Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 139464, 20 pages
http://dx.doi.org/10.1155/2013/139464

Research Article

Cellular Harmony Search for Optimization Problems

Mohammed Azmi Al-Betar,[1,2] Ahamad Tajudin Khader,[1] Mohammed A. Awadallah,[1] Mahmmoud Hafsaldin Alawan,[1] and Belal Zaqaibeh[3]

[1] *School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia*
[2] *Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan*
[3] *Department of Computer Science, Jadara University, P.O. Box 733, Irbid, Jordan*

Correspondence should be addressed to Mohammed Azmi Al-Betar; mohbetar@cs.usm.my

Received 8 July 2013; Revised 15 August 2013; Accepted 16 August 2013

Academic Editor: Zong Woo Geem

Structured population in evolutionary algorithms (EAs) is an important research track where an individual only interacts with its neighboring individuals in the breeding step. The main rationale behind this is to provide a high level of diversity to overcome the genetic drift. Cellular automata concepts have been embedded to the process of EA in order to provide a decentralized method in order to preserve the population structure. Harmony search (HS) is a recent EA that considers the whole individuals in the breeding step. In this paper, the cellular automata concepts are embedded into the HS algorithm to come up with a new version called cellular harmony search (cHS). In cHS, the population is arranged as a two-dimensional toroidal grid, where each individual in the grid is a cell and only interacts with its neighbors. The memory consideration and population update are modified according to cellular EA theory. The experimental results using benchmark functions show that embedding the cellular automata concepts with HS processes directly affects the performance. Finally, a parameter sensitivity analysis of the cHS variation is analyzed and a comparative evaluation shows the success of cHS.

1. Introduction

The optimization techniques have the utility of navigating the search space using effective operators driven by control parameters. The tricky point of the success of any optimization method is its ability to strike a suitable balance between exploration (diversification) and exploitation (intensification) of the problem search space [1]. Exploration is the optimization method capability of navigating a promising region of the search space, if necessary, while exploitation refers to the capability of fine-tuning the already-navigated regions to converge into the local optima [1].

Harmony search (HS) algorithm is a recent evolutionary algorithm (EA) proposed by Geem et al. [2] to imitate the musical improvisation process. Due to its advantages over other optimization methods, it stipulates fewer mathematical requirements in the initial search [3]. It has a novel stochastic derivative which reduces the number of iterations required to converge towards local minima [4], in addition to being simple, adaptable, general, and scalable [5]. Therefore, HS algorithm has been intensively tailored for several optimization problems such as timetabling [6, 7], nurse restoring [8], space allocation [9], and many others [10–13]. However, due to the complex nature of some optimization problems and the avoidance of a premature convergence situation, HS theories are modified [14]. Furthermore, the control parameter adaptation for HS is also studied [5, 15–17].

In a procedural context, HS, which is an iterative improvement algorithm, initiates with a population of random individuals stored in harmony memory (HM). At each iteration, a new individual is generated based on three operators: (i) memory consideration, which *selects* the variables of a new individual from whole HM individuals; (ii) pitch adjustment, which is responsible for local improvement, and (iii) random consideration, used to provide random elements for the new individual. The new individual is then evaluated

Table 1: CEC'2005 functions.

Abb.	Function name	Search range	x^*	$f(x^*)$
f_1	Shifted sphere function	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−450
f_2	Shifted Schwefel's problem 1.2	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, \ldots, o_D)$	−450
f_3	Shifted rotated high conditioned elliptic function	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−450
f_4	Shifted Schwefel's problem 1.2 with noise in fitness	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−450
f_5	Schwefel's problem 2.6 with global optimum on bounds	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−310
f_6	Shifted Rosenbrock's function	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	390
f_7	Shifted rotated Griewank's function without bounds	$[0, 600]$	$\mathbf{o} = (o_1, o_2, o_D)$	−180
f_8	Shifted rotated Ackley's function with global optimum on bounds	$[-32, 32]$	$\mathbf{o} = (o_1, o_2, o_D)$	−140
f_9	Shifted rastrigin's function	$[-5, 5]$	$\mathbf{o} = (o_1, o_2, o_D)$	−330
f_{10}	Shifted rotated Rastrigin's function	$[-5, 5]$	$\mathbf{o} = (o_1, o_2, o_D)$	−330
f_{11}	Shifted rotated Weierstrass function	$[-0.5, 0.5]$	$\mathbf{o} = (o_1, o_2, o_D)$	90
f_{12}	Schwefel's problem 2.13	$[-\pi, \pi]$	α	−460
f_{13}	Expanded extended Griewank's plus Rosenbrock's function (F8F2)	$[-5, 5]$	$\mathbf{o} = (o_1, o_2, o_D)$	−130
f_{14}	Shifted rotated expanded Scaffer's F6	$[-100, 100]$	$\mathbf{o} = (o_1, o_2, o_D)$	−300
f_{15}	Hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	120
f_{16}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	120
f_{17}	Rotated hybrid composition function with noise in fitness	$[-5, 5]$	$\mathbf{o}_1$	120
f_{18}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	10
f_{19}	Rotated hybrid composition function with a narrow basin for the global optimum	$[-5, 5]$	$\mathbf{o}_1$	10
f_{20}	Rotated hybrid composition function with the global optimum on the bounds	$[-5, 5]$	$\mathbf{o}_1$	10
f_{21}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	360
f_{22}	Rotated hybrid composition function with high condition number matrix	$[-5, 5]$	$\mathbf{o}_1$	360
f_{23}	Noncontinuous rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	360
f_{24}	Rotated hybrid composition function	$[-5, 5]$	$\mathbf{o}_1$	260
f_{25}	Rotated hybrid composition function without bounds	$[2, 5]$	$\mathbf{o}_1$	260

and replaces the worst individual in HM, if it is better. This process is cycled until a desired condition is met (see Table 3).

As other EAs, HS algorithm interacts with whole individuals in the HM during each breeding step. The update process selects the worst individual from a single HM and replaces it with a new one, if better. The decentralized methods used in other structured EAs have shown that the performance of EA is improved. Examples include cellular genetic algorithm (cGA) [18], distributed EA (dEA) [19], cellular PSO [20], and others [21]. The main idea of these structured methods is to partition the population into several sets with common features [22].

Cellular genetic algorithm (cGA), in particular, is a decentralized method where the population is represented as a toroidal grid of two-dimensions, as shown in Figure 3 [23, 24]. The individuals are located in this toroidal in a predefined topology and solely interact with their nearest neighbors in the breeding step [22]. Note that all the neighborhoods have the same size and identical shape. This concept embedded in cGA provides useful advantages for the optimization domain [23] and parallel implementations [25, 26] because it assists in providing a high-level of diversity and yields a small diffusion of solutions through the search. The cGA provides a proper exploitation power inside each neighborhood of an individual by the operators of GA [22]. Theoretically, it has been shown that nonrandom mating keeps genetic diversity at higher level, thus preventing the algorithms from converging prematurely to the local optima [27].

The main objective of this paper is to embed the cellular automata concepts in the HS algorithm optimization framework, where the HM is arranged as a two-dimensional toroidal grid. The population diffusion will be expectably preserved, maintaining a high-level of diversity during the search, thus avoiding genetic drift. The improvisation process of HS algorithm is adjusted to interact with the neighborhoods of specific individuals. The updating process of HM is done within the neighborhoods of that individual. The results show that the new decentralized version of HS (i.e., cHS) algorithm improves the performance of HS using standard benchmark functions.

The rest of the paper is organized as follows. The basics of HS algorithm are described in Section 2. The proposed cellular harmony search (cHS) is discussed in Section 3. Results of the experiments are presented in Section 4. Finally, the conclusion and promising future research directions are provided in Section 5.

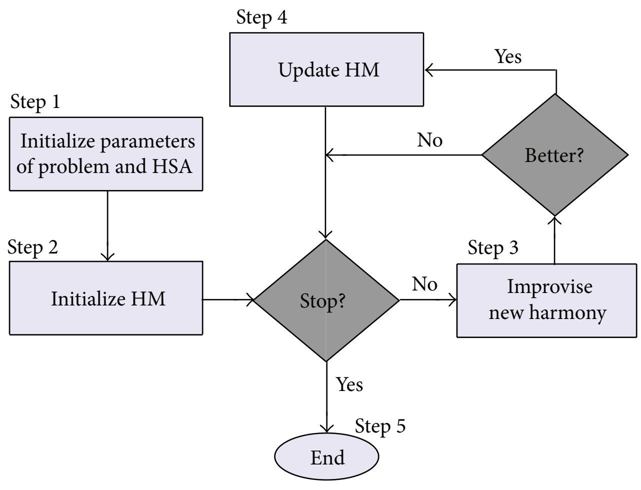


Figure 1: The flowchart of the HS algorithm.

2. The Harmony Search Algorithm

The harmony search (HS) algorithm is a recent evolutionary approach proposed by Geem et al. [2]. It is initiated with a set of individuals stored in an augmented matrix called harmony memory (HM). It is a centralized algorithm where at each breeding step, it generates a new individual by interacting with the whole individuals in HM. HS follows three rules in the breeding step to generate a new individual: memory consideration, random consideration, and pitch adjustment. If the new individual is better than the worst individual in the whole HM, the replacement process is triggered. This process is repeated as many times as the HS is stagnated. The steps of HS algorithm are flowcharted in Figure 1, where each step is described below in more detail.

Step 1 (initialize parameters). The optimization problem is initially represented as $\min\{f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}\}$, where $f(\mathbf{x})$ is the objective function and $\mathbf{x} = \{x_i \mid i = 1, \ldots, N\}$ is the set of decision variables. $\mathbf{X} = \{\mathbf{X}_i \mid i = 1, \ldots, N\}$ is the possible value range for each decision variable, where $\mathbf{X}_i \in [LB_i, UB_i]$, and LB_i and UB_i are the lower and upper bounds for the decision variable x_i, respectively, and N is the number of decision variables. The parameters of the HS algorithm are also predefined in the following step.

(a) The harmony memory consideration rate (HMCR), used in the breeding step to determine whether the value of a decision variable is to be selected from the individuals stored in the harmony memory (HM).

(b) The harmony memory size (HMS) which determines the number of individuals in HM.

(c) The pitch adjustment rate (PAR), which is used to decide the adjustments of some decision variables selected from memory.

(d) The distance bandwidth (BW) which determines the distance of the adjustment that occurs to the individual in the pitch adjustment operator.

(e) The number of improvisations (NI) which is similar to the number of generations.

Step 2 (initialize the harmony memory (HM)). The harmony memory (HM) is a matrix of size $N \times \text{HMS}$ which includes sets of individuals determined by HMS (see (1)). In this step, these individuals are randomly generated as follows: $x_i^j = LB_i + (UB_i - LB_i) \times U(0,1)$, for all $i = 1, 2, \ldots, N$ and for all $j = 1, 2, \ldots, \text{HMS}$, and $U(0,1)$ generate a random number between 0 and 1. Consider

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

Step 3 (improvise a new individual). The HS algorithm generates a new individual, $\mathbf{x}' = (x_1', x_2', \ldots, x_N')$, using three operators: (1) memory consideration, (2) random consideration, and (3) pitch adjustment.

Memory Consideration. In memory consideration, the value of the first decision variable in the new individual x_1' is randomly *selected* from the historical values, $\{x_1^1, x_1^2, \ldots, x_1^{\text{HMS}}\}$, stored in whole HM individuals. Values of the other decision variables, $(x_2', x_3', \ldots, x_N')$, are sequentially assigned in a similar way with a probability of HMCR, where HMCR $\in (0,1)$. It is worth mentioning that this process interacts with the whole individuals in HM which might lead to a premature convergence situation due to the genetic drift.

