Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 587820, 12 pages
http://dx.doi.org/10.1155/2014/587820

*Research Article*

# A Cooperative Harmony Search Algorithm for Function Optimization

**Gang Li and Qingzhong Wang**

*College of Automation, Harbin Engineering University, Harbin 150001, China*

Correspondence should be addressed to Qingzhong Wang; wqz_heu@yahoo.com

Received 12 December 2013; Accepted 26 January 2014; Published 6 March 2014

Academic Editor: Rongni Yang



Harmony search algorithm (HS) is a new metaheuristic algorithm which is inspired by a process involving musical improvisation. HS is a stochastic optimization technique that is similar to genetic algorithms (GAs) and particle swarm optimizers (PSOs). It has been widely applied in order to solve many complex optimization problems, including continuous and discrete problems, such as structure design, and function optimization. A cooperative harmony search algorithm (CHS) is developed in this paper, with cooperative behavior being employed as a significant improvement to the performance of the original algorithm. Standard HS just uses one harmony memory and all the variables of the object function are improvised within the harmony memory, while the proposed algorithm CHS uses multiple harmony memories, so that each harmony memory can optimize different components of the solution vector. The CHS was then applied to function optimization problems. The results of the experiment show that CHS is capable of finding better solutions when compared to HS and a number of other algorithms, especially in high-dimensional problems.

## 1. Introduction

Optimization is a very ancient problem which has been researched by numerous mathematicians since time immemorial, including Newton, Gauss, and John von Neumann. They developed numerous mathematical theories and methods that made a considerable contribution to optimization. However, almost all mathematical methods require first-order derivatives or second-order derivatives, or even higher derivatives. When the object function is not too complex, we can compute its derivatives easily, but unfortunately most object functions in the real world are so complex that we cannot compute the derivatives, and even worse, they may have no derivatives at all. In this case, it is very difficult to implement mathematical methods.

What should we do, then, to solve complex, nonlinear, nondifferentiable, and multimodal problems? We can perhaps learn from the nature. Many natural phenomena as well as the activities of animals provide us with inspiration and we are able to imitate these phenomena or activities in order to solve complex problems. Over the last four decades, a large number of methods inspired by nature have been developed in order to solve very complex optimization problems [1], and these were called metaheuristic algorithms. All these metaheuristic algorithms imitate natural phenomena, such as evolutionary algorithms (EAs), which include genetic algorithms (GAs) [2], evolutionary programming (EP) [3], and evolution strategy (ES) [4], all of which simulate biological evolution. PSO and ant colony optimization (ACO), for instance, mimic the foraging behavior of animals [5, 6], and simulated annealing (SA) simulates physical annealing process [7]. As a metaheuristic algorithm, HS is no exception, it is inspired by the improvisation process of musical performers [8]. Although the algorithms mentioned above imitate different phenomena, they have some common factors: (1) they all have a random process; (2) the solutions that they show us are just approximate results; (3) they all suffer from the "curse of dimensionality" [9], meaning that the search space will increase exponentially when the number of the dimensions increases. The probabilities of finding the

optimality region will thus decrease. One way of overcoming this drawback is to partition higher dimensional search space into lower dimensional subspaces.

Potter and De Jong [10, 11] have proposed a general framework for cooperative coevolutionary algorithms and then they applied the method to the GA (CCGA). They suggested that the higher dimensional solution vectors should be split into smaller vectors, with each vector just being a part of a complete solution, so that they must cooperate with each other to constitute a potential solution. In this paper, we apply Potter and De Jong's cooperative method to the HS to enhance the performance of standard HS.

The rest of this paper is organized as follows. Section 2 presents an overview of the standard HS and some other improved HS algorithms and Section 3 demonstrates the convergence of HS and explains why the improved HS performs better than the standard HS. In Section 4, the cooperative method is briefly introduced, and then the proposed CHS algorithm is elaborated. Section 5 describes the benchmark functions used to test the new algorithm and the experimental results can be found in Section 6. Finally, the conclusions are given in Section 7.

## 2. Harmony Search Algorithms

The HS algorithm, originally conceived by Geem et al. in 2001 [8], was inspired by musical improvisation. There are always three ways open to a musician [12], when he or she is improvising. The first is when he or she plays a piece of music that he or she remembers exactly; the second is when a musician plays something that is similar to what he or she remembers exactly, the musician possibly being engaged in improvisation based on the original harmony by adjusting the pitch slightly. The last one involves a composition that is new. The process employed by the musicians in order to find the best harmony is likely to be the process of optimization. In fact, the HS algorithm mimics the process used to solve optimization problems, with this algorithm being widely applied in order to solve optimization problems, including water network design [13, 14], PID controller design [15], Cluster analysis, and function optimization [16, 17]. Several approaches have been taken in order to improve the performance of the standard HS, some of which are discussed in the following subsections.

*2.1. The Standard HS (SHS).* When we use an algorithm to solve problems, firstly we must know what the problems are. Assuming that there is an unconstrained optimization problem which can be described as follows:

$$\begin{aligned} &\min \quad f(X) \\ &\text{s.t.} \quad Lx_i \le x_i \le Ux_i, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

where $f(X)$ is the object function, $X$ is the set of each decision variable, $x_i$ is the $i$th decision variable, $Lx_i$ and $Ux_i$ are the lower and upper bounds of the $i$th decision variable, and $n$ is the number of decision variables. To apply SHS to solve the optimization problem mentioned above, five steps should be taken [1] as follows.

*Step 1.* Initialize the problem algorithm parameters. The problem parameters like $n$, $Lx_i$, $Ux_i$ should be initialized in this step. In addition, four algorithm parameters should be initialized, including harmony memory size (HMS), harmony memory considering rate (HMCR), pitch adjusting rate (PAR), and the maximum number of improvisation ($T_{\max}$), or stopping criterion.

*Step 2.* Initialize the harmony memory. The SHS is similar to GAs. GAs are population based optimization algorithms, but the "population" in SHS is referred to as harmony memory (HM), which is composed of solution vectors. The structure of HM is as follows:

$$\mathrm{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_1^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_n^{\mathrm{HMS}} \end{bmatrix}, \tag{2}$$

where $x_j^i$ is the $j$th decision variable of the $i$th solution vector. Generally, we initialize the HM randomly. For $x_j^i$, it can be generated by using the following equation:

$$x_j^i = Lx_j + \left(Ux_j - Lx_j\right) \times \mathrm{rand}(), \tag{3}$$

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

*Step 3.* Improvise a new harmony. This step is the core step of SHS. A new harmony vector $X^{\mathrm{new}} = (x_1^{\mathrm{new}}, x_2^{\mathrm{new}}, \ldots, x_n^{\mathrm{new}})$ is improvised according to these three rules: (1) harmony memory consideration; (2) pitch adjustment; (3) randomization. The probabilities of harmony consideration and pitch adjustment are dependent on HMCR and PAR. For instance, the $i$th variable $x_i^{\mathrm{new}}$ of the new harmony vector can be improvised as follows:

$$x_i^{\mathrm{new}} = x_i^{\mathrm{new}} \in \left\{x_i^1, x_i^2, \ldots, x_i^{\mathrm{HMS}}\right\}, \tag{4}$$

$$x_i^{\mathrm{new}} = Lx_j + \left(Ux_i - Lx_i\right) \times \mathrm{rand}(). \tag{5}$$

In fact, before the variable is generated, we should generate a random number $r_1$ between 0 and 1, then we compare $r_1$ with HMCR, if $r_1 \le$ HMCR; (4) is used to improvise the variable; otherwise, the variable will be improvised by using (5). For example, if HMCR = 0.95, then the HS algorithm will choose a decision variable from the HM with a 95% probability. And if the variable is chosen from the HM, then it will be adjusted with probability PAR as follows:

$$x_i^{\mathrm{new}} = x_i^{\mathrm{new}} \pm \mathrm{bw} \times \mathrm{rand}() \quad \text{with probability PAR}, \tag{6}$$

where bw is an arbitrary distance bandwidth and rand( ) is a random number between 0 and 1.

