Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(4): 4226–4246.
DOI: 10.3934/mbe.2021212
Received: 06 April 2021
Accepted: 12 May 2021
Published: 17 May 2021

*Research article*

# Adaptive harmony search algorithm utilizing differential evolution and opposition-based learning

**Di-Wen Kang, Li-Ping Mo*, Fang-Ling Wang and Yun Ou**

College of Information Science and Engineering, Jishou Unversity, Jishou 416000, China

* **Correspondence:** Email: zmx89@jsu.edu.cn.

**Abstract:** An adaptive harmony search algorithm utilizing differential evolution and opposition-based learning (AHS-DE-OBL) is proposed to overcome the drawbacks of the harmony search (HS) algorithm, such as its low fine-tuning ability, slow convergence speed, and easily falling into a local optimum. In AHS-DE-OBL, three main innovative strategies are adopted. First, inspired by the differential evolution algorithm, the differential harmonies in the population are used to randomly perturb individuals to improve the fine-tuning ability. Then, the search domain is adaptively adjusted to accelerate the algorithm convergence. Finally, an opposition-based learning strategy is introduced to prevent the algorithm from falling into a local optimum. The experimental results show that the proposed algorithm has a better global search ability and faster convergence speed than other selected improved harmony search algorithms and selected metaheuristic approaches.

**Keywords:** harmony search algorithm; differential evolution; opposition-based learning; adaptive adjustment strategy; optimization

## 1. Introduction

Metaheuristic algorithms (the genetic algorithm (GA) [1], the particle swarm optimization (PSO) algorithm [2], the harmony search (HS) algorithm [3], etc.) are widely used to solve many optimization problems due to the mechanisms providing high-quality solutions, reasonable processing times, and other advantages. Among the series of classic metaheuristic algorithms, the HS algorithm has several advantages. For example, all individuals have a chance to influence the generation of new individuals, and the value of each dimension of the new individual is generated independently [4, 5]. Therefore, the HS algorithm is used in various field optimization problems, such as structural design [6], object detection [7], economic dispatch [8], vehicle routing [9], the estimation of the optimum number of wind turbines [10], buffer allocation problems [11], weighted Fuzzy production rule extraction [12], and tool indexing [13]. Although research on the HS algorithm has gained fruitful and meaningful achievements, the development of the algorithm is still in the initial era because of the algorithm's shortcomings, such as its slow convergence and easy falling into a local optimum [4, 5, 14]. Currently, many improved HS algorithms have been suggested to obtain better performance, and they have focused on two aspects: 1) Automatically adjusting related parameters using dynamic mechanisms. E.g., [15] presents an improved harmony search (IHS) algorithm that includes dynamic adaptation for the pitch adjustment rate (PAR) and bandwidth (bw). [16] replaces the pitch adjustment step with a new strategy based on the optimal value in the harmony memory (HM). [17] presents a global dynamic harmony search algorithm (GDHS). In the GDHS, all parameters and the search domain can be adjusted adaptively. 2) Hybridization with other metaheuristic algorithms results in unique highlights. E.g., [18] presents an improved differential-based harmony search algorithm with linear dynamic domain (ID-HS-LDD), which has two characteristics. On one hand, the fine-tuning parameter is generated by utilizing differential evolution. On the other hand, an adaptively adjusted mechanism is used to control the search domain. [19] uses a new harmony selection mechanism inspired by a global best concept of PSO and the roulette wheel memory consideration. [20] presents an improved HS algorithm hybridized with differential evolution. [21] modifies the HS algorithm using the cooling strategy of the simulated annealing (SA) algorithm. The abovementioned HS variants all achieve better performance than the HS algorithm. However, some variants accelerate the convergence speed but easily fall into local optima, leading to premature convergence. Some variants do not consider the harmonies in HM when fine-tuning, leading to low precision. Some variants even adopt the forced convergence strategy, but this strategy is only effective for objective functions that can obtain the optimal solution at position zero. It makes sense to find a new variant of the HS algorithm to overcome the above shortcomings.

Inspired by the above references, an adaptive HS algorithm utilizing differential evolution and opposition-based learning (AHS-DE-OBL) is employed in this manuscript to solve the HS algorithm's existing shortcomings by hybridizing differential evolution (DE), opposition-based learning (OBL), and the adaptive adjustment of algorithm parameters. The DE algorithm is an optimization algorithm utilizing the theory of swarm intelligence to generate optimal values through cooperation and competition among the individuals in a group [22]. The mutation operation in DE considers the influence of the original individual on the new individual. In order to make the fine-tuning parameter change with the distribution of the harmonies in HM, AHS-DE-OBL generates a fine-tuning parameter utilizing DE in each iteration's improvisation stage. The OBL strategy can regularly generate new individuals and speed up convergence by comparing individuals' adaptability with their opposite individuals and retaining the fittest [23]. In order to increase the richness of HM and avoid easily becoming stuck in the local optimum, AHS-DE-OBL generates extra harmonies utilizing OBL. After updating, AHS-DE-OBL dynamically adjusts the search domain according to the distribution of harmony in HM. In the experiment, ten classic benchmark functions are used to compare AHS-DE-OBL with other HS variants. The experimental results show that the AHS-DE-OBL algorithm has better performance than the selected HS variants and the selected other metaheuristic algorithms.