Random Consideration. Decision variables that are not assigned with values according to memory consideration are randomly assigned according to their possible range by random consideration with a probability of (1-HMCR) as follows:

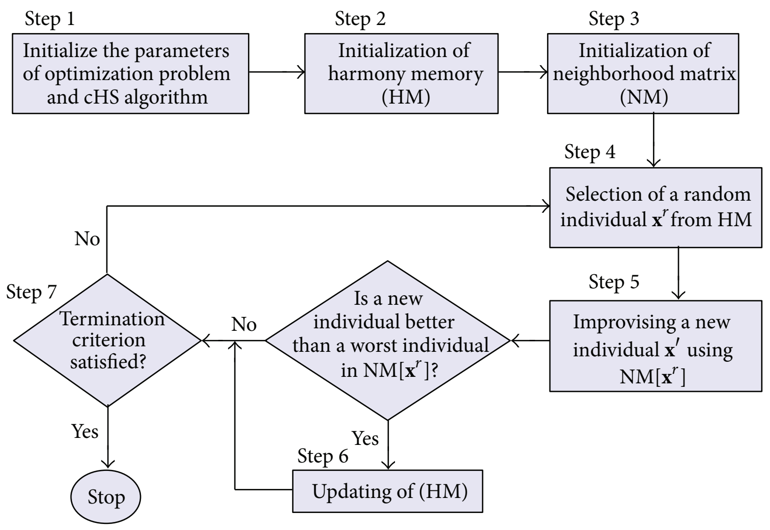
$$x_i' \longleftarrow \begin{cases} x_i' \in \left\{x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}\right\} & \text{w.p.} \quad \text{HMCR}, \\ x_i' \in \mathbf{X}_i & \text{w.p.} \quad 1 - \text{HMCR}. \end{cases} \tag{2}$$

Pitch Adjustment. Each decision variable x_i' of a new individual, $\mathbf{x}' = (x_1', x_2', x_3', \ldots, x_N')$, that has been assigned a value by memory consideration is pitch adjusted with the probability of PAR, where PAR $\in (0,1)$ as follows:

$$\text{Pitch adjusting decision for } x_i' \longleftarrow \begin{cases} \text{Yes} & \text{w.p.} \quad \text{PAR}, \\ \text{No} & \text{w.p.} \quad 1 - \text{PAR}. \end{cases} \tag{3}$$

In pitch adjustment, if the decision for x_i' is Yes, the value of x_i' is modified to its neighboring value as follows: $x_i' = x_i' \pm U(0,1) \times \text{BW}$.

Step 4 (update HM). If the new individual, $\mathbf{x}' = (x_1', x_2', \ldots, x_N')$, has better objective function value than that of the worst individual $\mathbf{x}^{\text{worst}}$ stored in HM (i.e., $\mathbf{x}^{\text{worst}} = \mathbf{x}^{\text{HMS}}$ in case HM is sorted), the worst individual will be replaced



Step 1
Initialize the parameters of optimization problem and cHS algorithm
Step 2
Initialization of harmony memory (HM)
Step 3
Initialization of neighborhood matrix (NM)
Step 4
Selection of a random individual $\mathbf{x}^r$ from HM
Step 5
Improvising a new individual $\mathbf{x}'$ using NM[$\mathbf{x}^r$]
Is a new individual better than a worst individual in NM[$\mathbf{x}^r$]?
No
Yes
Step 6
Updating of (HM)
Step 7
Termination criterion satisfied?
No
Yes
Stop

Figure 2: The flowchart of cHS.

by the new one. Note that the worst individual is selected from the whole HM where the decentralized structure of the population is not observed.

Step 5 (check the termination rule). The HS algorithm will repeat Steps 3 and 4 of HS until a termination rule is met, which is normally decided by the value of NI parameter.

3. Cellular Harmony Search (cHS) Algorithm

There has been much interest from researchers and scientists from different fields exploiting the cellular automata (CA) in physics, biology, social science, computer science, and so on. The initial concepts of CA were developed by Neumann [28] and have been an effective research tool to be incorporated with a wide variety of disciplines.

The concepts of cellular automata (CA) are normally concerned with individual perspective. The main idea from CA is to provide a population of a particular structure formulated as a toroidal grid. The cell in the toroidal grid refers to an individual who communicates with his closest neighboring individuals so that all the individuals have exactly the same number of neighbors. This leads us to a kind of locality known as *isolation by distance*. Normally, the Manhattan distance is used to measure the distance between any individual and his neighbors. Note that the neighboring individuals have identical shapes and the same size [22].

There exist two-different kinds of cellular models based on how the breeding cycle is performed to the individuals. To put it differently, if the cycle is performed to the whole individuals at the same time, the cellular model is said to be synchronous, where the individuals of the next generation are simultaneously build. On the other hand, if the individuals of the population are sequentially updated with a particular order policy, an asynchronous cellular model is stated. For more discussion about the theory of cellular automata, relevant papers can be seen in [22, 29].

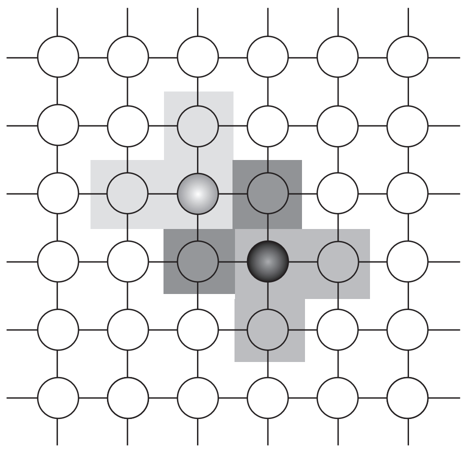
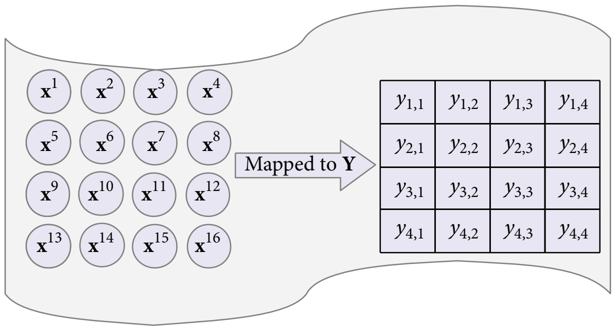


Figure 3: The population arrangement based on cellular structure where each cell is an individual in HM.

Cellular harmony search (cHS) algorithm can be considered as a new decentralized variation of HS which hinges on the structured HM. The individuals in the HM are arranged in the form of two-dimensional toroidal grid. This is meant to keep a high level of diversity during the breeding step and thus increases the chance to converge into global minima. This can be achieved by avoiding the genetic drift and providing a more suitable population diffusion during the search.

Some steps of cHS algorithm to the original version of HS algorithm presented in Section 2 have been adjusted. The adjustments are flowcharted in Figure 2. The breeding step of cHS solely interacts with the neighboring individuals of a randomly selected individual using "cellular memory consideration" operators. The update of HM step is adjusted to replace the worst individual amongst the neighboring individuals with a new individual, but not amongst the whole HM. Figure 3 shows the population on cellular structure (toroidal grid population), which inspires the concept of



x^1 x^2 x^3 x^4
x^5 x^6 x^7 x^8
x^9 x^{10} x^{11} x^{12}
x^{13} x^{14} x^{15} x^{16}
Mapped to **Y**
$y_{1,1}$ $y_{1,2}$ $y_{1,3}$ $y_{1,4}$
$y_{2,1}$ $y_{2,2}$ $y_{2,3}$ $y_{2,4}$
$y_{3,1}$ $y_{3,2}$ $y_{3,3}$ $y_{3,4}$
$y_{4,1}$ $y_{4,2}$ $y_{4,3}$ $y_{4,4}$

Figure 4: The individuals of HM mapped into the toroidal mesh **Y**.

small neighborhood regions in the cellular automata. Particularly, the overlap of the neighborhood provides implicit technique migration into population. Therefore, the best solutions are diffused smoothly in the whole population, where the diversity of the cellular harmony search is preserved throughout the search. The cellular HS model includes several components.

(1) Cell: the selected random individual in the population (the number of individual is HMS).

(2) Cell space: the set of the whole individuals in HM.

(3) Neighborhood: the set of potential mates of any individual.

(4) Neighborhood shapes: the way of selecting the neighborhoods of the cell as seen in Figure 5.

(5) Discrete time limit: the number of generations in HS algorithm which is normally determined by NI parameter.

The detailed steps of cHS are discussed in the steps below.

Step 1 (initialize cHS parameters and optimization problem). It is clear that the successful search of any metaheuristic method is based on skillful parameter setting. The parameters have different effects on optimization solutions. The parameters of cHS are harmony memory size (HMS), harmony memory consideration rate (HMCR), pitch adjusting rate (PAR), number of improvisation (NI), and the size of neighborhood (NH) determined by cellular structure (see Figure 5), where if NH = L9, this means the neighbors structure is length with 8 neighbors.

Step 2 (initialize the harmony memory (HM)). This step is the same as Step 2 in the original version of HS. Note that the individuals of the HM are arranged as a two-dimensional toroidal grid as shown in Figure 4.

Step 3 (initialization of neighborhood matrix (NM)). NM is a binary matrix of size HMS×HMS (see (4)). The binary values are assigned to each element $\text{NM}_{i,j}$ based on NH parameter. This matrix is used during the breeding step to determine the neighboring individuals of any randomly selected individual. The **NM** matrix is filled by binary value as in (5). Consider

$$\mathbf{NM} = \begin{bmatrix} \text{NM}_{1,1} & \cdots & \cdots & \text{NM}_{1,\text{HMS}} \\ \vdots & \ddots & \ddots & \vdots \\ \text{NM}_{\text{HMS},1} & \cdots & \cdots & \text{NM}_{\text{HMS},\text{HMS}} \end{bmatrix}, \quad (4)$$

$$\text{NM}_{i,j} \longleftarrow \begin{cases} 1 & \mathbf{x}^j \in \mathcal{N}\left(\mathbf{x}^i\right) \quad \forall i \wedge j \in \{1, \dots, \text{HMS}\}. \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

The $\mathcal{N}(\mathbf{x}^{\mathbf{i}})$ is a set of all neighboring individuals of the individual $\mathbf{x}^i$ arranged in two-dimensional toroidal mesh. This set is determined based on a *neighborhood shape* as seen in Figure 5.

Figure 4 shows how the HM individuals are mapped to toroidal grid. Note that the element $Y_{1,1}$ reflects the index of the individual 1 in HM, while the $Y_{K,K}$ reflects the individual index HMS in HM. The HMS value is the square value of K (i.e., K^2 = HMS).

To map the element $(Y_{i,j})$ in toroidal grid **Y** and the individual index r in HM, the following will be used:

$$r = \left((i-1) \times K + j\right). \quad (6)$$

To map the index of the individual r in HM to the element $(Y_{i,j})$ in the matrix NM, the following will be used:

$$i = \text{int}\left(\frac{(r-1)}{K}\right) + 1, \quad (7)$$

$$j = ((r-1) \bmod K) + 1. \quad (8)$$

This mapping mechanism between the individual index of HM and the elements in **Y** is very useful to determine the neighboring of any individual in HM. As shown in Figure 5, there are several neighborhood shapes to determine the neighbors of any individual. For example, the L5 neighborhood shape takes the nearest neighbors of a given cell axial direction. Therefore, to determine the individual indexes in HM that belong to the neighbors of the individual index r using L5, the i and j should be calculated using (7) and (8). And the set of neighbors of individual r called $\vartheta = \{Y_{i-1,j}, Y_{i+1,j}, Y_{i,j-1}, Y_{i,j+1}\}$, then (8) is used to map the elements in **Y** to the corresponding individual indexes in HM. The same step is used if different neighborhood shapes (i.e., L9, C9,...) are used to determine the neighboring individuals of $\mathbf{x}^r$.

As can be noted, the individuals in the set ϑ will be assigned by 1 in **NM** for the individual $\mathbf{x}^r$ while others will be assigned by 0.

Step 4 (select random individual from HM). The selection of a random individual $\mathbf{x}^r$, $r \in \{1, \dots, \text{HMS}\}$, is done to determine the neighboring individuals taken from the matrix **NM** (i.e., $\xi_r = \{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^m\}$), where m is the number of the neighboring individuals; that is, $m \leq$ HMS. The neighboring individuals interact together in order to generate the new individual in the next step.