*Step 4.* Update the HM. If the new improvised harmony vector is better than the worst harmony vector in the HM in

```
Initialize parameters and HM
while t < T_max do
  For i = 1 to n do
    if rand() ≤ HMCR then
      x_i^new = x_i^j, where j ~ U(1, 2, ..., HMS)
      if rand() ≤ PAR then
        x_i^new = x_i^new ± bw × rand()
      end if
    else
      x_i^new = Lx_j + (Ux_i − Lx_i) × rand()
    end if
  end for
  Update HM
end while
```

Pseudocode 1: Pseudocode for the SHS algorithm.

terms of the object function value, the worst harmony in the HM is superseded by the new harmony vector.

*Step 5.* Check stopping criterion. If the stopping criterion is satisfied, the iteration is terminated. If not, Steps 3 and 4 are repeated.

The pseudocode of SHS is shown in Pseudocode 1.

*2.2. Improved HSs.* The SHS algorithm was introduced in the last subsection, and in this subsection several improved HSs are introduced. PAR and bw are two important parameters of HS, deciding the accuracy of the solution. As the number of iteration increases, the HM becomes better and the solution is closer to the global optimal position. We should use a smaller bw to adjust the pitches, and this adjustment should be with a higher probability, but all parameters are fixed within the SHS algorithm, meaning that they cannot change. If we choose a low PAR and a narrow bw, the SHS will converge slowly, but on the other hand, if we choose a very high PAR and a wide bw, the SHS will converge fast, although the solution may scatter around some potential optimals as a random search. A dynamic adjustment strategy for parameters, especially for PAR and bw, is therefore very necessary.

A dynamic adjustment strategy for PAR and bw was proposed by Mahdavi et al. in 2007 [18]. They suggested that PAR should increase linearly and bw should index decrease. They can change with generation number as shown in Figure 1 by using the following equations:

$$\begin{aligned} \mathrm{PAR}(t) &= \mathrm{PAR}_{\min} + \frac{\mathrm{PAR}_{\max} - \mathrm{PAR}_{\min}}{T_{\max}} \times t, \\ \mathrm{bw}(t) &= \mathrm{bw}_{\max} \times \exp\left(\frac{\ln\left(\mathrm{bw}_{\min}/\mathrm{bw}_{\max}\right)}{T_{\max}} \times t\right), \end{aligned} \tag{7}$$

where PAR($t$) and bw($t$) are the pitch adjusting rate and bandwidth for each generation, $\mathrm{PAR}_{\min}$ and $\mathrm{bw}_{\min}$ are the minimum pitch adjusting rate and bandwidth, $\mathrm{PAR}_{\max}$ and $\mathrm{bw}_{\max}$ are the maximum pitch adjusting rate and bandwidth, and $t$ is the generation number. The drawback of the IHS is that we have to specify the $\mathrm{PAR}_{\min}$, $\mathrm{PAR}_{\max}$, $\mathrm{bw}_{\min}$, and $\mathrm{bw}_{\max}$, which is essentially very difficult, and we are unable to guess without experience.

The IHS algorithm merely changes the parameters dynamically, while some other improved HSs are capable of changing the search strategy of SHS, such as the global best HS (GHS) proposed by Omran and Mahdavi [19]. The GHS is inspired by the concept of swarm intelligence as proposed in PSO. In PSO, the position of each particle presents a candidate solution, so that, when a particle flies through the search space, the position of the particle is influenced by the best position itself has visited so far and the position of the best particle among the swarm, in other words, there are some similarities between the new position and the best position. In GHS, the new harmony vector can mimic the best harmony in the HM. The difference between GHS and SHS is the pitch adjustment strategy, with the variable being adjusted in SHS using (6), while in GHS it is adjusted using the following equation:

$$x_i^{\mathrm{new}} = x_k^{\mathrm{best}} \quad \text{with probability PAR}, \tag{8}$$

where best is the index of the best harmony in the HM and $k \sim U(1, n)$.

The results of the experiment show that these two improved harmony search algorithms can find better solutions when compared with SHS.

## 3. Convergence of HS Algorithm

As a metaheuristic algorithm, HS is capable of finding local optima and the global optimum, but why can it converge on local or global optima, and which operator or parameter may have effects on the speed of convergence? All of these problems are unsolved. In this section, we endeavor to solve the problems.

As we noted in the last section, HS has an operator referred to as "HM updating," the fourth step of HS. This is an indispensable step for HS, because this operator guarantees the convergence of HS, without it HS may not find even a local minimum. In order to explain this, some definitions and theorems are necessary.

*Definition 1* (monotone sequences). A sequence $\{x_n\}$ is said to be monotonically increasing provided that

$$x_{n+1} \geq x_n \tag{9}$$

for every natural number $n$.

A sequence $\{x_n\}$ is said to be monotonically decreasing provided that

$$x_{n+1} \leq x_n \tag{10}$$

for every natural number $n$.

A sequence $\{x_n\}$ is called monotone if it is either monotonically increasing or monotonically decreasing.

**Theorem 2** (the monotone convergence theorem). *A monotone sequence converges if and only if it is bounded.*

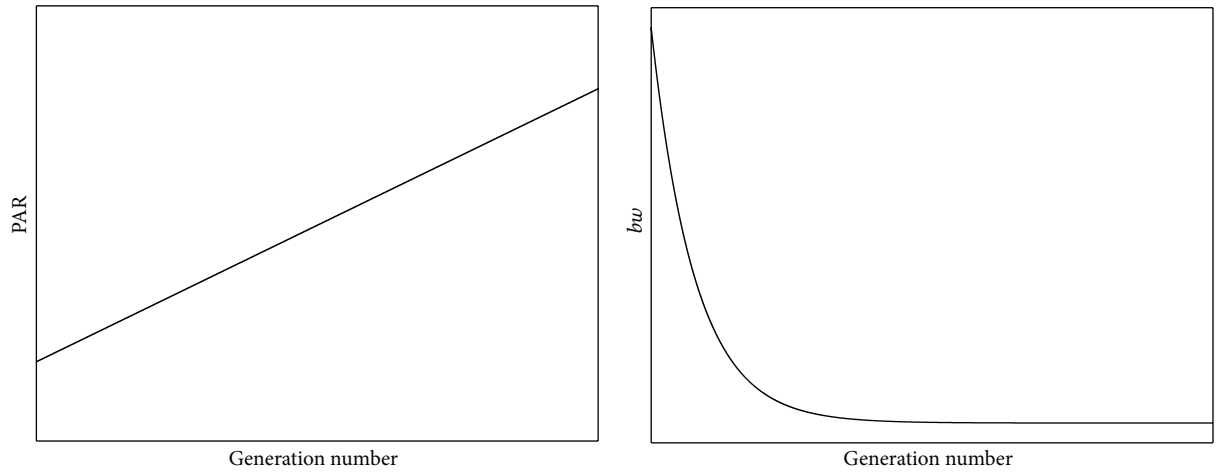


Figure 1: Variation of PAR and bw versus generation number.

