7.715319	3.350214	5.286654	**2994.471066**
ABC	3.499999	0.7	17	7.3	7.8	3.350215	5.287800	**2994.471066**
UABC	3.50000	0.7	17	7.3	7.715320	3.350215	5.286654	2997.058412
MBA	3.50000000	0.700000	17.0000000	7.300033	7.715772	3.350218	5.286654	2994.482453
SSO-C	3.50000000	0.700000	17.0000000	7.30001	7.71532	3.35021	5.28665	2996.113298
AFA	3.50000000	0.700000	17.0000000	7.302489	7.800067	3.350219	5.286683	2996.372698
MHS–PCLS	3.50000000	0.700000	17	7.30000	7.7153199	3.3502146	5.2866545	2994.471068

Bold values indicate the best results

Table 19 Comparison of the statistical results given by various algorithms for the speed reducer design problem

Methods	*Best*	*Mean*	*Worst*	*Std*	*NFE*	*Time*(*s*)
DSS-DE	**2994.471066**	**2994.471066**	**2994.471066**	3.6e−12	30, 000	–
DELC	**2994.471066**	**2994.471066**	**2994.471066**	1.9e−12	NA	–
HEA-ACT	2994.499107	2994.613368	2994.752311	7.0e−02	40,000	–
MDE	2996.256689	2996.367220	NA	8.2e−03	24000	–
MAL-DE	**2994.471066**	**2994.471066**	**2994.471066**	**0.00**	120, 000	–
ABC	2997.058	2997.058	NA	0.00	30, 000	–
UABC	2994.471066	2994.471072	NA	5.98e−06	15, 000	–
TLBO	2996.34817	2996.34817	NA	0	10, 000	–
MBA	2994.482453	2996.769019	2999.652444	1.56e+00	**6300**	–
SSO-C	2996.113298	2996.113298	2996.113298	0.00	25, 000	–
AFA	2996.372698	2996.514874	2996.669016	0.09	50, 000	–
MHS–PCLS	2994.471068	2994.471077	2994.471106	7.142949e−06	10, 000	1.606

Bold values indicate the best results

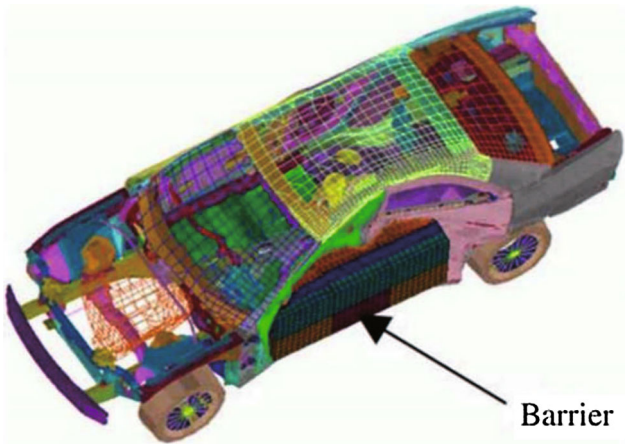


Fig. 10 The finite element model of the car side impact design (Gandomi et al. 2013a)

illustrated in Fig. 10. Gu et al. (2001) developed the response surface model (RSM) of the objective function and the constraints based on the Latin hypercube sampling method. The mathematical model of this problem is provided in "Appendix".

In this case study, the result obtained by MHS–PCLS is compared with PSO, DE, GA, firefly algorithm (FA)and cuckoo search algorithm (CS). The results of these algorithms are taken from the previous literature (Gandomi et al. 2011, 2013a). The simulations were conducted with 20,000 FEs for all the algorithms. Because the number of independent runs are not reported in Gandomi et al. (2011), Gandomi et al. (2013a), 30 independent runs were conducted for MHS–PCLS. The computation results are presented in Table 20. The results confirm that the proposed MHS–PCLS can achieve the best results in terms of $Mean$, $Worst$, and SD indexes. Although MHS–PCLS cannot obtain the best result of $Best$, it has the minimum solution differences, which indicates that it has the best robustness. It can be concluded that the MHS–PCLS is competitive and more robust compared to PSO, DE GA, FA, and CS in solving this complex design problem.