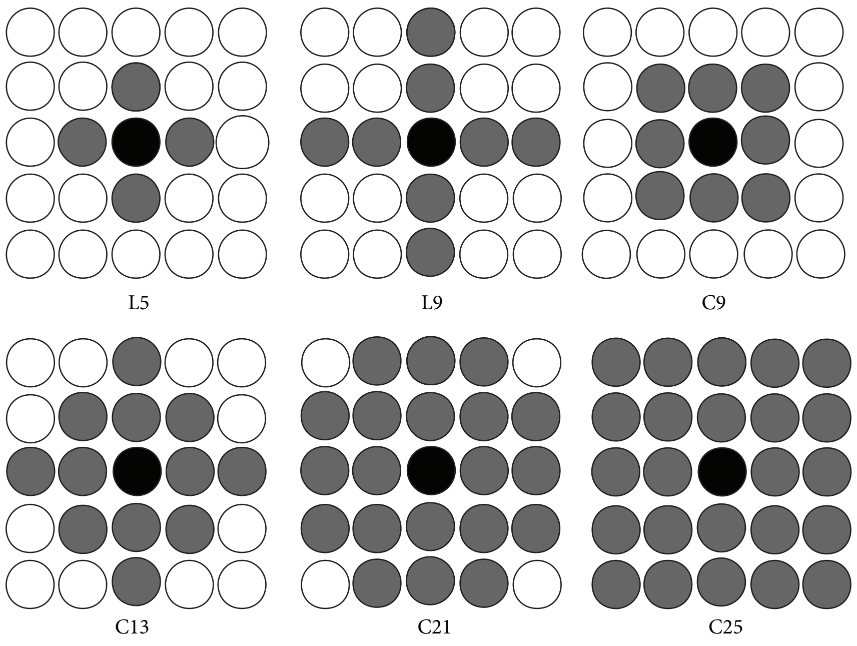


Figure 5: Neighboring shapes [22].

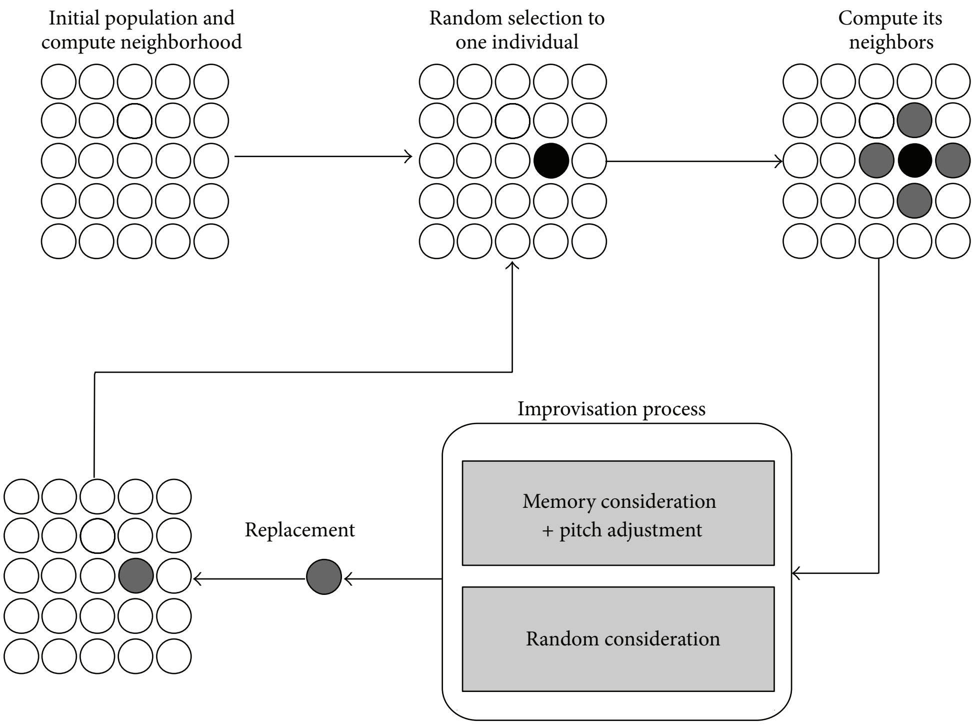
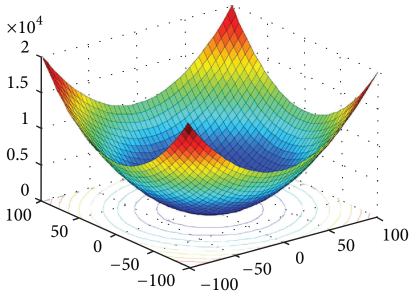

Initial population and compute neighborhood
Random selection to one individual
Compute its neighbors
Improvisation process
Memory consideration + pitch adjustment
Random consideration
Replacement

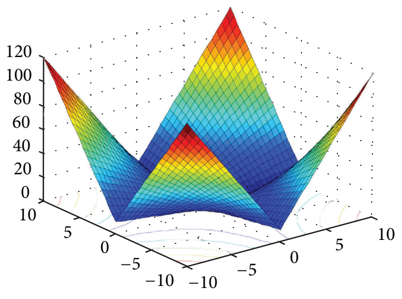
Figure 6: The processes fluency of cHS algorithm.

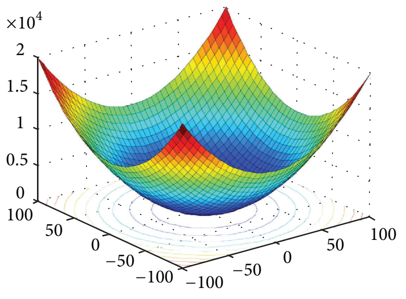
Step 5 (generate a new individual). In this step, a new individual, $\mathbf{x} = (\acute{x}_1, \acute{x}_2, \ldots, \acute{x}_N)$, is generated based on the three operators: (1) cellular memory consideration, (2) pitch adjustment, and (3) random consideration. The whole process is drawn in Figure 6.

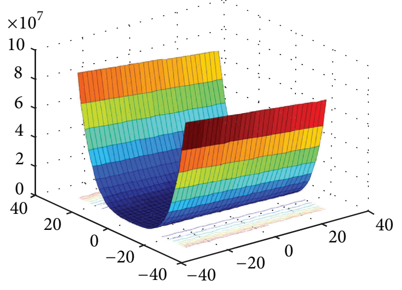
The pitch adjustment and random consideration operators in cHS algorithm are the same as those in the original version of HS algorithm. However, the memory consideration is modified to be inline with the concepts of cellular automata as follows.

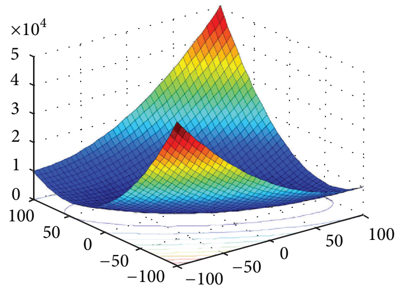
Cellular Memory Consideration. As in cellular GA, the cellular memory consideration solely interacts with the close individuals of $\mathbf{x}^r$ taken from the set of ξ_r. The other individuals are not used in the breeding step. The value of the first decision

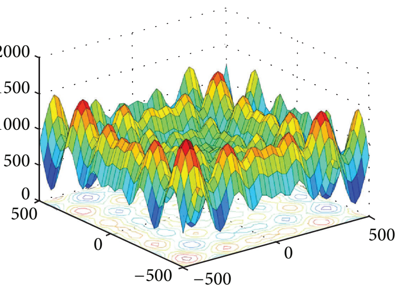

(a) 2D sphere function

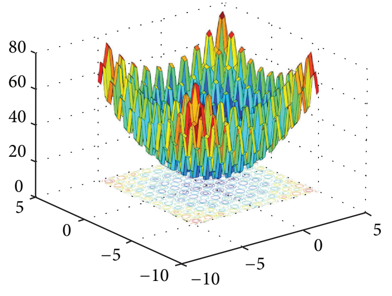

(b) 2D Schwefel's problem 2.22

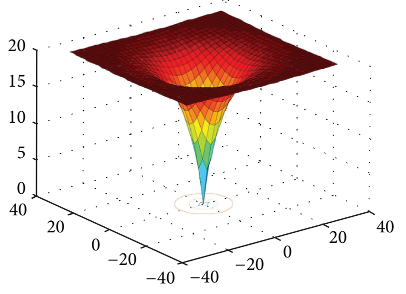

(c) 2D step function

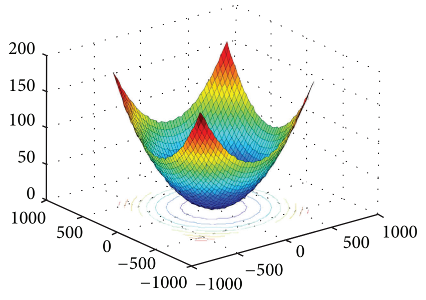

(d) 2D Rosenbrock function


(e) 2D Rotated hyper-ellipsoid function


(f) 2D Schwefel's problem 2.26 function


(g) 2D Rastrigin function


(h) 2D Ackley function


(i) 2D Griewank function

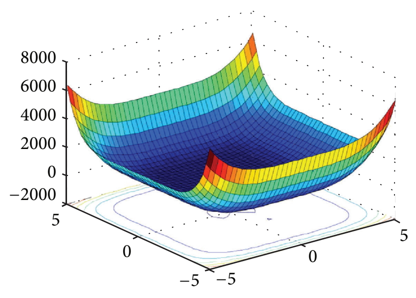
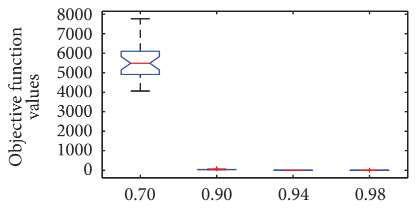

(j) Six-Hump Camel-Back function

Figure 7: The benchmark functions landscape where the value of $N = 2$ (2 dimensions).

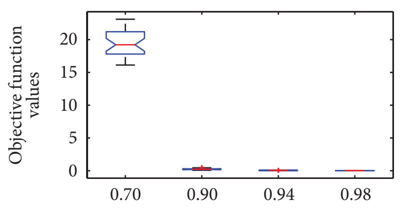
variable $\acute{x}_1$ is randomly *selected* from the historical values, $\{x_1^1, x_1^2, \ldots, x_1^m\}$, stored individuals in the set ξ_r. The other decision variables, $(x'_2, x'_3, \ldots, x'_N)$, are sequentially assigned in a similar way with a probability of HMCR, where HMCR $\in (0, 1)$.

It is worth mentioning that cellular memory consideration is able to control the diffusion between the individuals in HM, and thus, it is able to preserve the cHS diversity as long as the search process is iterated. By this strategy, the population is structured and it is possible to improve the numerical behavior of the cHS algorithm.

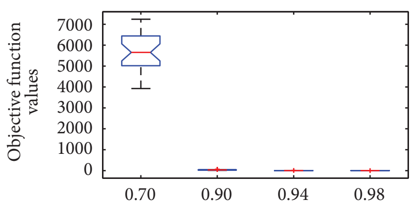
Step 6 (update the harmony memory). This step is modified in cHS algorithm. The worst individual from the set of neighbors ξ_r, that is, $(\mathbf{x}^{\text{worst}} \mid \mathbf{x}^{\text{worst}} \in \xi_r \wedge f(\mathbf{x}^{\text{worst}}) \geq f(\mathbf{x}^i), \forall i \in \{1, \ldots, m\})$, is replaced by the new individual $\acute{x}$, if better. Note that the replacement process is done taking into account the neighboring individuals in the set ξ_r only.



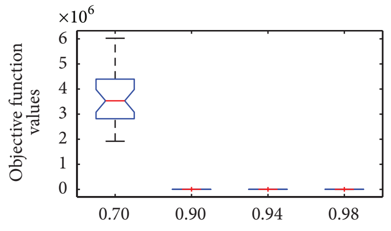
(a) 2D sphere function



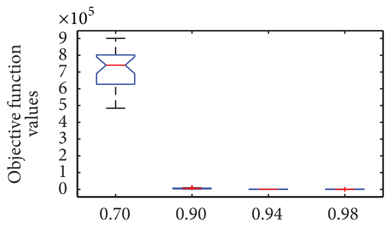
(b) 2D Schwefel's problem 2.22



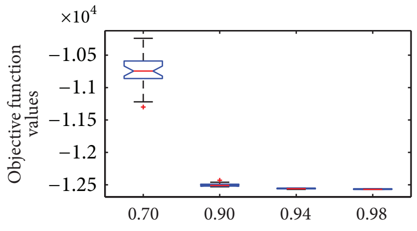
(c) 2D step function



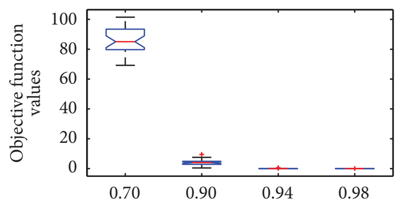
(d) 2D Rosenbrock function



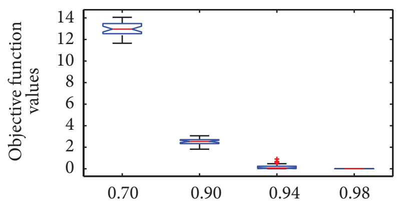
(e) 2D Rotated hyper-ellipsoid function



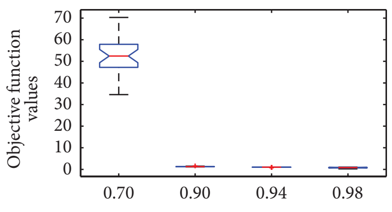
(f) 2D Schwefel's problem 2.26 function



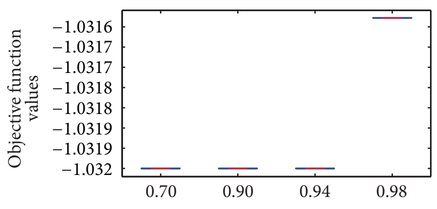
(g) 2D Rastrigin function



(h) 2D Ackley function



(i) 2D Griewank function



(j) Six-Hump Camel-Back function

Figure 8: The box plots for showing the effect varying HMCR values using the ten global optimization functions.