*Proof.* Firstly, let us suppose that the sequence $\{x_n\}$ is a monotonically increasing sequence. Then we can define $S = \{x_n \mid n \in \mathbb{N}\}$, assuming that $S$ is bounded above. According to the *Completeness Axiom*, $S$ has a least upper bound, we can call it $x$. We claim that the sequence $\{x_n\}$ converges to $x$. To claim this, we must prove that for all integers, $n \geq N$ and $\epsilon > 0$ provided that

$$\left|x_n - x\right| < \epsilon, \tag{11}$$

that is,

$$x - \epsilon < x_n < x + \epsilon \quad \forall \text{ integers } n \geq N. \tag{12}$$

Since $x$ is the least upper bound for the set $S$, so

$$x_n \leq x < x + \epsilon \quad \forall \text{ integers } n \geq N. \tag{13}$$

Furthermore, $x - \epsilon$ is not an upper bound for set $S$, so there must be a natural number $N$ such that $x - \epsilon < x_N$, since the sequence is monotonically increasing, so

$$x - \epsilon < x_N \leq x_n \quad \forall \text{ integers } n \geq N. \tag{14}$$

We have now proved that a monotonically increasing sequence must converge to its least upper bound and, in the same way, we can also claim that a monotonically decreasing sequence must converge to its greatest lower bound. □

To return to the HS algorithm mentioned in the previous section: when HS finds a solution which is better than the worst harmony in the current HM, then the operator "HM updating" is implemented, which means that for the current iteration, the algorithm finds a solution that is not worse than the one that was found by the algorithm in the last iteration. Suppose that the problem to be solved is the minimum problem described as (1), the current solution offered is $x_{n+1}$ and the last found solution is $x_n$, then we have $x_{n+1} \leq x_n$, and if we let the best solution in HM for each iteration constitute a sequence $S = \{x_n \mid n \in N\}$, the sequence must be a monotonically decreasing sequence as in Definition 1, and as we have proved, if the sequence has a greatest lower bound, the sequence must converge to the greatest lower bound.

In fact, for a continuous function, suppose that the function is named $f(X)$, and if $X \in [a, b]^n$, where $n$ is the dimension of $X$, then there must be a number $f(X_1)$ and a number $f(X_2)$ (assuming that $f(X_1) \leq f(X_2)$) satisfying the following inequality:

$$f(X_1) \leq f(X) \leq f(X_2). \tag{15}$$

So in the case that the object function $f(X)$ is continuous and $x_i \in [a_i, b_i]$, where $i = 1, 2, \ldots, n$ and $X = (x_1, x_2, \ldots, x_n)$, then the sequence $\{f_k\}$ which is composed of the best solution $f_k$ in the HM for the $k$th iteration is bounded and $f(X_1) < f_k < f(X_2)$, as we have mentioned above for a minimum problem, the sequence $\{f_k\}$ is monotonically decreasing, so $\{f_k\}$ must converge.

We have explained that HS must converge, but can it converge to $f(X_1)$? In fact, $f(X_1)$ is the global optimum of the object function. From Theorem 2, we know that if $f(X_1)$ is the greatest lower bound, $\{f_k\}$ must converge to $f(X_1)$. The problem is whether $f(X_1)$ is the greatest lower bound for $\{f_k\}$. This is, in fact, difficult to solve. HS is a stochastic algorithm and for each iteration, the best solution in HM is random so that it is not a fixed predictable number, but we are able to calculate the probability that the HM is updated with. An example would be as follows: suppose that the object function is a minimum problem, the expression is $f(x) = x^2$ which has only one dimension, and $x \in [x_L, x_U]$, and HMS = 1, for the $k$th iteration, the best solution is $f(x_k)$ (suppose that $x_k > 0$), so the HM is updated with probability HMCR × PAR × 0.5 + (1 − HMCR) × $(x_k/(x_U - x_L))$ when $2x_k >$ bw or HMCR × PAR × 0.5 × $(2x_k/\text{bw})$ + (1 − HMCR) × $(x_k/(x_U - x_L))$ when $2x_k \leq$ bw. The higher the probability is, the more quickly the HM is updated. From the expression, it is clear that the probability decreases by $x_k$, so that as the iteration number increases, the probability decreases sharply, and the convergence curve becomes flatter.

```
Initialize parameters of algorithm
for each subpopulation do
    Initialize all the subpoputions randomly
    Evaluate the fitness of each individual
end for
while termination condition is false do
    for each subpopulation do
        Using GA operators to generate offsprings
        Evaluate the fitness of each off spring
    end for
end while
```

Pseudocode 2: Pseudocode for the CCGA.

```
Initialize the parameters
Divide the n decision variables into m groups
for each HM do
    Initialize HM randomly
end for
while termination condition is false do
    calculate PAR and bw by using
    Equation (7)
    for each HM do
        generate a new harmony
        evaluate the new harmony
    end for
end while
```

Pseudocode 3: Pseudocode for the CHS algorithm.

This might also explain why IHS performs better than SHS. The two parameters PAR and bw are very important for HS. In SHS, the two parameters are fixed while in IHS, PAR increases linearly and the bw index decreases, so that, from the expression of the HM updated probability, we can see that this is very reasonable, owing to the fact that this strategy is capable of making the probability decrease slowly, with IHS generally performing better. The results of this experiment are demonstrated in Section 6.

## 4. CHS Algorithm

*4.1. A Brief Overview of Cooperative Method.* The natural world is a very complex system, and evolution is one element within it. EAs imitate the evolution of species, but they are just a simple simulation and almost all EAs just mimic the evolution of one species. There are a large number of species on the earth, with some of them being closely associated [20]. In order to survive, different individuals from different species or the same species may compete or cooperate with each other. Cooperative coevolutionary algorithms just act as models of symbiotic coevolution. The individuals are from different species or subpopulations, and they have to cooperate with some others. The difficulty involved in cooperative coevolutationary algorithms is how to split the fitness that is achieved by collective effort definitively [21, 22].

Potter and De Jong applied the method to GA, coming up with CCGA. They evolved the solutions in different subpopulations, and each subpopulation has just one dimension of the solution vector. The pseudocode for the CCGA is given in Pseudocode 2. The initial fitness of each subpopulation member is computed by combining it with a random individual from each of the other subpopulations and then using the object function to evaluate the complete solution. After initializing, one individual of a subpopulation cooperates only with the best individual from each of the other subpopulations.

Potter and De Jong also found that the cooperative approach does not perform well when problem parameters are strongly interdependent, due to the greediness of the credit assignment approach [11]. To reduce greediness, Potter and De Jong suggested that random collaboration should be used to generate a complete solution. An individual from one subpopulation can collaborate with the best individual of each of the other subpopulations and then constitute one complete solution, or the individual can cooperate with one individual of each of the other subpopulations randomly and then generate another solution. The fitness of the individual is equal to the better fitness between the two solutions.

*4.2. The CHS.* In the CHS algorithm, to reduce the complexity, we use $m$ ($m \leq n$) HMs instead of $n$ HMs, where $n$ is the number of decision variables. The pseudocode for CHS algorithm is shown in Pseudocode 3. Each HM may thus contain more than one decision variables, so that, for each HM, we can use the HS operators including harmony consideration and pitch adjustment to generate a new harmony vector, but the vector is just a part of a complete solution vector, and it must collaborate with harmony vectors from every other HM, and as we have mentioned above, when initializing HMs, the cooperation is random. When calculating a new harmony vector which is improvised by using HS operators, it cooperates only with the best ones of each HM. This is similar to several musicians working together to find the best harmony, with each musician being in charge of a part of the whole symphony, assuming that all of them are selfless. When they work together, each one shares the best harmony that he or she has found by himself or herself. In our daily life, teamwork is very universal, and generally it is more effective than work being carried out alone.