Table 20 Statistical results of the car side impact design example by different methods

Methods	PSO	DE	GA	FA	CS	MHS–PCLS
x_1	0.50000	0.50000	0.50005	0.50000	0.50000	0.50004
x_2	1.11670	1.11670	1.28017	1.36000	1.11643	1.11640
x_3	0.50000	0.50000	0.50001	0.50000	0.50000	0.50003
x_4	1.30208	1.30208	1.03302	1.20200	1.30208	1.30230
x_5	0.50000	0.50000	0.50001	0.50000	0.50000	0.50000
x_6	1.50000	1.50000	0.50000	1.12000	1.50000	1.50000
x_7	0.50000	0.50000	0.50001	0.50000	0.50000	0.50000
x_8	0.3450	0.3450	0.34994	0.3450	0.3450	0.34499
x_9	0.19200	0.19200	0.19200	0.19200	0.19200	0.19215
x_{10}	−19.54935	−19.54935	10.3119	8.87307	−19.54935	−19.5690
x_{11}	−0.00431	−0.00431	0.00167	−18.99808	−0.00431	0.19207
Best	22.84474	22.84298	22.85653	22.84298	**22.84294**	22.84361
Mean	22.89429	23.22828	23.51585	22.89376	22.85858	**22.84501**
Worst	23.21354	24.12206	26.240578	24.06623	23.25998	**22.84906**
SD	0.1507	0.34451	0.66555	0.16667	0.07612	**0.0011**
Time(s)	–	–	–	–	–	1.739

Bold values indicate the best results

Conclusions

A parallel chaotic local search enhanced harmony search (MHS–PCLS) algorithm was proposed in this paper. This algorithm conducts a parallel chaotic local search from several different initial points and thus reduces the sensitivity of the initial chaotic maps. This mechanism provides improved robustness in the search process. In the numerical simulations, the performance of MHS–PCLS with different chaotic maps was investigated. The ICMIC map performed marginally better than the other maps. Several well-known benchmark problems were tested, and the results confirmed that MHS–PCLS achieved statistically superior performance compare to the other HS variants with a significance level at 5 %. MHS–PCLS was further combined with a modified Deb's constraint handling method to adapt to constrained optimization problems. The test results of several engineering design optimization problems and a complex case study of car side impact design validated its effectiveness. MHS–PCLS not only obtains competitive results compared to the previous methods, but also requires fewer function evaluations in the majority of cases.The proposed method is thus an efficient solver in constrained optimization problems. In the future, MHS–PCLS can be extended to solve engineering design optimization problems involving expensive simulation.

Acknowledgements The authors would like to thank the cloud system in HUST for providing us the computing services. This research work is supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 51435009, 61232008 and 51421062, and Youth Science & Technology Chenguang Program of Wuhan under Grant no. 2015070404010187.

Appendix: Mathematical model of the design problems

1. Design of tension/compression spring

$$Minimize: \quad f(\boldsymbol{x}) = (x_3+2)x_2x_1^2, \tag{12}$$

$$Subject\ \ to: \begin{cases} g_1(\boldsymbol{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0 \\ g_2(\boldsymbol{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0 \\ g_3(\boldsymbol{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0 \\ g_4(\boldsymbol{x}) = \frac{x_1+x_2}{1.5} - 1 \le 0 \end{cases} \tag{13}$$

where $0.05 \le x_1 \le 2, 0.25 \le x_2 \le 1.3, 2 \le x_3 \le 15$.

2. Design of welded beam

$$\begin{aligned} Minimize: \quad f(\boldsymbol{x}) &= 1.10471x_1^2x_2 \\ &\quad + 0.04811x_3x_4(14.0+x_2), \end{aligned} \tag{14}$$

$$Subject\ \ to: \begin{cases} g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - 13000 \le 0 \\ g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - 30000 \le 0 \\ g_3(\boldsymbol{x}) = x_1 - x_4 \le 0 \\ g_4(\boldsymbol{x}) = 0.1047x_1^2 \\ \qquad + 0.04811x_3x_4(14.0+x_2) - 5.0 \le 0 \\ g_5(\boldsymbol{x}) = 0.125 - x_1 \le 0 \\ g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - 0.25 \le 0 \\ g_7(\boldsymbol{x}) = 6000 - P_c(\boldsymbol{x}) \le 0 \end{cases} \tag{15}$$