Step 7 (check stopping criterion). In this step, the cHS will stop if the maximum number of the iteration (i.e., NI) is reached; otherwise the algorithm repeats Steps 4 to 6. The pseudocode of cHS is demonstrated in Algorithm 1.

4. Experimental and Comparative Evaluation

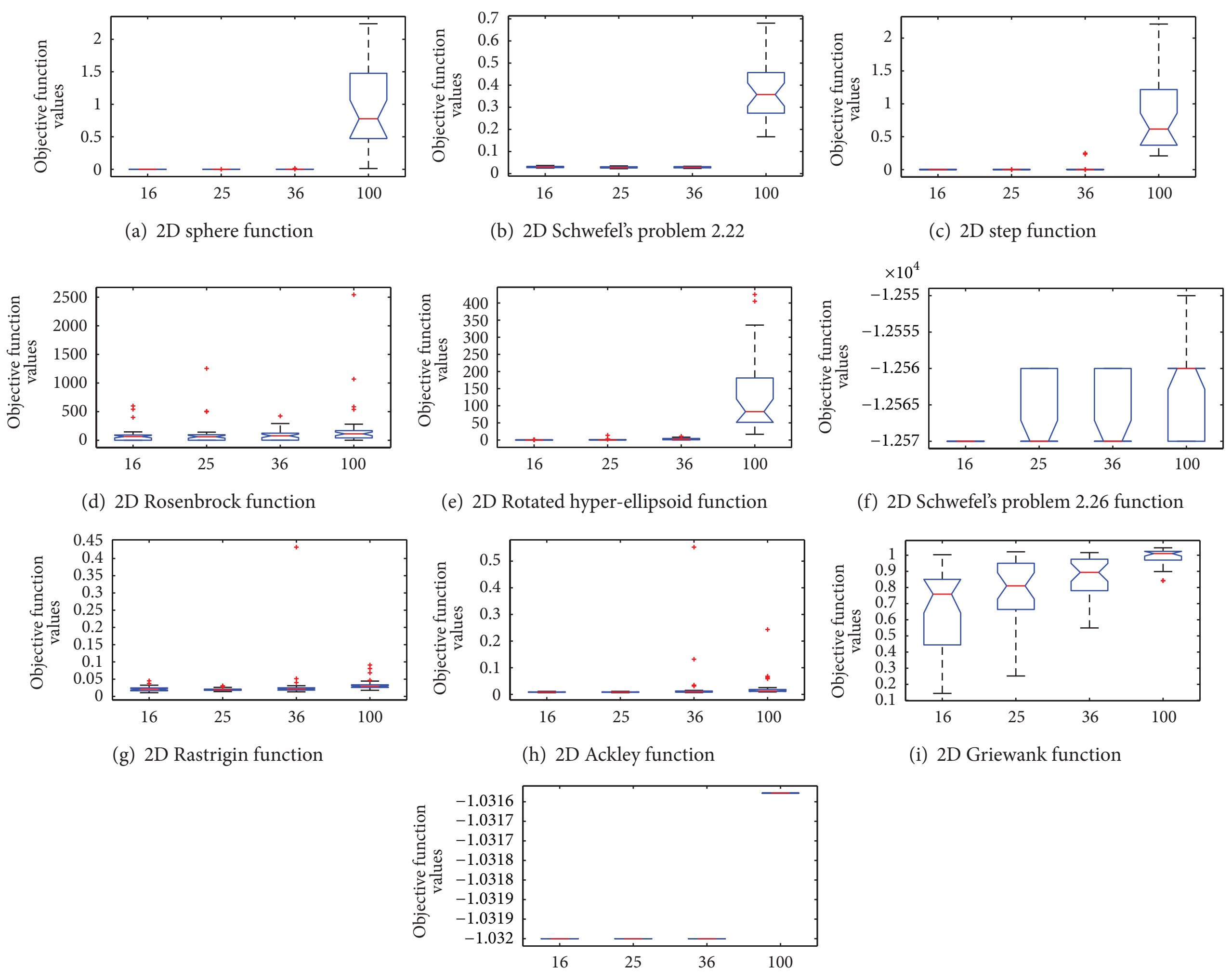
In this section, cHS algorithm is evaluated using benchmark functions circulated in the literature used to evaluate different variations of HS algorithm. The comparative evaluation is demonstrated while the sensitivity analysis of the control parameters of the proposed method is carried out.

4.1. Comparative Evaluation. In this section, a set of test functions designed for the special session on real-parameter optimization organized in the 2005 IEEE Congress on Evolutionary Computation (CEC 2005) [30], is used. The CEC 2005 comprises 25 test functions including 5 unimodals and 20 multimodals functions, as shown in Table 1. Note that a full discussion about these functions can be taken from Sunganthan et al. [30]. The CEC 2005 provides a suitable number of comparative methods in which the proposed method can be evaluated, as abbreviated in Table 2.

The experiments done followed the conditions of CEC 2005 [30], where the 25 repeated runs have been performed for each test function. The 25 runs have been summarized in terms of average of the error (AE) of the best individual (i.e., AE $= |f(\mathbf{x}^*) - f(\mathbf{x}^{\text{best}})|$). Note that $\mathbf{x}^*$ is a given optimal solution while the $\mathbf{x}^{\text{best}}$ is the average best solution obtained in 25 runs. The dimension $N = 10$ and the cHS is iterated 100,000 evaluations of the fitness function.

Notably, most of the winner comparative methods are hybrid versions of a particular EA, where their results are very efficient to the tested functions. It is appeared that the AE of HS and cHS are very close or sometimes better than those achieved by the comparative methods. In particular, the HS algorithm is able to achieve very powerful results for most test functions and excels some of the best results reported by the comparative methods. For example, HS has achieved the smallest AE for f_1, f_2, f_9, f_{14}, and f_{23}.

It is noted that the AE obtained by cHS is not the best in most cases. This is because the main idea of proposing



(a) 2D sphere function

(b) 2D Schwefel's problem 2.22

(c) 2D step function

(d) 2D Rosenbrock function

(e) 2D Rotated hyper-ellipsoid function

(f) 2D Schwefel's problem 2.26 function

(g) 2D Rastrigin function

(h) 2D Ackley function

(i) 2D Griewank function

(j) Six-Hump Camel-Back function

FIGURE 9: The box plots for showing the effect varying HMS values using the ten global optimization functions.

TABLE 2: Key to CEC'2005 comparative methods.

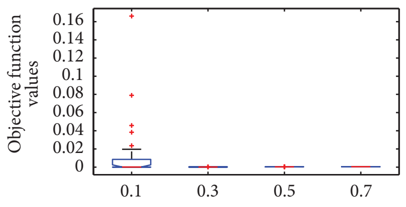
Key	Method name	Reference
BLXGL50	Hybrid real-coded genetic algorithms with female and male differentiation	[31]
BLX-MA	Adaptive local search parameters for real-coded memetic algorithms	[32]
CoEVO	Real-parameter optimization using the mutation step coevolution	[33]
DE	Real-parameter optimization with differential evolution	[34]
DMS-L-PSO	Dynamic multiswarm particle swarm optimizer with local search	[35]
EDA	Experimental results for the special session on real-parameter optimization at CEC'2005 : a simple, continuous EDA	[36]
K-PCX	A population-based, steady-state procedure for real-parameter optimization	[37]
G-CMA-ES	A restart CMA evolution strategy with increasing population size	[38]
L-CMA-ES	Performance evaluation of an advanced local search evolutionary algorithm	[39]
L-SaDE	Self-adaptive differential evolution algorithm	[40]
SPC-PNX	Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX	[41]

Table 3: Average error rate obtained in CEC'2005 special session in dimension 10.

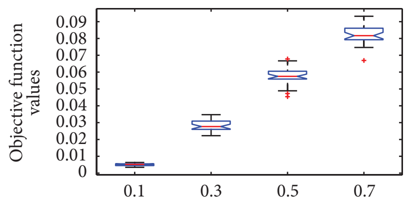
Algorithm	*F*1	*F*2	*F*3	*F*4	*F*5	*F*6	*F*7	*F*8	*F*9
BLX-GL50	1.00*E* − 009	1.00*E* − 009	5.71*E* + 002	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.17*E* − 002	2.04*E* + 001	1.15*E* + 000
BLX-MA	1.00*E* − 009	1.00*E* − 009	4.77*E* + 004	2.00*E* − 008	2.12*E* − 002	1.49*E* + 000	1.97*E* − 001	2.02*E* + 001	4.38*E* − 001
COEVO	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	2.13*E* + 000	1.25*E* + 001	3.71*E* − 002	2.03*E* + 001	1.92*E* + 001
DE	1.00*E* − 009	1.00*E* − 009	1.94*E* − 006	1.00*E* − 009	1.00*E* − 009	1.59*E* − 001	1.46*E* − 001	2.04*E* + 001	9.55*E* − 001
DMS-L-PSO	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.89*E* − 003	1.14*E* − 006	6.89*E* − 008	4.52*E* − 002	2.00*E* + 001	1.00*E* − 009
EDA	1.00*E* − 009	1.00*E* − 009	2.12*E* + 001	1.00*E* − 009	1.00*E* − 009	4.18*E* − 002	4.20*E* − 001	2.03*E* + 001	5.42*E* + 000
IPOP-CMA-ES	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	2.00*E* + 001	2.39*E* − 001
K-PCX	1.00*E* − 009	1.00*E* − 009	4.15*E* − 001	7.94*E* − 007	4.85*E* + 001	4.78*E* − 001	2.31*E* − 001	2.00*E* + 001	1.19*E* − 001
LR-CMA-ES	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	1.76*E* + 006	1.00*E* − 009	1.00*E* − 009	1.00*E* − 009	2.00*E* + 001	4.49*E* + 001
L-SADE	1.00*E* − 009	1.00*E* − 009	1.67*E* − 005	1.42*E* − 005	1.23*E* − 002	1.20*E* − 008	1.99*E* − 002	2.00*E* + 001	1.00*E* − 009
SPC-PNX	1.00*E* − 009	1.00*E* − 009	1.08*E* + 005	1.00*E* − 009	1.00*E* − 009	1.89*E* + 001	8.26*E* − 002	2.10*E* + 001	4.02*E* + 000
HS	1.00*E* − 009	1.00*E* − 009	5.20*E* + 004	9.63*E* + 000	1.40*E* − 003	5.42*E* − 002	1.82*E* − 001	2.02*E* + 001	1.00*E* − 009
cHS	1.00*E* − 009	1.00*E* − 009	1.49*E* + 007	4.66*E* + 003	1.33*E* + 004	5.84*E* + 007	7.85*E* + 001	2.03*E* + 001	2.69*E* + 001