We should ask why it brings better solutions when using cooperative method. There are few mathematical foundations for the method, but this can be explained by using an example. HM updating is very important for the HS algorithm, when the HM is updated, it indicates that a better solution may be found. If the HM is updated with a higher probability or in other words, the algorithm has a higher probability of finding better solutions, it will converge faster. For instance, suppose that the object function is $\min f(x, y) = x^2 + y^2$, it has two decision variables $x$ and $y$, and $x \in [x_{\min}, x_{\max}]$, $y \in [y_{\min}, y_{\max}]$. It is quite obvious that the origin is the global optimum. Assume that the worst solution vector in the initial HM is $(x_0, y_0)$, which is called point $A$ and is shown in Figure 2. If we use the CHS algorithm, two HMs will be

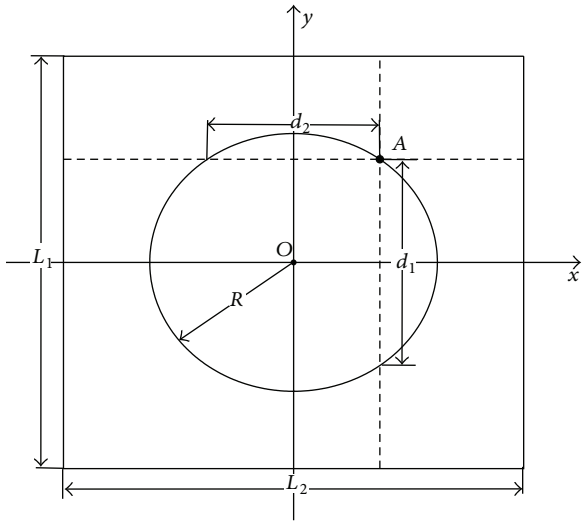


Figure 2: Diagram illustrating the advantage of CHS algorithm.

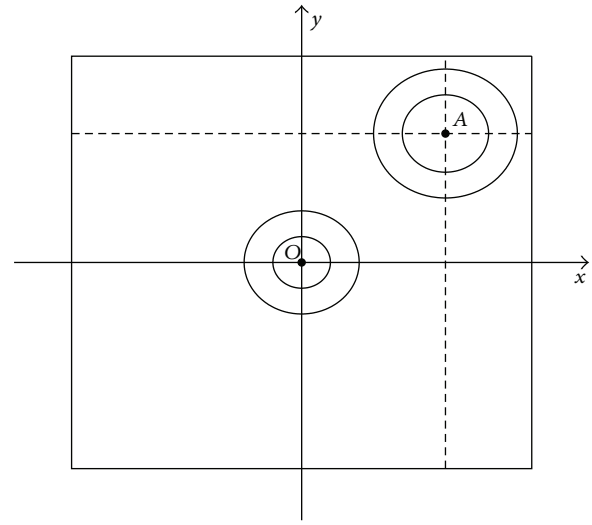


Figure 3: Diagram illustrating the disadvantage of CHS algorithm.

used, one HM ($HM_1$) representing $x$ and another ($HM_2$) representing $y$. In one cycle, the HMs are used one by one and when $HM_1$ is used to generate a new harmony vector, this new vector must cooperate with the best one in $HM_2$ in order to constitute a complete solution vector, meaning that $y$ is fixed when the algorithm searches in the subspace $x$, and it is the same for $y$.

When $y$ is fixed, the CHS algorithm is able to find a better solution with probability $d_2/L_2$, and when $x$ is fixed, the probability is $d_1/L_1$, so after one cycle, the CHS algorithm has the $P_{\text{CHS}}$ probability of updating the HM, where $P_{\text{CHS}} = d_1/L_1 + d_2/L_2$, but for SHS algorithm, if the HM is updated, the algorithm must find a solution which is located in the circular region, so the HM is updated with probability $P_{\text{SHS}}$, where $P_{\text{SHS}} = \pi R^2/L_1L_2$. Also, $P_{\text{CHS}} = d_1/L_1 + d_2/L_2 = (d_2L_1 + d_1L_2)/L_1L_2$, because of $d_2L_1 + d_1L_2 > d_2^2 + d_1^2 > \pi R^2$, so $P_{\text{CHS}} > P_{\text{SHS}}$, which means that the CHS algorithm has a higher probability of updating HM than the SHS algorithm, meaning that the CHS is able to find better solutions than the SHS algorithm.

The cooperative method also has disadvantages, however, with one being that the CHS algorithm is easier to trap in a local minimum. This phenomenon is shown in Figure 3, in which point $A$ is a local minimum point and point $O$ or the origin is the global minimum point; the area where $A$ is located in is local minimum area and the global minimum region is where $O$ is located. If the algorithm is trapped in point $A$, when we use the CHS algorithm in every iteration, only one subspace is searched. Assuming that $y$ is fixed and $x$ is searched, irrespective of the value of $x$, the CHS algorithm has no way of reaching the global minimum region, and it is the same for $y$. In this case, the CHS algorithm will never escape from the local minimum. For the SHS algorithm, however, it is possible to reach the global minimum area, although the probability is low.

## 5. Experimental Setup

*5.1. Benchmark Functions.* Ten benchmark functions (five unrotated and five rotated) have been chosen to test the performance of the CHS algorithm, and SHS, his, and GHS algorithms are also tested for the sake of comparison. The unrotated functions are as follows.

The Quadric function is

$$f_1(X) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2, \tag{16}$$

where $n = 30$, $X \in [-100, 100]^{30}$. This function is a non-separable unimodal function, with the global minimum point being $X^* = (0, 0, \ldots, 0)$.

Ackley's function is

$$f_2(X) = -20 \exp\left( -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) + 20 - \exp\left( \frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + e, \tag{17}$$

where $n = 30$, $X \in [-30, 30]^{30}$. Ackley's function is a separable multimodal function, with the global optimum being $X^* = (0, 0, \ldots, 0)$.

The Generalized Rastrigin function is

$$f_3(X) = \sum_{i=1}^{n} \left( x_i^2 - 10 \cos(2\pi x_i) + 10 \right), \tag{18}$$

where $n = 30$, $X \in [-5.12, 5.12]^{30}$. The Rastrigin function is a separable multimodal function, with the global optimum being $X^* = (0, 0, \ldots, 0)$.

The Generalized Griewank function is

$$f_4(X) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left( \frac{x_i}{\sqrt{i}} \right) + 1, \tag{19}$$

where $n = 30$, $X \in [-600, 600]^{30}$. This function is a very complex, nonlinear, separable, and multimodal function and is very difficult for optimization algorithms. The global optimum is $X^* = (0, 0, \ldots, 0)$.

Rosenbrock's function (or Banana-valley function) is

$$f_5(X) = \sum_{i=1}^{n-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + \left(1 - x_i\right)^2\right), \quad (20)$$

where $n = 30$, $X \in [-2.048, 2.048]^{30}$. Rosenbrock's function is a naturally nonseparable function, the global optimum being $X^* = (1, 1, \ldots, 1)$ which is located in a narrow valley; it is very hard for algorithms to find the correct direction and reach to the global minimum area.

The parameters of rotated functions are the same as unrotated functions and all the orthogonal matrices are fixed, meaning that all functions are rotated at a fixed angle.

*5.2. Configuration of Algorithms.* The iteration number of each experiment is $3 \times 10^4$, with all the experiments being ran independently 30 times. The results reported are averages, and the best solutions have been calculated from all 30 runs. The parameters of the algorithms are as follows.

(i) Parameters of the SHS algorithm: HMS = 30 and HMCR = 0.95. This was suggested by Yang [12]. PAR = 0.3 and bw = 0.01. The values of the last two parameters (PAR, bw) were suggested by Omran and Mahdavi [19].

(ii) Parameters of the IHS algorithm: HMS = 30 and HMCR = 0.95, with PAR increasing linearly over time (by using (7)), and $\text{PAR}_{\min} = 0.01$, $\text{PAR}_{\max} = 0.99$ which were suggested by Omran and Mahdavi [19]. bw decreases by using (6), and $\text{bw}_{\min} = 1e^{-005}$, $\text{bw}_{\max} = 5$.