where $\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$, $\tau' = \frac{6000}{\sqrt{2}x_1x_2}$
$\tau'' = \frac{MR}{J}$, $M = 6000(14 + \frac{x_2}{2})$, $R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2}$
$J = 2\{\sqrt{2}x_1x_2[\frac{x_2^2}{12} + (\frac{x_1+x_3}{2})^2]\}$, $\sigma(\boldsymbol{x}) = \frac{504000}{x_4x_3^2}$
$\delta(\boldsymbol{x}) = \frac{2.1952}{x_3^2x_4}$, $P_c(\boldsymbol{x}) = 64746.022(1 - 0.0282346x_3)x_3x_4^3$
$0.1 \le x_1 \le 2, 0.1 \le x_2 \le 10, 0.1 \le x_3 \le 10, 0.1 \le x_4 \le 2$.

3. Design of pressure vessel

$$\begin{aligned} Minimize: \quad f(\boldsymbol{x}) &= 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 \\ &\quad + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \end{aligned} \tag{16}$$

$$Subject\ \ to: \begin{cases} g_1(\boldsymbol{x}) = -x_1 + 0.0193x_3 \le 0 \\ g_2(\boldsymbol{x}) = -x_2 + 0.00954x_3 \le 0 \\ g_3(\boldsymbol{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\ g_4(\boldsymbol{x}) = x_4 - 240 \le 0 \end{cases} \tag{17}$$

where $0 \le x_1 \le 99, 0 \le x_2 \le 99, 10 \le x_3 \le 200, 10 \le x_4 \le 200$.

4. Design of speed reducer

$$Subject\ \ to: \begin{cases} g_1(\boldsymbol{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0 \\ g_2(\boldsymbol{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0 \\ g_3(\boldsymbol{x}) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \le 0 \\ g_4(\boldsymbol{x}) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \le 0 \\ g_5(\boldsymbol{x}) = \frac{[745(x_4/x_2x_3)^2 + 16.9\times10^6]^{1/2}}{110x_6^3} - 1 \le 0 \\ g_6(\boldsymbol{x}) = \frac{[745(x_5/x_2x_3)^2 + 157.5\times10^6]^{1/2}}{85x_7^3} - 1 \le 0 \\ g_7(\boldsymbol{x}) = \frac{x_2x_3}{40} - 1 \le 0 \\ g_8(\boldsymbol{x}) = \frac{5x_2}{x_1} - 1 \le 0 \\ g_9(\boldsymbol{x}) = \frac{x_1}{12x_2} - 1 \le 0 \\ g_{10}(\boldsymbol{x}) = \frac{1.5x_6+1.9}{x_4} - 1 \le 0 \\ g_{11}(\boldsymbol{x}) = \frac{1.1x_6+1.9}{x_5} - 1 \le 0 \end{cases} \tag{19}$$

where $2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.3 \le x_5 \le 8.3, 2.9 \le x_4 \le 3.9, 5.0 \le x_4 \le 5.5$.

5. Car side impact design

$$\begin{aligned} Minimize: \quad f(\boldsymbol{x}) &= 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 - 1.508x_1(x_6^2 + x_7^2) \\ &\quad + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \end{aligned} \tag{20}$$

$$Subject\ \ to: \begin{cases} g_1 = F_a = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} - 1 \le 0 \\ g_2 = VC_u = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} \\ \quad +0.080405x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} - 0.32 \le 0 \\ g_3 = VC_m = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 \\ \quad +0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} \\ \quad +0.00121x_8x_{11} - 0.32 \le 0 \\ g_4 = VC_l = 0.074 - 0.061x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.27x_2^2 - 0.32 \le 0 \\ g_5 = \Delta_{ur} = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} - 32 \le 0 \\ g_6 = \Delta_{mr} = 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} \\ \quad -9.98x_7x_8 + 22.0x_8x_9 - 32 \le 0 \\ g_7 = \Delta_{lr} = 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} - 32 \le 0 \\ g_8 = F_p = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 - 4 \le 0 \\ g_9 = V_{MBP} = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} - 9.9 \le 0 \\ g_10 = V_{FD} = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 - 15.7 \le 0 \end{cases} \tag{21}$$

$$\begin{aligned} Minimize: \\ f(\boldsymbol{x}) &= 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ &\quad - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\ &\quad + 0.7854(x_4x_6^2 + x_5x_7^2) \end{aligned} \tag{18}$$

where $0.5 \le x_1 \sim x_7 \le 1.5, x_8, x_9 \in (0.192, 0.345), -30 \le x_{10}, x_{11} \le 30$.