Algorithm	*F*10	*F*11	*F*12	*F*13	*F*14	*F*15	*F*16	*F*17	*F*18
BLX-GL50	4.97*E* + 000	2.33*E* + 000	4.07*E* + 002	7.50*E* − 001	2.17*E* + 000	4.00*E* + 002	9.35*E* + 001	1.09*E* + 002	4.20*E* + 002
BLX-MA	5.64*E* + 000	4.56*E* + 000	7.43*E* + 001	7.74*E* − 001	2.03*E* + 000	2.70*E* + 002	1.02*E* + 002	1.27*E* + 002	8.03*E* + 002
COEVO	2.68*E* + 001	9.03*E* + 000	6.05*E* + 002	1.14*E* + 000	3.71*E* + 000	2.94*E* + 002	1.77*E* + 002	2.12*E* + 002	9.02*E* + 002
DE	1.25*E* + 001	8.47*E* − 001	3.17*E* + 001	9.77*E* − 001	3.45*E* + 000	2.59*E* + 002	1.13*E* + 002	1.15*E* + 002	4.00*E* + 002
DMS-L-PSO	3.62*E* + 000	4.62*E* + 000	2.40*E* + 000	3.69*E* − 001	2.36*E* + 000	4.85*E* + 000	9.48*E* + 001	1.10*E* + 002	7.61*E* + 002
EDA	5.29*E* + 000	3.94*E* + 000	4.42*E* + 002	1.84*E* + 000	2.63*E* + 000	3.65*E* + 002	1.44*E* + 002	1.57*E* + 002	4.83*E* + 002
IPOP-CMA-ES	7.96*E* − 002	9.34*E* − 001	2.93*E* + 001	6.96*E* − 001	3.01*E* + 000	2.28*E* + 002	9.13*E* + 001	1.23*E* + 002	3.32*E* + 002
K-PCX	2.39*E* − 001	6.65*E* + 000	1.49*E* + 002	6.53*E* − 001	2.35*E* + 000	5.10*E* + 002	9.59*E* + 001	9.73*E* + 001	7.52*E* + 002
LR-CMA-ES	4.08*E* + 001	3.65*E* + 000	2.09*E* + 002	4.94*E* − 001	4.01*E* + 000	2.11*E* + 002	1.05*E* + 002	5.49*E* + 002	4.97*E* + 002
L-SADE	4.97*E* + 000	4.89*E* + 000	4.50*E* − 007	2.20*E* − 001	2.92*E* + 000	3.20*E* + 001	1.01*E* + 002	1.14*E* + 002	7.19*E* + 002
SPC-PNX	7.30*E* + 000	1.91*E* + 000	2.60*E* + 002	8.38*E* − 001	3.05*E* + 000	2.54*E* + 002	1.10*E* + 002	1.19*E* + 002	4.40*E* + 002
HS	5.97*E* + 000	1.47*E* + 000	2.67*E* + 005	5.93*E* − 002	1.95*E* + 000	1.00*E* − 009	1.01*E* + 002	1.06*E* + 002	8.81*E* + 002
cHS	6.17*E* + 001	9.74*E* + 000	1.28*E* + 004	3.90*E* + 000	3.97*E* + 000	3.476*E* + 02	2.73*E* + 002	2.89*E* + 002	8.85*E* + 002

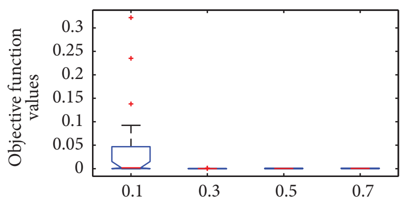
Algorithm	*F*19	*F*20	*F*21	*F*22	*F*23	*F*24	*F*25
BLX-GL50	4.49*E* + 002	4.46*E* + 002	6.89*E* + 002	7.59*E* + 002	6.39*E* + 002	2.00*E* + 002	4.04*E* + 002
BLX-MA	7.63*E* + 002	8.00*E* + 002	7.22*E* + 002	6.71*E* + 002	9.27*E* + 002	2.24*E* + 002	3.96*E* + 002
COEVO	8.45*E* + 002	8.63*E* + 002	6.35*E* + 002	7.79*E* + 002	8.35*E* + 002	3.14*E* + 002	2.57*E* + 002
DE	4.20*E* + 002	4.60*E* + 002	4.92*E* + 002	7.18*E* + 002	5.72*E* + 002	2.00*E* + 002	9.23*E* + 002
DMS-L-PSO	7.14*E* + 002	8.22*E* + 002	5.36*E* + 002	6.92*E* + 002	7.30*E* + 002	2.24*E* + 002	3.66*E* + 002
EDA	5.64*E* + 002	6.52*E* + 002	4.84*E* + 002	7.71*E* + 002	6.41*E* + 002	2.00*E* + 002	3.73*E* + 002
IPOP-CMA-ES	3.26*E* + 002	3.00*E* + 002	5.00*E* + 002	7.29*E* + 002	5.59*E* + 002	2.00*E* + 002	3.74*E* + 002
K-PCX	7.51*E* + 002	8.13*E* + 002	1.05*E* + 003	6.59*E* + 002	1.06*E* + 003	4.06*E* + 002	4.06*E* + 002
LR-CMA-ES	5.16*E* + 002	4.42*E* + 002	4.04*E* + 002	7.40*E* + 002	7.91*E* + 002	8.65*E* + 002	4.42*E* + 002
L-SADE	7.05*E* + 002	7.13*E* + 002	4.64*E* + 002	7.35*E* + 002	6.64*E* + 002	2.00*E* + 002	3.76*E* + 002
SPC-PNX	3.80*E* + 002	4.40*E* + 002	6.80*E* + 002	7.49*E* + 002	5.76*E* + 002	2.00*E* + 002	4.06*E* + 002
HS	7.54*E* + 002	8.00*E* + 002	8.53*E* + 002	7.42*E* + 002	5.59*E* + 002	2.36*E* + 002	3.29*E* + 002
cHS	1.09*E* + 003	1.08*E* + 003	1.09*E* + 003	9.00*E* + 002	1.30*E* + 003	1.06*E* + 003	8.84*E* + 002



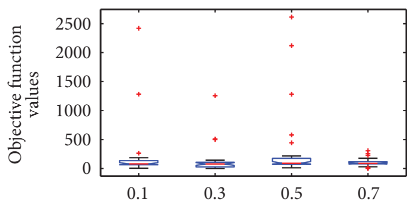
(a) 2D sphere function.



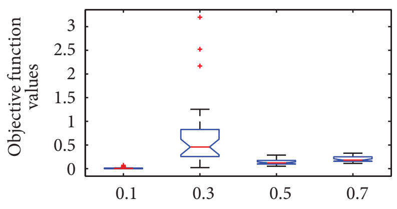
(b) 2D Schwefel's problem 2.22



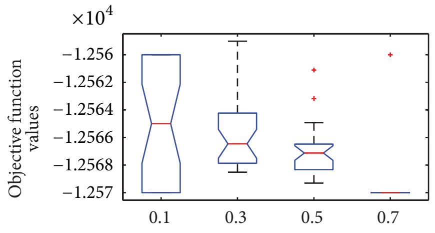
(c) 2D step function



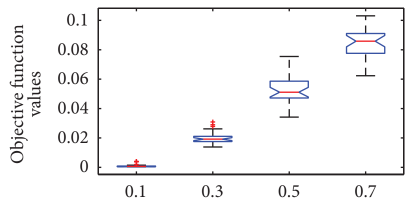
(d) 2D Rosenbrock function



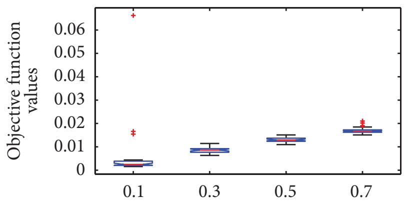
(e) 2D Rotated hyper-ellipsoid function



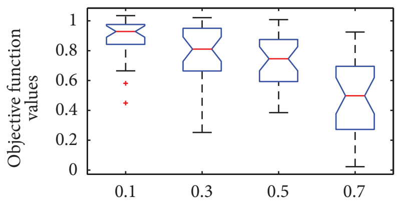
(f) 2D Schwefel's problem 2.26 function



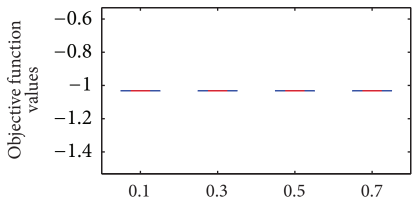
(g) 2D Rastrigin function



(h) 2D Ackley function



(i) 2D Griewank function



(j) Six-Hump Camel-Back function

FIGURE 10: The box plots for showing the effect varying PAR values using the ten global optimization functions.

the cHS is to ensure a high-level of diversity, and thus, the cHS required a higher number of evaluations to hit better results. However, the results obtained by cHS are very close to the winner results.

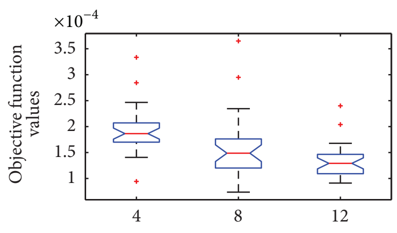
4.2. Sensitivity Analysis of cHS Parameters. All the experiments are run using a computer with 2.66 Intel Core 2 Quad with 4 GB of RAM. The operating system used is Microsoft windows Vista Enterprise Service Pack 1. The source code is implemented using MATLAB Version 7.6.0.324(R2008a).

The common parameters among all algorithms used in the experiments are set based on empirical guidelines [2]. For the sake of studying the effect of different parameter settings, in general, the parameters setting used for evaluating the cHS method are as follows: HMS = 25, HMCR = 0.98, PAR = 0.3, $N = 30$, NH = 9, and NI = 5×10^4. The neighborhood shape used is C9, presented in Figure 5.

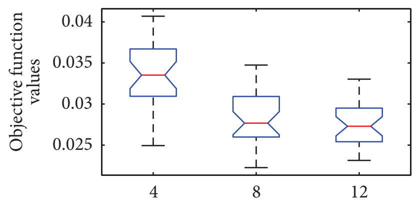
All functions are implemented in 30 dimensions (30D). For the scalability study in Section 4.2.6, the functions are implemented in 10 dimensions (10D), 50 dimensions (50D), and 100 dimensions (100D). The stopping criterion used is the maximum number of improvisation (NI).

All the results in this section are presented in Tables 4 to 8, demonstrating the mean and standard deviation for thirty independent runs. The best solution has been highlighted in bold font. A comparative analysis between cHS algorithm and the original variant of HS for the common parameters is also conducted.

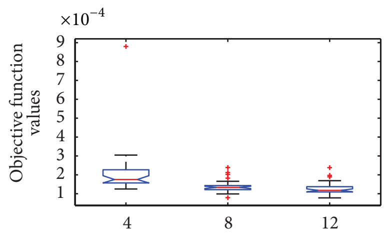
4.2.1. Benchmark Functions. The global minimization benchmark functions are used to study the sensitivity analysis of the parameters of the proposed method (cHS) against the original version of HS algorithm. Five functions are defined by Whitley et al. [42] and the other five were described by Yao et al. [43]. These functions provide a balance between unimodal and multimodal functions. These functions are commonly used to evaluate the state-of-the-art variations of harmony search algorithms [5, 44, 45].



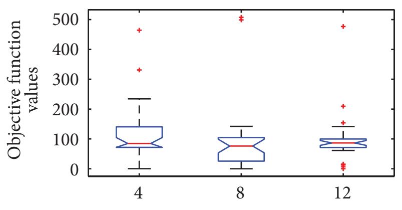
(a) 2D sphere function



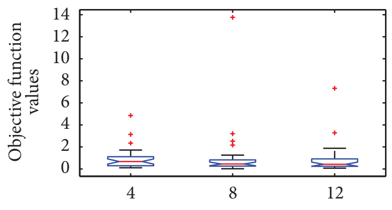
(b) 2D Schwefel's problem 2.22



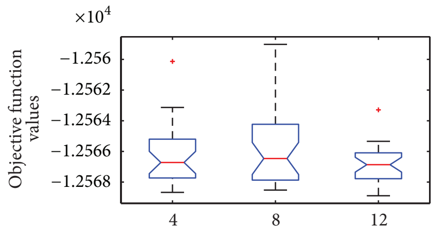
(c) 2D step function



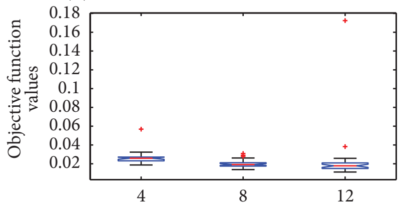
(d) 2D Rosenbrock function



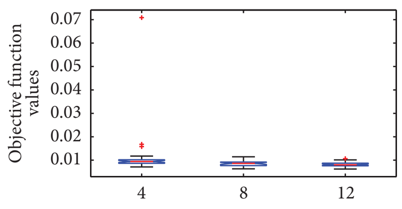
(e) 2D Rotated hyper-ellipsoid function



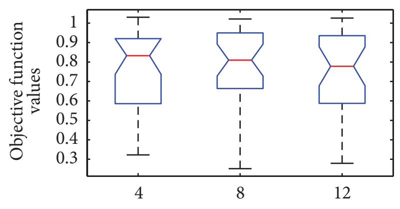
(f) 2D Schwefel's problem 2.26 function



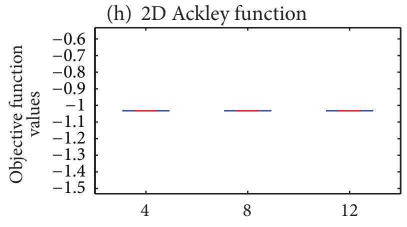
(g) 2D Rastrigin function



(h) 2D Ackley function



(i) 2D Griewank function



(j) Six-Hump Camel-Back function

Figure 11: The box plots for showing the effect varying NH values using the ten global optimization functions.