(iii) Parameters of the GHS algorithm: HMS = 30, HMCR = 0.95, $\text{PAR}_{\min} = 0.01$, and $\text{PAR}_{\max} = 0.99$.

(iv) Parameters of the CHS algorithm: HMS = 30, HMCR = 0.95, $\text{PAR}_{\min} = 0.01$, $\text{PAR}_{\max} = 0.99$, $\text{bw}_{\min} = 1e^{-005}$, $\text{bw}_{\max} = 5$, and the number of groups $m = 6, 10, 15, 30$.

## 6. Results

This section shows the experimental results gathered by allowing all of the methods tested to run for a fixed number of function evaluations, that is, $3 \times 10^4$. All the results are shown in Table 1 to Table 10, for each table, the first column represents the algorithms used, the second lists the mean error and 95% confidence interval after $3 \times 10^4$ times iteration, and the third lists the best solution found by the algorithm after 30 runs. We should not forget that all the functions have a minimum function value of 0.

Table 1 shows the experimental results of the unrotated Quadric function; this is a nonseparable function, and it is hard to optimize for HS algorithms, but GHS is better able to solve the problem during the 30 runs; $\text{CHS}_{30}$ also finds a satisfying solution. Table 2 shows the experimental results of

Table 1: Results of unrotated Quadric function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $5.46e+003 \pm 5.97e+002$ | $2.38e+003$ |
| IHS | $5.87e+003 \pm 7.37e+002$ | $1.67e+003$ |
| GHS | $2.55e+003 \pm 2.11e+003$ | $3.64e+000$ |
| $\text{CHS}_6$ | $1.91e+003 \pm 2.61e+001$ | $5.53e+002$ |
| $\text{CHS}_{10}$ | $2.05e+003 \pm 3.82e+002$ | $9.13e+002$ |
| $\text{CHS}_{15}$ | $2.29e+003 \pm 6.42e+002$ | $3.97e+002$ |
| $\text{CHS}_{30}$ | $1.24e+003 \pm 4.95e+002$ | $6.36e+001$ |

Table 2: Results of rotated Quadric function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $8.09e+002 \pm 3.04e+002$ | $9.72e+002$ |
| IHS | $1.09e+003 \pm 4.35e+002$ | $1.05e+002$ |
| GHS | $8.99e+002 \pm 6.22e+002$ | $7.78e-002$ |
| $\text{CHS}_6$ | $1.78e+002 \pm 1.16e+002$ | $6.24e-008$ |
| $\text{CHS}_{10}$ | $1.61e+002 \pm 1.84e+002$ | $2.39e-010$ |
| $\text{CHS}_{15}$ | $7.24e+001 \pm 5.43e+001$ | $3.81e-011$ |
| $\text{CHS}_{30}$ | $4.33e+000 \pm 8.31e+000$ | $1.04e-011$ |

Table 3: Results of unrotated Ackley's function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $5.54e-001 \pm 1.16e-001$ | $2.96e-002$ |
| IHS | $1.08e-002 \pm 1.76e-002$ | $3.90e-005$ |
| GHS | $1.78e-002 \pm 5.05e-003$ | $5.78e-006$ |
| $\text{CHS}_6$ | $8.96e-006 \pm 1.79e-007$ | $7.29e-006$ |
| $\text{CHS}_{10}$ | $4.86e-006 \pm 1.31e-007$ | $4.01e-006$ |
| $\text{CHS}_{15}$ | $1.40e-006 \pm 4.58e-008$ | $1.18e-006$ |
| $\text{CHS}_{30}$ | $7.06e-008 \pm 3.57e-009$ | $5.18e-008$ |

Table 4: Results of rotated Ackley's function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $3.95e+000 \pm 2.37e-001$ | $2.62e+000$ |
| IHS | $1.72e-003 \pm 3.22e-003$ | $4.02e-005$ |
| GHS | $3.48e-001 \pm 2.16e-001$ | $6.64e-005$ |
| $\text{CHS}_6$ | $3.54e+000 \pm 3.54e-001$ | $1.50e+000$ |
| $\text{CHS}_{10}$ | $5.33e+000 \pm 6.98e-001$ | $3.03e+000$ |
| $\text{CHS}_{15}$ | $7.15e+000 \pm 9.90e-001$ | $2.41e+000$ |
| $\text{CHS}_{30}$ | $6.86e+000 \pm 1.07e+000$ | $3.57e+000$ |

Table 5: Results of unrotated Rastrigin function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $4.39e-001 \pm 1.65e-001$ | $2.96e-002$ |
| IHS | $1.44e+000 \pm 3.12e-001$ | $2.31e-002$ |
| GHS | $2.20e-003 \pm 1.28e-003$ | $4.41e-007$ |
| $\text{CHS}_6$ | $3.01e-008 \pm 1.31e-009$ | $2.14e-008$ |
| $\text{CHS}_{10}$ | $8.88e-009 \pm 4.35e-010$ | $6.14e-009$ |
| $\text{CHS}_{15}$ | $7.59e-010 \pm 6.88e-011$ | $4.63e-010$ |
| $\text{CHS}_{30}$ | $1.69e-012 \pm 2.22e-013$ | $7.99e-013$ |

Table 6: Results of rotated Rastrigin function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $1.92e+002 \pm 2.99e+000$ | $1.77e+002$ |
| IHS | $1.92e+002 \pm 2.85e+000$ | $1.73e+002$ |
| GHS | $1.86e+002 \pm 3.81e+000$ | $1.65e+002$ |
| $CHS_6$ | $8.22e+001 \pm 8.38e+000$ | $4.38e+001$ |
| $CHS_{10}$ | $1.09e+002 \pm 1.38e+001$ | $4.28e+001$ |
| $CHS_{15}$ | $1.37e+002 \pm 1.14e+001$ | $8.86e+001$ |
| $CHS_{30}$ | $1.71e+002 \pm 1.38e+001$ | $6.96e+001$ |

Table 7: Results of unrotated Griewangk function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $1.06e+000 \pm 7.92e-003$ | $1.02e+000$ |
| IHS | $1.05e+000 \pm 5.35e-003$ | $1.02e+000$ |
| GHS | $3.52e-002 \pm 2.78e-002$ | $3.31e-006$ |
| $CHS_6$ | $5.02e-002 \pm 9.63e-003$ | $6.61e-012$ |
| $CHS_{10}$ | $7.06e-002 \pm 2.14e-002$ | $1.82e-012$ |
| $CHS_{15}$ | $7.79e-002 \pm 1.49e-002$ | $9.86e-003$ |
| $CHS_{30}$ | $7.07e-002 \pm 2.03e-002$ | $1.72e-002$ |

Table 8: Results of rotated Griewangk function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $1.06e+000 \pm 8.65e-003$ | $1.03e+000$ |
| IHS | $1.05e+000 \pm 6.78e-003$ | $1.00e+000$ |
| GHS | $4.79e-001 \pm 7.81e-002$ | $1.36e-004$ |
| $CHS_6$ | $9.61e-003 \pm 3.90e-003$ | $4.96e-012$ |
| $CHS_{10}$ | $1.50e-002 \pm 5.09e-003$ | $2.04e-012$ |
| $CHS_{15}$ | $1.22e-002 \pm 5.19e-003$ | $2.53e-013$ |
| $CHS_{30}$ | $1.09e-002 \pm 5.78e-003$ | $8.00e-016$ |