References

Abedinpourshotorban, H., Hasan, S., Shamsuddin, S. M., & As' Sahra, N. F. (2016). A differential-based harmony search algorithm for the

optimization of continuous problems. *Expert Systems with Applications, 62*, 317–332

Akay, B., & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, *23*(4), 1001–1014.

Alatas, B. (2010). Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications*, *37*(8), 5682–5687.

Al-Betar, M. A., Awadallah, M. A., Khader, A. T., & Abdalkareem, Z. A. (2015). Island-based harmony search for optimization problems. *Expert Systems with Applications*, *42*(4), 2026–2035.

Al-Betar, M. A., Doush, I. A., Khader, A. T., & Awadallah, M. A. (2012). Novel selection schemes for harmony search. *Applied Mathematics and Computation*, *218*(10), 6095–6117.

Al-Betar, M. A., Khader, A. T., Geem, Z. W., Doush, I. A., & Awadallah, M. A. (2013). An analysis of selection methods in memory consideration for harmony search. *Applied Mathematics and Computation*, *219*(22), 10753–10767.

Arora, J. (2004). *Introduction to optimum design*. New York: Academic Press.

Askarzadeh, A., & Zebarjadi, M. (2014). Wind power modeling using harmony search with a novel parameter setting approach. *Journal of Wind Engineering and Industrial Aerodynamics*, *135*, 70–75.

Baykasoglu, A. (2012). Design optimization with chaos embedded great deluge algorithm. *Applied Soft Computing*, *12*(3), 1055–1067.

Baykasoğlu, A., & Ozsoydan, F. B. (2015). Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Applied Soft Computing*, *36*, 152–164.

Brajevic, I., & Tuba, M. (2013). An upgraded artificial bee colony (abc) algorithm for constrained optimization problems. *Journal of Intelligent Manufacturing*, *24*(4), 729–740.

Castelli, M., Silva, S., Manzoni, L., & Vanneschi, L. (2014). Geometric selective harmony search. *Information Sciences*, *279*, 468–482.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, *41*(2), 113–127.

Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, *16*(3), 193–203.

Cuevas, E., & Cienfuegos, M. (2014). A new algorithm inspired in the behavior of the social-spider for constrained optimization. *Expert Systems with Applications*, *41*(2), 412–425.

Das, S., Mukhopadhyay, A., Roy, A., Abraham, A., & Panigrahi, B. K. (2011). Exploratory power of the harmony search algorithm: Analysis and improvements for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *41*(1), 89–106.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, *15*(1), 4–31.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2), 311–338.

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39.

Eberhart, R. C., Kennedy, J., et al. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science, New York, NY* (Vol. 1, pp. 39–43).

Enayatifar, R., Yousefi, M., Abdullah, A. H., & Darus, A. N. (2013). Lahs: A novel harmony search algorithm based on learning automata. *Communications in Nonlinear Science and Numerical Simulation*, *18*(12), 3481–3497.

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & Structures*, *89*(23), 2325–2336.

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013a). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers*, *29*(1), 17–35.

Gandomi, A., Yang, X. S., Talatahari, S., & Alavi, A. (2013b). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, *18*(1), 89–98.

Gandomi, A. H., Yun, G. J., Yang, X. S., & Talatahari, S. (2013c). Chaos-enhanced accelerated particle swarm optimization. *Communications in Nonlinear Science and Numerical Simulation*, *18*(2), 327–340.

Gao, W. F., Liu, S. Y., & Huang, L. L. (2012). Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Communications in Nonlinear Science and Numerical Simulation*, *17*(11), 4316–4327.

Gao, L. Q., Li, S., Kong, X., & Zou, D. X. (2014a). On the iterative convergence of harmony search algorithm and a proposed modification. *Applied Mathematics and Computation*, *247*, 1064–1095.