Most of the benchmark functions have standard solution space range of the objective function. Otherwise, unsymmetrical initialization ranges are used for these functions whose global optima are at the center of the solution space. These benchmark functions are as follows.

(1) *Sphere* function, defined as

$$f_1(x) = \sum_{i=1}^{N_d} x_i^2, \tag{9}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(a)).

(2) *Schwefel's* problem 2.22, defined as

$$f_2(x) = \sum_{i=1}^{N_d} |x_i| + \prod_{i=1}^{N_d} |x_i|, \tag{10}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-10 \le x_i \le 10$) (Figure 7(b)).

(3) *Step* function, defined as

$$f_3(x) = \sum_{i=1}^{N_d} \left(\lfloor x_i + 0.5 \rfloor\right)^2, \tag{11}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(c)).

(4) *Rosenbrock* function, defined as

$$f_4(x) = \sum_{i=1}^{N_d-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right), \tag{12}$$

where global optimum $x^* = (1, 1, \ldots, 1)$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(d)).

(5) *Rotated hyper-ellipsoid* function, defined as

$$f_5(x) = \sum_{i=1}^{N_d} \left(\sum_{j=1}^{i} x_j\right)^2, \tag{13}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-100 \le x_i \le 100$) (Figure 7(e)).

```
Set HMCR, PAR, NI, HMS, bw.
$x_i^j = LB_i + U(0,1) \times (UB_i - LB_i), \forall i = 1,2,\ldots,N, \forall j = 1,2,\ldots,\text{HMS}, U(0,1);$
Calculate $(f(\acute{x}_i)), \forall j = 1,2,\ldots,\text{HMS};$
Generate(**NM**) {generate the neighborhood matrix}
**While** !*StopCondition*() **do**
    select random individual $\mathbf{x}^r$
    $\acute{x} = \varnothing;$
    **for** $i = 1,\ldots,N$ **do**
        **if** $U(0,1) \le \text{HMS}$ **then**
            $\acute{x}^i \in \xi_r;$ {Memory Consideration}
            **if** $U(0,1) \le \text{PAR}$ **then**
                $\acute{x}^i = \acute{x}^i + U(-1,1) \times bw;$ {Pitch Adjustment}
            **end if**
        **else**
            $\acute{x}^i = LB_i + U(0,1) \times (UB_i - LB_i);$ {Random Consideration}
        **end if**
    **end for**
    find($x^{\text{worst}}$) where $x^{\text{worst}} \in \xi_r$
    **if** $f(\acute{x}) < f(x^{\text{worst}})$ **then**
        include $\acute{x}$ to **HM**;
        exclude $\acute{x}^{\text{worst}}$ from **HM**;
    **end if**
**end while**
```

Algorithm 1: The pseudocode of cellular harmony search (cHS).

(6) *Generalized Schwefel's* problem 2.26, defined as

$$f_6(x) = -\sum_{i=1}^{N_d} \left(x_i \sin \sqrt{(|x_i|)} \right), \tag{14}$$

where global optimum $x^* = (420.9687, 420.9687, \ldots, \ldots, \ldots, 420.9687)$ and $f(x^*) = -12569.5$ (search range $-500 \le x_i \le 500$) (Figure 7(f)).

(7) *Rastrigin* function, defined as

$$f_7(x) = \sum_{i=1}^{N_d} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right), \tag{15}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-5.12 \le x_i \le 5.12$) (Figure 7(g)).

(8) *Ackley's* function, defined as

$$f_8(x) = -20 \exp -0.2 \sqrt{\left(\frac{1}{30} \sum_{i=1}^{N_d} x_i^2 \right)} - \exp\left(\frac{1}{30} \sum_{i=1}^{N_d} \cos(2\pi x_i) \right) + 20 + e, \tag{16}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-32 \le x_i \le 32$) (Figure 7(h)).

(9) *Griewank* function, defined as

$$f_9(x) = \frac{1}{4000} \sum_{i=1}^{N_d} x_i^2 - \prod_{i=1}^{N_d} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1, \tag{17}$$

where global optimum $x^* = 0$ and $f(x^*) = 0$ (search range $-600 \le x_i \le 600$) (Figure 7(i)).

(10) *Six-Hump Camel-Back* function, defined as

$$f_{10}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \tag{18}$$

where global optimum $x = (-0.08983, 0.7126)$, $(-0.08983, 0.7126)$ and $f(x^*) = 1.0316285$ (search range $-5 \le x_i \le 5$) (Figure 7(j)).

Figure 7 depicts the corresponding landscape of the search space of each function [46].

4.2.2. The Effect of HMCR. Table 4 summarizes the effect of HMCR on the performance of cHS and the original version of HS using four values of HMCR (i.e., 0.7, 0.9, 0.94, and 0.98). The best solution obtained at each corresponding value is highlighted in bold font. Figure 8 is the box plot visualizing the effect of various values of HMCR on the behavior of cHS algorithm using the ten global minimization functions.

The results show that increasing the HMCR value improves the performance of the cHS for all functions, except six-hump where the opposite function is true. Where a small value is used, HMCR increases the diversity and hence prevents the cHS from convergence (i.e., it results in random search). Thus, it is generally better to use a large value for the HMCR (i.e., ≤0.98).

The high value of HMCR means high probability of using the harmony memory that leads to less exploration of search space. Using a probability of HMCR close to 1 (high value) might lead the algorithm to fall into the local minima. Using less probability of HMCR allows more randomly generated

Table 4: The effect of varying the HMCR parameter on HS and cHS for ten functions (f_1–f_{10}).

Function	Alg.	0.7	0.9	0.94	0.98
Sphere	cHS	5.514E + 03	3.383E + 01	2.240E + 00	**1.323E − 04**
		(8.887E + 02)	(1.164E + 01)	(9.216E − 01)	**(3.285E − 05)**
	HS	3.750E + 03	1.663E + 01	1.152E + 00	6.086E − 03
		(6.455E + 02)	(4.716E + 00)	(7.069E − 01)	(1.406E − 03)
Schwefel's 2.22	cHS	1.939E + 01	2.584E − 01	5.397E − 02	**2.738E − 02**
		(1.966E + 00)	(1.006E − 01)	(8.303E − 03)	**(2.542E − 03)**
	HS	1.581E + 01	5.359E + 00	4.081E − 01	2.20087E − 01
		(2.380E + 00)	(1.036E + 00)	(4.914E − 01)	(2.49573E − 02)
Step	cHS	5.700E + 03	3.233E + 01	2.437E + 00	**1.288E − 04**
		(8.707E + 02)	(1.085E + 01)	(1.132E + 00)	**(3.421E − 05)**
	HS	3.949E + 03	1.605E + 01	8.041E + 00	6.510E − 03
		(6.671E + 02)	(4.439E + 00)	(5.844E − 01)	(1.019E − 03)
Rosenbrock	cHS	3.578E + 06	1.276E + 03	1.944E + 02	**9.892E + 01**
		(9.728E + 05)	(9.524E + 02)	(1.177E + 02)	**(8.330E + 01)**
	HS	2.143E + 06	5.444E + 02	1.449E + 02	1.489E + 02
		(5.423E + 05)	(2.240E + 02)	(7.499E + 01)	(1.22097E + 02)
Rotated hyper	cHS	7.206E + 05	5.588E + 03	4.250E + 01	**8.421E − 01**
		(1.122E + 05)	(2.071E + 03)	(2.364E + 02)	**(1.381E + 00)**
	HS	5.155E + 05	2.972E + 03	1.373E + 02	2.647E + 00
		(7.586E + 04)	(9.061E + 02)	(1.137E + 02)	(1.531E + 00)
Schwefel's 2.26	cHS	−1.075E + 04	−1.250E + 04	−1.256E + 04	**−1.257E + 04**
		(2.536E + 02)	(2.208E + 01)	(3.982E + 00)	(2.371E + 00)
	HS	−1.123E + 04	−1.253E + 04	−1.256E + 04	**−1.257E + 04**
		(2.643E + 02)	(1.389E + 01)	(2.465E + 00)	**(2.131E + 00)**
Rastrigin	cHS	8.581E + 01	4.029E + 00	7.068E − 02	2.380E − 02
		(8.587E + 00)	(1.780E + 00)	(9.444E − 02)	(2.85250E − 02)
	HS	6.612E + 01	1.548E + 00	1.289E − 01	**1.582E − 02**
		(8.693E + 00)	(1.198E + 00)	(3.037E − 01)	**(7.023E − 03)**
Ackley	cHS	1.295E + 01	2.522E + 00	1.891E − 01	**3.890E − 02**
		(6.804E − 01)	(3.160E − 01)	(2.418E − 01)	**(1.686E − 01)**
	HS	1.209E + 01	1.830E + 00	6.493E − 01	1.060E − 01
		(5.334E − 01)	(3.862E − 01)	(1.282E − 01)	(5.763E − 02)
Griewank	cHS	5.234E + 01	1.298E + 00	1.024E + 00	7.870E − 01
		(8.732E + 00)	(1.066E − 01)	(3.995E − 02)	(2.086E − 01)
	HS	3.895E + 01	1.166E + 00	1.013E + 00	**6.566E − 01**
		(6.677E + 00)	(5.312E − 02)	(1.920E − 02)	**(2.352E − 01)**
Six-Hump	cHS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		(2.882E − 10)	(2.998E − 11)	(9.912E − 11)	(4.150E − 11)
	HS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		(1.291E − 10)	(5.571E − 11)	(3.175E − 11)	(2.922E − 09)

solutions. Therefore, the diversity increases in a way that prevents the convergence. The results reveal that cHS and HS have identical sensitivity to the different values of HMCR for all functions, and in 0.98 probabilities to use HMCR they get the best results for the majority of optimization functions. Furthermore, the results produced by cHS are better than those produced by HS in almost all tested functions.

4.2.3. The Effect of HMS. Table 5 summarizes the effect of HMS on the performance of cHS and basic HS using four values of HMS (i.e., 16, 25, 36, and 100). The best solution obtained at each corresponding value is highlighted in bold font. Figure 9 is the box plot visualizing the effect of various values of HMS on the behavior of cHS algorithm using the ten global minimization functions.

It is revealed that both cHS and the basic HS are not sensitive to the HMS. For (f_4, f_6, f_{10}), cHS obtained the best results when the value of HMS is large (see Figure 9). Apparently, with different values of HMS, cHS performance is better for most of the functions because the overlap of

Table 5: The effect of varying the HMS parameter on HS and cHS for ten functions (f_1–f_{10}).