Table 9: Results of unrotated Rosenbrock function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $5.54e+001 \pm 7.88e+000$ | $3.85e+000$ |
| IHS | $4.92e+001 \pm 9.34e+000$ | $1.32e+001$ |
| GHS | $6.89e+001 \pm 1.63e+001$ | $2.67e+001$ |
| $CHS_6$ | $3.16e+001 \pm 6.67e+000$ | $1.18e-002$ |
| $CHS_{10}$ | $3.57e+001 \pm 8.76e+000$ | $2.36e+000$ |
| $CHS_{15}$ | $2.77e+001 \pm 7.78e+000$ | $4.49e+000$ |
| $CHS_{30}$ | $1.51e+001 \pm 2.20e+000$ | $1.53e-003$ |

Table 10: Results of rotated Rosenbrock function.

| Algorithm | Mean | Best |
|---|---|---|
| SHS | $2.81e+001 \pm 3.80e-001$ | $2.58e+001$ |
| IHS | $2.80e+001 \pm 3.40e-001$ | $2.61e+001$ |
| GHS | $2.91e+001 \pm 9.03e-001$ | $2.78e+001$ |
| $CHS_6$ | $2.99e+001 \pm 5.48e+000$ | $2.11e+001$ |
| $CHS_{10}$ | $2.55e+001 \pm 4.57e+000$ | $1.64e+001$ |
| $CHS_{15}$ | $2.24e+001 \pm 1.45e+000$ | $1.67e+001$ |
| $CHS_{30}$ | $1.89e+001 \pm 1.18e+000$ | $1.63e+001$ |

the rotated Quadric function. The results are very different from those shown in Table 1.

The results of the experiment show that the rotated Quadric function is more easily optimized by HS algorithms and that the CHS can always find more satisfying solutions than other HS algorithms. As the number of groups $m$ increases, the solution becomes better and better, but at the same time, the algorithm becomes more and more complex. Figure 4 shows the best function value profile of each algorithm both for unrotated and rotated Quadric functions. This figure is a more visual representation. It shows the best results among 30 independent runs for each algorithm. From Figure 4, we can deduce that in unrotated cases, all the algorithms converge very sharply, while in rotated cases, the solutions found by CHS are better.

The results of Ackley's function are very interesting and very different from the results of the Quadric function. Table 3 shows the experimental results of unrotated Ackley's function; this is a multimodal function and is easily optimized by HS algorithms. Almost all the algorithms are capable of finding a satisfactory solution, but among the tested algorithms, CHS performs better with both the mean and the best results than the other algorithms, and the higher $m$ is, the better the solution is. However, in terms of the rotated Ackley's function, CHS does not perform as well as it is illustrated in Table 4. In terms of the rotated Ackley's function, IHS and GHS perform better, especially the IHS algorithm. This can be deduced from Figure 5. When the search space is rotated, all the algorithms perform worse, especially CHS, the performance of which is as bad as SHS. GHS converges very fast to a local optimum, although it is capable of finding a better solution. This is because GHS resembles the standard PSO, which is not good at dealing with multimodal problems.

The Rastrigin function is also a multimodal function with many local optima. In an unrotated case, CHS performs relatively well, especially $CHS_{30}$. It is the performance leader, as shown in Table 5. The results of the experiment are very similar to those observed with unrotated Ackley's function: the higher $m$ is, the better the solution found by CHS will be. In the rotated case, however, none of the algorithms performs so well, as shown in Table 6. For CHS, with the increase of $m$, the mean of the solutions becomes worse, which is the reverse of the unrotated case. Figure 6 shows the best function value profile. From Figure 6(a), we can find that, generally, CHS converges, offering a better solution faster, while Figure 6(b) shows a very different result.

Tables 7 and 8 show the results of the Griewank function, which is a nonlinear multimodal and complex function. In unrotated cases, CHS performs better than both SHS and IHS, but not so well as GHS. Furthermore, when $m$ increases, the performance of CHS is very similar. This is very different from the results of unrotated Ackley's function and unrotated Rastrigin function, which are shown from Table 3 to Table 6. In rotated cases, the mean for each algorithm is very similar, but the best solutions for each algorithm are distinct from each other, as shown in Table 8. We might also observe that the higher $m$ is, the better the solution found by CHS. Figure 7

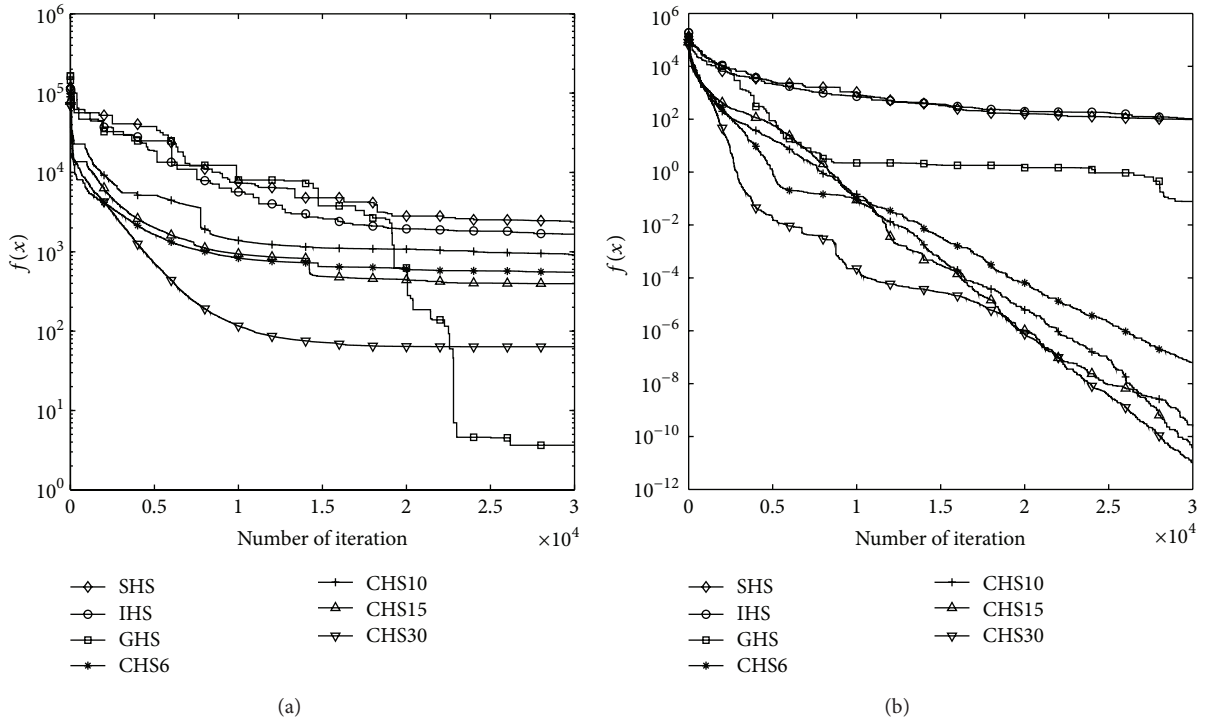


FIGURE 4: Quadric ($f_1$) best function value profile. (a) Unrotated Quadric best function value profile. (b) Rotated Quadric best function value profile.