Gao, K., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., & Chong, C. (2014b). Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. *Information Sciences*, *289*, 76–90.

García-Torres, J. M., Damas, S., Cordón, O., & Santamaría, J. (2014). A case study of innovative population-based algorithms in 3d modeling: Artificial bee colony, biogeography-based optimization, harmony search. *Expert Systems with Applications*, *41*(4), 1750–1762.

Geem, Z. W., Kim, J. H., & Loganathan, G. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, *76*(2), 60–68.

Gu, L., Yang, R., Tho, C., Makowskit, M., Faruquet, O., Li, Y., et al. (2001). Optimisation and robustness for crashworthiness of side impact. *International Journal of Vehicle Design*, *26*(4), 348–360.

He, Q., & Wang, L. (2007a). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, *20*(1), 89–99.

He, Q., & Wang, L. (2007b). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation*, *186*(2), 1407–1422.

Hosseini, S. D., Shirazi, M. A., & Ghomi, S. M. T. F. (2014). Harmony search optimization algorithm for a novel transportation problem in a consolidation network. *Engineering Optimization*, *46*(11), 1538–1552.

Huang, F. Z., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and computation*, *186*(1), 340–356.

Jaberipour, M., & Khorram, E. (2010). Two improved harmony search algorithms for solving engineering optimization problems. *Communications in Nonlinear Science and Numerical Simulation*, *15*(11), 3316–3331.

Jia, D., Zheng, G., & Khan, M. K. (2011). An effective memetic differential evolution algorithm based on chaotic local search. *Information Sciences*, *181*(15), 3175–3187.

Jordehi, A. R. (2015). Chaotic bat swarm optimisation (cbso). *Applied Soft Computing*, *26*, 523–530.

Kannan, B., & Kramer, S. N. (1994). An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design*, *116*(2), 405–411.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization*, *39*(3), 459–471.

Kaveh, A. (2014). Chaos embedded metaheuristic algorithms. In *Advances in metaheuristic algorithms for optimal design of structures* (pp. 369–391). Cham, Switzerland: Springer.

Koceski, S., Panov, S., Koceska, N., Zobel, P. B., & Durante, F. (2014). A novel quad harmony search algorithm for grid-based path finding. *International Journal of Advanced Robotic Systems*, *11*, 144–155.

Kramer, O. (2010). A review of constraint-handling techniques for evolution strategies. *Applied Computational Intelligence and Soft Computing, 2010*, 1–11.

Kundu, S., & Parhi, D. R. (2016). Navigation of underwater robot based on dynamically adaptive harmony search algorithm. *Memetic Computing*, *8*(2), 125–146.

Li, X., Qin, K., Zeng, B., Gao, L., & Su, J. (2016). Assembly sequence planning based on an improved harmony search algorithm. *The International Journal of Advanced Manufacturing Technology 84*(9), 2367–2380.

Long, W., Liang, X., Huang, Y., & Chen, Y. (2014). An effective hybrid cuckoo search algorithm for constrained global optimization. *Neural Computing and Applications*, *25*(3–4), 911–926.

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, *188*(2), 1567–1579.

Maleki, A., & Pourfayaz, F. (2015). Sizing of stand-alone photovoltaic/wind/diesel system with battery and fuel cell storage devices by harmony search algorithm. *Journal of Energy Storage*, *2*, 30–42.

Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M. N., Salcedo-Sanz, S., et al. (2013). A survey on applications of the harmony search algorithm. *Engineering Applications of Artificial Intelligence*, *26*(8), 1818–1831.

Mezura-Montes, E., & Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, *1*(4), 173–194.

Mohamed, A. W., & Sabry, H. Z. (2012). Constrained optimization based on modified differential evolution algorithm. *Information Sciences*, *194*, 171–208.

Moraglio, A., Togelius, J., & Silva, S. (2013). Geometric differential evolution for combinatorial and programs spaces. *Evolutionary Computation*, *21*(4), 591–624.

Omran, M. G., & Mahdavi, M. (2008). Global-best harmony search. *Applied Mathematics and Computation*, *198*(2), 643–656.

Pan, Q. K., Suganthan, P. N., Tasgetiren, M. F., & Liang, J. J. (2010). A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, *216*(3), 830–848.