Function	Alg.	16	25	36	100
Sphere	cHS	$1.188E-04$	$1.323E-04$	$8.506E-04$	$6.950E-01$
		$(2.204E-05)$	$(3.285E-05)$	$(2.744E-03$	$(5.385E-01)$
	HS	$\mathbf{7.961E-05}$	$6.086E-03$	$3.934E-03$	$7.274E-01$
		$\mathbf{(1.511E-05)}$	$(1.40654E-03)$	$(1.721E-02)$	$(5.510E-01)$
Schwefel's 2.22	cHS	$3.011E-02$	$2.738E-02$	$2.869E-02$	$3.850E-01$
		$(3.522E-03)$	$(2.542E-03)$	$(3.043E-03)$	$(1.391E-01)$
	HS	$\mathbf{2.208E-02}$	$2.200E-01$	$2.447E-02$	$3.210E-02$
		$\mathbf{(2.658E-03)}$	$(2.495E-02)$	$(5.482E-03)$	$(3.840E-03)$
Step	cHS	$1.225E-04$	$1.288E-04$	$1.637E-02$	$7.698E-01$
		$(2.373E-05)$	$(3.421E-05)$	$(6.139E-02)$	$(5.098E-01)$
	HS	$\mathbf{7.988E-05}$	$6.510E-03$	$3.161E-03$	$8.057E-01$
		$\mathbf{(1.123E-05)}$	$(1.019E-03)$	$(9.481E-03)$	$(4.692E-01)$
Rosenbrock	cHS	$1.106E+02$	$1.328E+02$	$1.025E+02$	$2.738E+02$
		$(1.449E+02)$	$(2.425E+02)$	$(8.530E+01)$	$(4.751E+02)$
	HS	$8.418E+02$	$\mathbf{9.892E+01}$	$1.148E+02$	$2.573E+02$
		$(7.497E+02)$	$\mathbf{(1.220E+02)}$	$(1.278E+02)$	$(4.789E+02)$
Rotated hyper	cHS	$\mathbf{2.604E-02}$	$8.421E-01$	$2.883E+00$	$1.649295E+01$
		$\mathbf{(3.042E-02)}$	$(1.381E+00)$	$(2.572E+00)$	$(2.234042E+01)$
	HS	$1.661E-01$	$2.647E+00$	$1.808E+00$	$1.259E+02$
		$(1.835E-01)$	$(1.531E+00)$	$(4.377E+00)$	$(1.060E+02)$
Schwefel's 2.26	cHS	$-1.075E+04$	$-1.256E+04$	$-1.256E+04$	$\mathbf{-1.257E+04}$
		$(1.163E+00)$	$(1.190E+00)$	$(2.122E+00)$	$(3.050E+00)$
	HS	$\mathbf{-1.257E+04}$	$\mathbf{-1.257E+04}$	$-1.256E+04$	$-1.256E+04$
		$\mathbf{(1.209E+00)}$	$(1.300E+00)$	$(2.125E+00)$	$(2.263E+00)$
Rastrigin	cHS	$\mathbf{2.159E-02}$	$2.380E-02$	$3.625E-02$	$1.010E-01$
		$\mathbf{(7.348E-03)}$	$(2.852E-02)$	$(7.533E-02)$	$(2.517E-01)$
	HS	$3.268E-02$	$1.418E+01$	$5.107E-02$	$2.503E-02$
		$(1.956E-03)$	$(2.376E+00)$	$(1.816E-01)$	$(8.612E-03)$
Ackley	cHS	$\mathbf{3.495E-03}$	$3.890E-02$	$3.369E-02$	$2.636E-02$
		$\mathbf{(2.452E-03)}$	$(1.686E-01)$	$(1.006E-01)$	$(4.388E-02)$
	HS	$6.760E-03$	$1.060E-01$	$2.313E-02$	$8.757E-02$
		$(8.735E-04)$	$(5.763E-02)$	$(8.375E-02)$	$(1.395E-01)$
Griewank	cHS	$6.481E-01$	$7.437E-01$	$8.571E-01$	$9.878E-01$
		$(2.614E-01)$	$(2.086E-01)$	$(1.486E-01)$	$(5.487E-02)$
	HS	$\mathbf{5.010E-01}$	$6.566E-01$	$8.020E-01$	$9.907E-01$
		$\mathbf{(2.409E-01)}$	$(2.352E-01)$	$(1.658E-01)$	$(5.565E-02)$
Six-Hump	cHS	$-1.031E+00$	$-1.031E+00$	$-1.031E+00$	$\mathbf{-1.032E+00}$
		$(5.325E-11)$	$(4.15028E-11)$	$(6.052E-11)$	$\mathbf{(2.001E-11)}$
	HS	$-1.031E+00$	$-1.031E+00$	$-1.031E+00$	$-1.030E+00$
		$(1.897E-11)$	$(2.922E-09)$	$(4.793E-10)$	$(2.235E-10)$

the neighborhoods provides an implicit mechanism of migration to the cHS. Since the best solutions spread smoothly through the whole population, the cHS diversity in the structured population is preserved longer than in the classical version of HS algorithm.

HM is analogous to the short-term memory of a musician that is known to be small. A plausible interpretation may rely on the high number of similar harmonies within the HM when the HMS is large that leads to shortages of diversity and, hence, lead to falling into local minima. Therefore, cHS is likely to be capable of maintaining the diversity than HS with the cellular structure.

4.2.4. The Effect of PAR. Table 6 summarizes the effect of PAR on the performance of cHS and basic HS using four values of PAR (0.1, 0.3, 0.7, and 0.9). The best solution at each corresponding value is highlighted in bold font. Figure 10 is the box plot visualizing the effect of various values of PAR on the behavior of cHS algorithm using the ten global minimization functions.

Table 6: The effect of varying the PAR parameter on HS and cHS for ten functions (f_1–f_{10}).

Function	Alg.	0.1	0.3	0.7	0.9
Sphere	cHS	1.343E − 02	**1.323E − 04**	3.136E − 04	4.713E − 04
		(3.391E − 02)	(**3.285E − 05**)	(4.066E − 05)	(5.207E − 05)
	HS	2.012E − 02	6.086E − 03	3.827E − 04	5.497E − 04
		(6.098E − 02)	(1.406E − 03)	(5.318E − 05)	(7.251E − 05)
Schwefel's 2.22	cHS	4.965E − 03	2.738E − 02	5.784E − 02	2.827E − 02
		(7.512E − 04)	(2.542E − 03)	(5.463E − 03)	(3.017E − 03)
	HS	**3.518E − 03**	2.200E − 01	7.583E − 02	9.548E − 02
		(**5.837E − 04**)	(2.495E − 02)	(5.482E − 03)	(4.834E − 03)
Step	cHS	5.526E − 02	**1.288E − 04**	2.929E − 04	4.606E − 04
		(1.177E − 01)	(**3.421E − 05**)	(3.776E − 05)	(5.136268E − 05)
	HS	1.682E − 02	6.510E − 03	3.753E − 04	5.291E − 04
		(3.939E − 02)	(1.019E − 03)	(5.417E − 05)	(4.401E − 05)
Rosenbrock	cHS	2.065E + 02	9.892E + 01	3.114E + 02	1.030E + 02
		(4.747E + 02)	(8.330E + 01)	(6.121E + 02)	(6.798E + 01)
	HS	1.813E + 02	1.489E + 02	1.312E + 02	**8.301E + 01**
		(3.929E + 02)	(1.220E + 02)	(2.515E + 02)	(**5.748E + 01**)
Rotated hyper	cHS	**1.038E − 02**	8.421E − 01	1.323E − 01	1.970E − 01
		(**1.603E − 02**)	(1.381E + 00)	(5.359E − 02)	(5.845E − 02)
	HS	5.404E + 00	2.647E + 00	4.726E − 01	4.660E − 01
		(1.507E + 01)	(1.531E + 00)	(3.248E − 01)	(3.479E − 01)
Schwefel's 2.26	cHS	**−1.257E + 04**	−1.256E + 04	−1.256E + 04	**−1.257E + 04**
		(2.536E + 02)	(1.190E + 00)	(3.982E + 00)	(2.371E + 00)
	HS	**−1.257E + 04**	−1.256E + 04	**−1.257E + 04**	**−1.257E + 04**
		(2.092E + 00)	(1.393E + 00)	(**1.082E + 00**)	(1.151E + 00)
Rastrigin	cHS	**9.355E − 04**	2.380E − 02	5.248E − 02	9.029E − 02
		(**8.827E − 04**)	(2.852E − 02)	(9.304E − 03)	(3.178E − 02)
	HS	3.468E − 03	1.418E + 01	7.952E − 02	1.092E − 01
		(1.763E − 03)	(2.376E + 00)	(1.720E − 02)	(1.377E − 02)
Ackley	cHS	**5.655E − 03**	3.890E − 02	5.730E − 02	3.827E − 02
		(**1.197E − 02**)	(1.686E − 01)	(1.815E − 01)	(1.152E − 03)
	HS	(1.878E − 02)	1.060E − 01	1.585E − 02	1.804E − 02
		(8.563E − 02)	(5.763E − 02)	(1.606E − 03)	(1.168E − 01)
Griewank	cHS	8.866E − 01	7.437E − 01	7.310E − 01	4.982E − 01
		(1.306E − 01)	(2.086E − 01)	(1.826E − 01)	(2.712E − 01)
	HS	8.542E − 01	6.566E − 01	4.912E − 01	**3.853E − 01**
		(1.270E − 01)	(2.352E − 01)	(2.591E − 01)	(**2.413E − 01**)
Six-Hump	cHS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		(1.496E − 10)	(4.150E − 11)	(3.543E − 11)	(**3.197E − 11**)
	HS	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
		(1.897E − 10)	(2.922E − 09)	(4.793E − 11)	(2.235E − 10)

It seems that using a relatively small value of PAR (i.e., ≤0.5) improves the performance of the cHA and HS. Most results at large value of PAR can increase the convergence speed of HS algorithm, while a small value of PAR increases diversity in HM. On the other hand, the small value of PAR allows more exploration of the search space, and the large value of PAR leads to a lower rate of exploration, where the diversity is reduced and then the algorithm might be trapped into the local optima. It is observed that cHS is able to get the best results than the original version of HS algorithm for

TABLE 7: The effect of varying the number of neighbors on cHS for ten functions ($f_1 - f_{10}$).

Function	NH = 4	NH = 8	NH = 12
Sphere	$5.338E - 03$	$1.587E - 04$	**1.32E − 04**
	$(2.819E - 02)$	$(5.907E - 05)$	**(3.29E − 05)**
Schwefel's 2.22	$3.331E - 02$	$2.827E - 02$	**2.74E − 02**
	$(3.789E - 03)$	$(3.017E - 03)$	**(2.54E − 03)**
Step	$2.074E - 04$	$1.761E - 04$	**1.29E − 04**
	$(1.339E - 04)$	$(2.058E - 04)$	**(3.42E − 05)**
Rosenbrock	$1.169E + 02$	$1.329E + 02$	**9.89E + 01**
	$(9.558E + 01)$	$(2.425E + 02)$	**(8.33E + 01)**
Rotated hyper-ellipsoid	$1.945E + 00$	$1.119E + 00$	**8.42E − 01**
	$(5.520E + 00)$	$(2.500E + 00)$	**(1.38E + 00)**
Schwefel's 2.26	**−1.257E + 04**	**−1.257E + 04**	$-1.26E + 04$
	(1.901E + 00)	$(2.371E + 00)$	$(1.19E + 00)$
Rastrigin	$3.548E - 02$	**2.021E − 02**	$2.38E - 02$
	$(5.073E - 02)$	**(4.141E − 03)**	$(2.85E - 02)$
Ackley	$2.901E - 02$	**8.698E − 03**	$3.89E - 02$
	$(6.715E - 02)$	**(1.152E − 03)**	$(1.69E - 01)$
Griewank	$7.530E - 01$	$7.870E - 01$	**7.44E − 01**
	$(2.071E - 01)$	$(1.903E - 01)$	**(2.09E − 01)**
Six-Hump Camel-Back	**−1.032E + 00**	**−1.032E + 00**	**−1.032E + 00**
	$(5.473E - 11)$	**(2.821E − 11)**	$(4.15E - 11)$

some benchmark functions (i.e., f_4, f_2, f_8, f_9), but not so for the others.

4.2.5. The Effect Number of Neighbors NH. Table 7 summarizes the effect number of neighbors (NH) according to cellular structure on the performance of cHS. Table 7 also exhibits the effect of NH for (f_1 to f_{10}) using four values of NH (4, 8, 12), where the value of HMS = 25. The best solution at each corresponding value is highlighted in bold font. The best results of comparing cHS with original version HS are highlighted in bold. Figure 11 is the box plot visualizing the effect of various NH values on the behavior of cHS algorithm using the ten global minimization functions.

The results show that cHS obtained the best result when the number of neighbors is large, except (f_6) which obtained the best result when the number of neighbors is small. Generally, this leads to the conclusion that cHS often shows better performance at larger values of NH between 8 and 12. Figure 11 shows the box plots of the recorded results which reveal the distribution of the 30 tested runs against the various value of NH.

4.2.6. Scalability Study of N. In this section, the results produced by cHS and HS, when the dimension of the function is set to $N = 10$, $N = 30$, $N = 50$, and $N = 100$ as shown in Table 8, are recorded.

In general, decreasing the dimensionality leads to better results in cHS and HS. This comes in line with the previous theory. However, it is observed from results (Table 8) that the proposed cHS algorithm outperforms the HS when dimensionality is large for the majority of the benchmark optimization functions. This shows that increasing the dimensionality of the problem needs a better algorithm like cHS.

5. Conclusion and Future Work

In this paper, a new version of HS algorithm called cellular harmony search (cHS) algorithm is proposed. cHS is an HS algorithm embedded with cellular automata concepts. The main idea of proposing cHS algorithm is to provide a structured population that preserves a high level of diversity during the search. In cHS, the HM individuals are arranged as a two-dimensional toroidal grid, where each individual is generated and only interacts with its neighboring individuals. The operators of the original version of HS algorithm are adjusted to observe the cellular GA theory, where the concepts of cell and cell search space are employed.