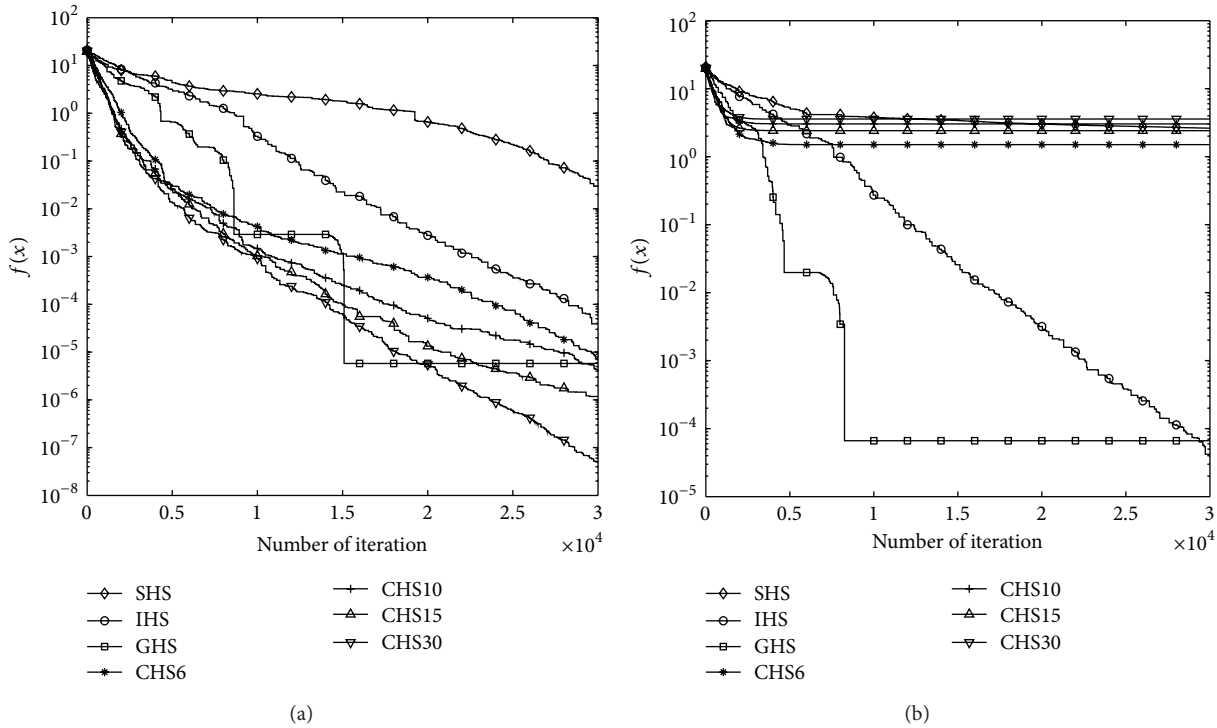


FIGURE 5: Ackley ($f_2$) best function value profile. (a) Unrotated Ackley best function value profile. (b) Rotated Ackley best function value profile.

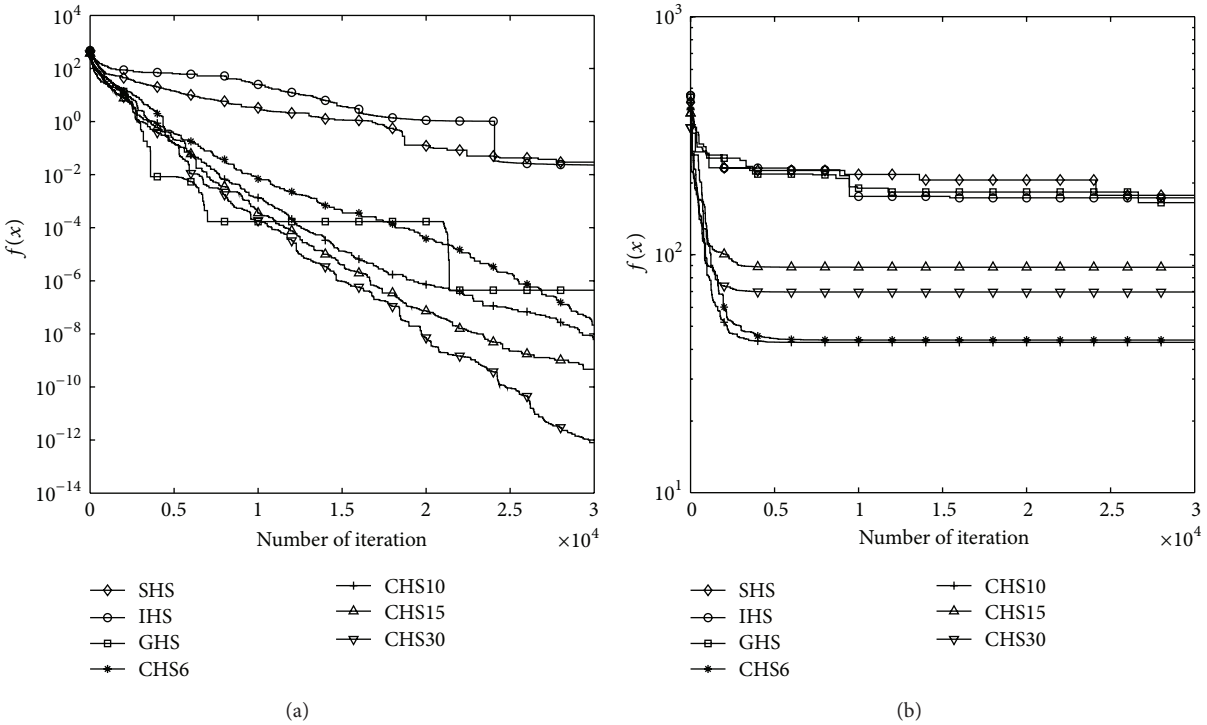


Figure 6: Rastrigin ($f_3$) best function value profile. (a) Unrotated Rastrigin best function value profile. (b) Rotated Rastrigin best function value profile.

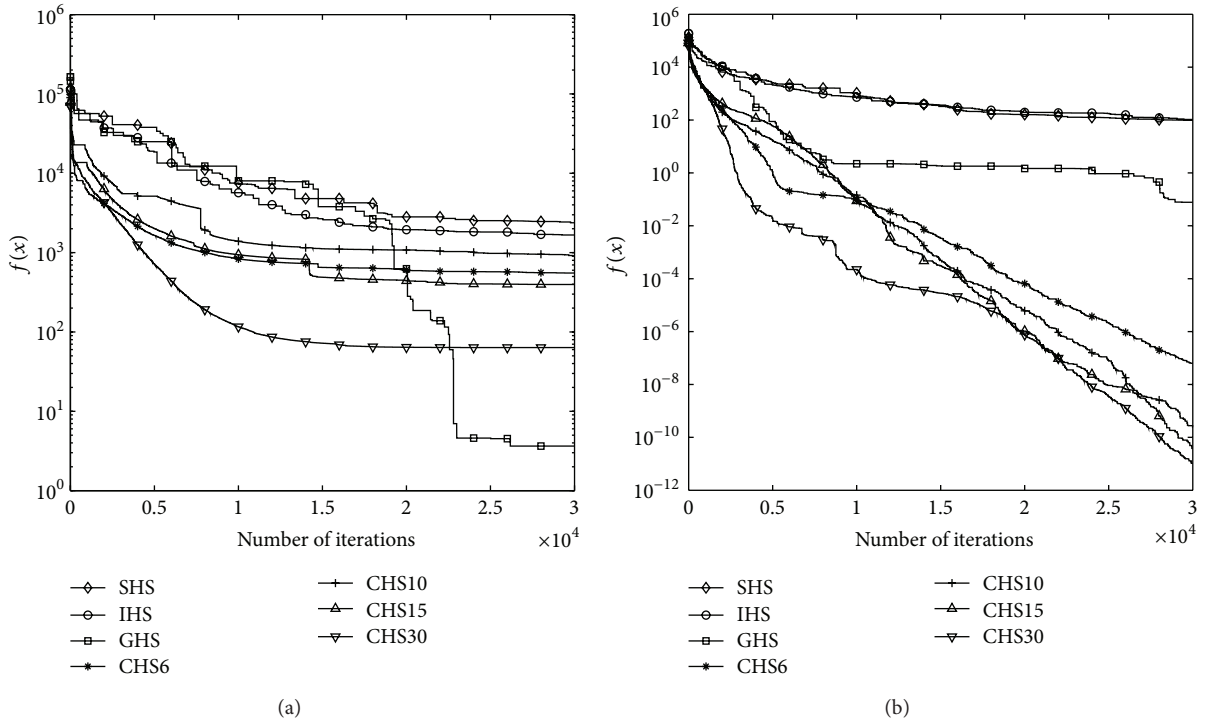


Figure 7: Griewangk ($f_4$) best function value profile. (a) Unrotated Griewangk best function value profile. (b) Rotated Griewangk best function value profile.