Pearl, R., & Reed, L. J. (1920). On the rate of growth of the population of the united states since 1790 and its mathematical representation. *Proceedings of the National Academy of Sciences*, *6*(6), 275–288.

Phatak, S., & Rao, S. S. (1995). Logistic map: A possible random-number generator. *Physical Review E*, *51*(4), 3670.

Rao, R. V., Savsani, V. J., & Vakharia, D. (2011). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, *43*(3), 303–315.

Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing*, *13*(5), 2592–2612.

Sarvari, H., & Zamanifar, K. (2012). Improvement of harmony search algorithm by using statistical analysis. *Artificial Intelligence Review*, *37*(3), 181–215.

Schuster, H. G., & Just, W. (2006). *Deterministic chaos: An introduction*. New York: Wiley.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, *12*(6), 702–713.

Sivaraj, R., & Ravichandran, T. (2011). A review of selection methods in genetic algorithm. *International Journal of Engineering Science and Technology*, *1*(3), 3792–3797.

Talatahari, S., Azar, B. F., Sheikholeslami, R., & Gandomi, A. (2012). Imperialist competitive algorithm combined with chaos for global optimization. *Communications in Nonlinear Science and Numerical Simulation*, *17*(3), 1312–1319.

Wang, Y., & Yao, M. (2009). A new hybrid genetic algorithm based on chaos and pso. In *IEEE International conference on intelligent computing and intelligent systems, 2009. ICIS 2009*. IEEE (Vol. 1, pp. 699–703).

Wang, G. G., Guo, L., Gandomi, A. H., Hao, G. S., & Wang, H. (2014). Chaotic krill herd algorithm. *Information Sciences*, *274*, 17–34.

Yang, X. S., & Deb, S., (2009). Cuckoo search via lévy flights. In *World congress on nature and biologically inspired computing 2009. NaBIC 2009*. IEEE (pp. 210–214).

Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms*. Beckington: Luniver press.

Yang, D., Liu, Z., & Zhou, J. (2014). Chaos optimization algorithms based on chaotic maps with different probability distribution and search speed for global optimization. *Communications in Nonlinear Science and Numerical Simulation*, *19*(4), 1229–1246.

Yassen, E. T., Ayob, M., Nazri, M. Z. A., & Sabar, N. R. (2015). Meta-harmony search algorithm for the vehicle routing problem with time windows. *Information Sciences*, *325*, 140–158.

Yi, J., Gao, L., Li, X., & Gao, J. (2016a). An efficient modified harmony search algorithm with intersect mutation operator and cellular local search for continuous function optimization problems. *Applied Intelligence*, *44*(3), 725–753.

Yi, J., Li, X., Xiao, M., Xu, J., & Zhang, L. (2016b). Construction of nested maximin designs based on successive local enumeration and modified novel global harmony search algorithm. *Engineering Optimization*, 1–20.

Yuan, X., Zhao, J., Yang, Y., & Wang, Y. (2014). Hybrid parallel chaos optimization algorithm with harmony search algorithm. *Applied Soft Computing*, *17*, 12–22.

Zahara, E., & Kao, Y. T. (2009). Hybrid nelder-mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, *36*(2), 3880–3886.

Zarei, O., Fesanghary, M., Farshi, B., Saffar, R. J., & Razfar, M. (2009). Optimization of multi-pass face-milling via harmony search algorithm. *Journal of Materials Processing Technology*, *209*(5), 2386–2392.

Zeng, B., & Dong, Y. (2016). An improved harmony search based energy-efficient routing algorithm for wireless sensor networks. *Applied Soft Computing*, *41*, 135–147.

Zhao, F., Liu, Y., Zhang, C., & Wang, J. (2015). A self-adaptive harmony pso search algorithm and its performance analysis. *Expert Systems with Applications*, *42*(21), 7436–7455.

Zheng, Y. J., Zhang, M. X., & Zhang, B. (2016). Biogeographic harmony search for emergency air transportation. *Soft Computing*, *20*(3), 967–977.

Zhou, Y. (2015). *Analysis, improvement and application of differential evolution (Unpublished doctoral dissertation)*. China: Huazhong University of Science and Technology.