Using ten global optimization functions circulated in the literature, the cHS is evaluated. The results support the theory of cellular automata, where in almost all cases the cHS outperforms HS algorithm. The sensitivity analysis of cHS parameters has suggested that the cHS is sensitive to the values of HMCR, PAR, N, and NH. The comparative evaluation is also conducted with two versions of HS algorithms proposed in [44, 45], where comparable results were obtained.

This is an initial investigation of using cellular automata concepts in the HS algorithm optimization framework.

Table 8: Mean and standard deviation of ten functions (f_1–f_{10}) when $N = (10, 30, 50, 100)$.

Function	Alg.	$N = 10$	$N = 30$	$N = 50$	$N = 100$
Sphere	cHS	$7.022E-08$	$1.323E-04$	$4.776E+02$	$2.036E+04$
		$(4.079E-08)$	$(3.285E-05)$	$(9.256E+01)$	$(2.584E+03)$
	HS	$\mathbf{3.033E-08}$	$6.086E-03$	$8.746E+02$	$1.591E+04$
		$\mathbf{(1.842E-08)}$	$(1.406E-03)$	$(1.484E+02)$	$(2.471E+03)$
Schwefel's 2.22	cHS	$3.548E-04$	$2.738E-02$	$8.810E+00$	$8.500E+01$
		$(1.327E-04)$	$(2.542E-03)$	$(9.866E-01)$	$(8.610E+00)$
	HS	$\mathbf{2.725E-04}$	$2.200E-01$	$1.050E+01$	$5.479E+01$
		$\mathbf{(6.537E-05)}$	$(2.495E-02)$	$(1.323E+00)$	$(3.951E+00)$
Step	cHS	$6.297E-08$	$1.288E-04$	$3.771E+02$	$1.960E+04$
		$(3.292E-08)$	$(3.42100E-05)$	$(1.029E+02)$	$(2.260E+03)$
	HS	$\mathbf{3.501E-08}$	$6.510E-03$	$7.214E+02$	$1.551E+04$
		$\mathbf{(2.873E-08)}$	$(1.019E-03)$	$(1.364E+02)$	$(1.169E+03)$
Rosenbrock	cHS	$3.758E+01$	$9.892E+01$	$2.090E+04$	$1.359E+07$
		$(7.344E+01)$	$(8.330E+01)$	$(9.488E+03)$	$(2.397E+06)$
	HS	$\mathbf{1.651E+01}$	$1.489E+02$	$9.093E+04$	$1.730E+07$
		$\mathbf{(3.809E+01)}$	$(1.220E+02)$	$(2.346E+04)$	$(2.195E+06)$
Rotated hyper-ellipsoid	cHS	$2.351E-06$	$8.421E-01$	$1.993E+05$	$3.097E+07$
		$(1.799E-06)$	$(1.381E+00)$	$(4.916E+04)$	$(3.408E+06)$
	HS	$\mathbf{8.554E-07}$	$2.647E+00$	$2.825E+05$	$2.723E+07$
		$\mathbf{(7.059E-07)}$	$(1.531E+00)$	$(4.035E+04)$	$(3.321E+06)$
Schwefel's 2.26	cHS	$\mathbf{-4.190E+03}$	$-1.256E+04$	$9.019E+02$	$8.874E+03$
		$(1.040E-01)$	$(1.190E+00)$	$(2.182E+02)$	$(8.167E+02)$
	HS	$\mathbf{-4.190E+03}$	$-1.256E+04$	$8.594E+02$	$5.119E+03$
		$\mathbf{(3.861E-03)}$	$(1.393E+00)$	$(1.645E+02)$	$(5.619E+02)$
Rastrigin	cHS	$8.538E-06$	$2.380E-02$	$3.594E+01$	$3.575E+02$
		$(4.752E-06)$	$(2.852E-02)$	$(4.537E+00)$	$(4.352E+01)$
	HS	$\mathbf{4.938E-06}$	$1.418E+01$	$4.766E+01$	$2.325E+02$
		$\mathbf{(5.215E-06)}$	$(2.376E+00)$	$(4.308E+00)$	$(1.703E+01)$
Ackley	cHS	$\mathbf{9.366E-06}$	$3.890E-02$	$5.168E+00$	$1.385E+01$
		$\mathbf{(4.385E-06)}$	$(1.686E-01)$	$(4.202E-01)$	$(3.524E-01)$
	HS	$2.101E-04$	$1.060E-01$	$6.197E+00$	$1.277E+01$
		$(1.146E-04)$	$(5.763E-02)$	$(3.790E-01)$	$(3.405E-01)$
Griewank	cHS	$9.958E-02$	$7.437E-01$	$5.828E+00$	$1.931E+02$
		$(4.819E-02)$	$(2.086E-01)$	$(1.398E+00)$	$(2.241E+01)$
	HS	$\mathbf{7.082E-02}$	$6.566E-01$	$8.641E+00$	$1.470E+02$
		$\mathbf{(4.053E-02)}$	$(2.352E-01)$	$(1.330E+00)$	$(1.583E+01)$
Six-Hump Camel-Back	cHS	$\mathbf{-1.032E+00}$	$-1.03163E+00$	$\mathbf{-1.032E+00}$	$\mathbf{-1.032E+00}$
		$(3.927E-11)$	$(4.150E-11)$	$\mathbf{(0)}$	$\mathbf{(0)}$
	HS	$\mathbf{-1.032E+00}$	$-1.03163E+00$	$\mathbf{-1.032E+00}$	$\mathbf{-1.032E+00}$
		$(6.526E-11)$	$(2.922E-09)$	$\mathbf{(0)}$	$\mathbf{(0)}$

The future, indeed, is pregnant with several research directions, such as

(i) analyzing the selection pressure and time complexity concepts of cHS algorithm,

(ii) studying the effect of the neighborhood shapes on the performance of cHS,

(iii) studying a new migration strategy to empower the interaction between the individuals and their neighbors.

Acknowledgment

The first author is grateful to be awarded Postdoctoral Research Fellowship from the School of Computer Sciences (USM). This paper is supported by Grant no. 203/PTS6728001 (Grant Type: LRGS).

References

[1] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

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

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

[4] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Applied Mathematics and Computation*, vol. 199, no. 1, pp. 223–230, 2008.

[5] M. A. Al-Betar, I. A. Doush, A. T. Khader, and M. A. Awadallah, "Novel selection schemes for harmony search," *Applied Mathematics and Computation*, vol. 218, no. 10, pp. 6095–6117, 2012.

[6] M. A. Al-Betar, A. T. Khader, and M. Zaman, "University course timetabling using a hybrid harmony search metaheuristic algorithm," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 42, pp. 664–681, 2012.

[7] M. A. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Annals of Operations Research*, vol. 194, no. 1, pp. 3–31, 2012.

[8] M. Awadallah, A. Khader, M. Al-Betar, and A. Bolaji, "Nurse rostering using modied harmony search algorithm," in *Swarm, Evolutionary, and Memetic Computing*, B. Panigrahi, P. Suganthan, S. Das, and S. Satapathy, Eds., vol. 7077 of *Lecture Notes in Computer Science*, pp. 27–37, Springer, Berlin, Germany, 2011.

[9] M. Awadallah, A. Khader, M. Al-Betar, and P. Woon, "Office-space-allocation problem using harmony search algorithm," in *Neural Information Processing*, T. Huang, Z. Zeng, C. Li, and C. Leung, Eds., vol. 7664 of *Lecture Notes in Computer Science*, pp. 365–374, Springer, Berlin, Germany, 2012.

[10] L. Guo, "A novel hybrid bat algorithm with harmony search for global numerical optimization," *Journal of Applied Mathematics*, vol. 2013, Article ID 696491, 21 pages, 2013.

[11] L. Zhang, Y. Xu, and Y. Liu, "An elite decision making harmony search algorithm for optimization problem," *Journal of Applied Mathematics*, vol. 2012, Article ID 860681, 15 pages, 2012.

[12] Z. W. Geem, "Economic dispatch using parameter-setting-free harmony search," *Journal of Applied Mathematics*, vol. 2013, Article ID 427936, 5 pages, 2013.

[13] J. Fourie, R. Green, and Z. W. Geem, "Generalised adaptive harmony search: a comparative analysis of modern harmony search," *Journal of Applied Mathematics*, vol. 2013, Article ID 380985, 13 pages, 2013.

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

[15] A. Mukhopadhyay, A. Roy, S. Das, S. Das, and A. Abraham, "Population-variance and explorative power of harmony search: an analysis," in *Proceedings of the 3rd International Conference on Digital Information Management (ICDIM '08)*, pp. 775–781, November 2008.

[16] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm*, Z. Geem, Ed., vol. 191 of *Studies in Computational Intelligence*, pp. 1–14, Springer, Berlin, Germany, 2009.

[17] M. A. Al-Betar, A. T. Khader, Z. W. Geem, I. A. Doush, and M. A. Awadallah, "An analysis of selection methods in memory consideration for harmony search," *Applied Mathematics and Computation*, vol. 219, no. 22, pp. 10753–10767, 2013.

[18] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 126–142, 2005.

[19] D. Whitley and T. Starkweather, "Genitor ii: a distributed genetic algorithm," *Journal of Experimental & Theoretical Articial Intelligence*, vol. 2, pp. 189–214, 1990.

[20] Y. Shi, H. Liu, L. Gao, and G. Zhang, "Cellular particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4460–4493, 2011.

[21] T. Y. Lim, "Structured population genetic algorithms: a literature survey," *Artificial Intelligence Review*, 2012.

[22] E. Alba and B. Dorronsoro, *Cellular Genetic Algorithms*, vol. 42, Springer, Berlin, Germany, 2008.

[23] S. Janson, E. Alba, B. Dorronsoro, and M. Middendorf, "Hierarchical cellular genetic algorithm," in *Evolutionary Computation in Combinatorial Optimization*, J. Gottlieb and G. Raidl, Eds., vol. 3906 of *Lecture Notes in Computer Science*, pp. 111–122, Springer, Berlin, Germany, 2006.

[24] J. Błazewicz, M. Drozdowski, F. Guinand, and D. Trystram, "Scheduling a divisible task in a two-dimensional toroidal mesh," *Discrete Applied Mathematics*, vol. 94, no. 1–3, pp. 35–50, 1999.

[25] E. Cantu-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, vol. 1, Springer, London, UK, 2000.

[26] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, no. 6-7, pp. 619–632, 1991.

[27] C. Fernandes and A. Rosa, "A study on non-random mating and varying population size in genetic algorithms using a royal road function," in *Proceedings of the Congress on Evolutionary Computation*, pp. 60–66, May 2001.

[28] J. V. Neumann, *Theory of Self-Reproducing Automata*, University of Illinois Press, Urbana, Ill, USA, 1966.

[29] B. Schönfisch and A. de Roos, "Synchronous and asynchronous updating in cellular automata," *BioSystems*, vol. 51, no. 3, pp. 123–143, 1999.

[30] P. Suganthan, N. Hansen, J. Liang et al., "Problem denitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, 2005.

[31] C. García-Martínez and M. Lozano, "Hybrid real-coded genetic algorithms with female and male differentiation," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 896–903, September 2005.

[32] D. Molina, F. Herrera, and M. Lozano, "Adaptive local search parameters for real-coded memetic algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 888–895, September 2005.

[33] P. Posik, "Real-parameter optimization using the mutation step coevolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 872–879, 2005.

[34] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 506–513, September 2005.

[35] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 522–528, September 2005.

[36] B. Yuan and M. Gallagher, "Experimental results for the special session on real-parameter optimization at CEC 2005: a simple, continuous EDA," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1792–1799, September 2005.

[37] A. Sinha, S. Tiwari, and K. Deb, "A population-based, steady-state procedure for real-parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 514–521, September 2005.

[38] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1777–1784, September 2005.

[39] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1769–1776, September 2005.

[40] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 1785–1791, September 2005.

[41] P. J. Ballester, J. Stephenson, J. N. Carter, and K. Gallagher, "Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 498–505, September 2005.

[42] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245–276, 1996.

[43] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

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

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

[46] M. A. Al-Betar, A. T. Khader, and F. Nadi, "Selection mechanisms in memory consideration for examination timetabling with harmony search," in *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference (GECCO '10)*, pp. 1203–1210, ACM, Portland, Ore, USA, July 2010.