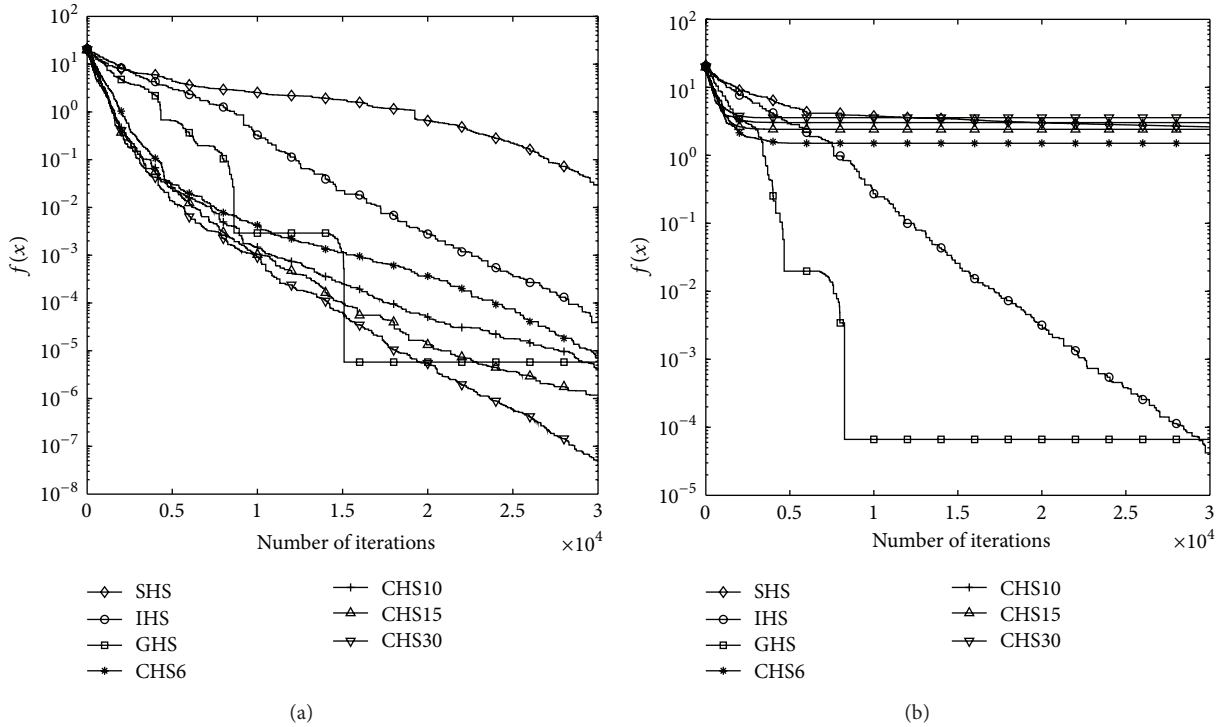


FIGURE 8: Rosenbrock ($f_5$) best function value profile. (a) Unrotated Rosenbrock best function value profile. (b) Rotated Rosenbrock best function value profile.

shows the best function value profile for each algorithm. For both the unrotated Griewank function and the rotated Griewank function, SHS and IHS trap in a local optimum easily, but GHS is capable of finding a better solution.

The results of the Rosenbrock function are shown in Tables 9 and 10. The global optimum of the Rosenbrock function is located in a narrow valley so that it is very difficult for the algorithms to find the correct direction in order to reach the global region, and thus, none of the algorithms is satisfactory, especially in rotated cases. This is shown more visually in Figure 8.

In this section, we have examined the results of the experiments and from these results, it has been possible to deduce that CHS is capable of solving all the problems that can be solved by SHS. In most cases, the results found by CHS are better than those found by the SHS algorithm or the IHS algorithm. We should, however, bear in mind the "no free lunch" theory, which acts as a reminder that no algorithm is capable of performing satisfactorily for all problems. Each algorithm, thus, has its own advantages and, as a result, is able to deal with one kind of problem only.

## 7. Conclusions

A cooperative harmony search algorithm has been presented in this paper, and as the results have shown, this cooperative approach constitutes a very real improvement in terms of the performance of SHS. This is especially so regarding the quality of the solutions, and in most cases, the higher the number of groups $m$, the better the solution. CHS can furthermore be applied as a solution to all the same problems as the SHS algorithm, so that, when the dimensionality of the problem increases, the performance of the CHS algorithm is significantly better.

Despite the fact that the CHS algorithm has numerous advantages, there are also some shortcomings. As we mentioned in Section 4, the CHS algorithm may be more easily trapped in a local optimum. As the CHS algorithm is not an answer to every problem, we must ask ourselves the question of when exactly it is that the CHS algorithm can perform better and which kinds of problems can be solved satisfactorily by the CHS algorithm. This is a problem that should be solved urgently. Another unsolved problem is why the cooperative method is actually better. We explained this in Section 4 by means of examples, but the experiment still requires mathematical analysis and proof.

Another aspect to which attention should be paid is the mathematical analysis of the HS algorithm. Although we have proved that the SHS algorithm must converge, can it in fact converge to the global optimum or to a local optimum? In addition, it is not clear that what the influences the parameters have on HS algorithm are, as all the parameters used here have been suggested by other scholars or selected by experience. If someone were to provide a mathematical analysis for the parameters, it would be very helpful for the future development of the HS algorithm.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work has been supported by the National Natural Science Foundation of China under Grant no. 51109045 and the Postdoctoral Foundation of Heilongjiang province under Grant no. LBH-Z10217.

## References

[1] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 1975.

[3] D. B. Fogel, L. J. Fogel, and J. W. Atmar, "Meta-evolutionary programming," in *Proceedings of the 25th Asilomar Conference on Signals, Systems & Computers*, pp. 540–545, IEEE, November 1991.

[4] I. Rechenberg, *Cybernetic Solution Path of an Experimental Problem*, 1965.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Wash, USA, December 1995.

[6] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[7] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[8] 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.

[9] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[10] K. A. De Jong and M. A. Potter, "Evolving complex structures via cooperative coevolution," in *Evolutionary Programming*, pp. 307–317, 1995.

[11] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature—PPSN III*, vol. 866, pp. 249–257, Springer, 1994.

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

[13] Z. W. Geem, "Optimal cost design of water distribution networks using harmony search," *Engineering Optimization*, vol. 38, no. 3, pp. 259–277, 2006.

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

[15] D.-X. Zou, L.-Q. Gao, P.-F. Wu, and J.-H. Wu, "A global harmony search algorithm and its application to PID control," *Journal of Northeastern University*, vol. 31, no. 11, pp. 1534–1537, 2010.

[16] M. Mahdavi and H. Abolhassani, "Harmony K-means algorithm for document clustering," *Data Mining and Knowledge Discovery*, vol. 18, no. 3, pp. 370–391, 2009.

[17] L. Wang, R. Yang, Y. Xu, Q. Niu, P. M. Pardalos, and M. Fei, "An improved adaptive binary harmony search algorithm," *Information Sciences*, vol. 232, pp. 58–87, 2013.

[18] 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.

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

[20] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, John Wiley & Sons, 2007.

[21] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[22] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving from Nature, PPSN XI*, vol. 6239 of *Lecture Notes in Computer Science*, pp. 300–309, Springer, 2010.