The outline of the remainder of this article is summarized as follows. In Section 2, a simple introduction to the HS algorithm is given. In Section 3, the influences of the parameters on the HS algorithm are analyzed. In Section 4, the new algorithm, named AHS-DE-OBL, is described in detail. Section 5 shows the experiment and discussions. Finally, Section 6 presents a summary.

## 2. Brief introduction of the HS algorithm

Assume $f(x)$ is the objective function, and the HS algorithm is used to search the minimum value in the interval [*L, U*], where *L* and *U* represent the lower and upper bounds of the search domain, respectively. The steps of the HS algorithm are as follows:

Step 1: Initialize the parameters. The parameters include the Harmony Memory Size (*HMS*), Harmony Memory Consideration Rate (*HMCR*), Pitch Adjustment Rate (*PAR*), and bandwidth (*bw*). The meaning of each parameter is shown in Table 1.

**Table 1.** Parameters and corresponding meanings.

| Parameters | Meaning |
|---|---|
| *HMS* | Harmony memory size. |
| *HMCR* | Probability of randomly selecting a harmony from the HM. |
| *PAR* | Probability of fine-tuning randomly selecting a harmony. |
| *bw* | Fine-tuning range. |

Step 2: Initialize the HM. As shown in Eq (1), randomly generate *HMS* harmonies and store the harmonies in the HM. In addition, a column is added to the HM to store the objective function value.

$$x_{i,j} = L + (U - L) \times rand \tag{1}$$

where $x_{i,j}$ is the *j*th dimension value of the ith harmony, and *rand* is a random number between zero and one.

Step 3: Improvisation. Generate a new harmony so that the HM can dynamically change. The specific process is shown below.

**Improvisation:** Generate a new harmony

```
for $j \leftarrow 1$ to $D$ do
    if $rand < HMCR$ then
        $x_{new,j} = x_{i,j} \quad i \in [1, \ldots, HMS]$
        if $rand < PAR$ then
            if $rand < 0.5$ then
                $x_{new,j} = x_{new,j} + rand \times bw$
            else
                $x_{new,j} = x_{new,j} - rand \times bw$
            end
        end
    else
        $x_{new,j} = L + (U - L) \times rand$
    end
end
```

where *D* is the dimension of the harmony.

Step 4: Update the HM. If the objective function of the new harmony is better than the worst harmony in the HM, the worst harmony will be replaced by the new harmony.

Step 5: Check the termination conditions. If not satisfied, repeat Steps 3 to 5; otherwise, the algorithm ends.

## 3. Parameter analysis

The elements in a harmony are affected by some random numbers; hence, the variance of the harmony vector in the HM is a random variable, and the expected variance will be a measure of the algorithm's exploration ability. The larger the expected variance is, the stronger the global search ability; conversely, the smaller the expected variance is, and the stronger the local search ability [18]. After the HM is updated, the new harmony generated during the improvisation stage may change the variance of the harmony in the HM. To avoid falling into the local optimum and to ensure that the searched area is as wide as possible, the parameters must be able to adjust the overall variance to achieve a proper balance between the global search and local search. In the HS algorithm, because the disturbance of each decision variable in the harmony is independent, the analysis of one-dimensional vectors does not lose generality and can be extended to multiple dimensions. Literature [24] analyzes the influence of the parameters in the HS algorithm on the exploration ability of the algorithm when L and U are symmetric about zero. To improve the universality of the conclusion in [24], the following theorem can be obtained.

Theorem 1: Assume that the harmony variables in the initial HM are $X = \{x_1, x_2, \ldots, x_{HMS}\}$. After several improvisation stages, the harmony variables in the HM are $Y = \{y_1 , y_2, \ldots, y_{HMS}\}$, and the expected variance of Y can be described as Eq (2):

$$\begin{aligned} E(Var(Y)) =& \frac{HMS-1}{HMS} \times [HMCR \times Var(x) + HMCR \times (1-HMCR) \times \bar{x}^2 \\ & \frac{1}{3} \times HMCR \times PAR \times bw^2 + (1-HMCR) \times (\frac{(U-L)^2}{12} \\ & + HMCR \times \frac{(U+L)\times(U+L-1)}{4})] \end{aligned} \tag{2}$$

where $Var(x)$ is the variance of the original harmony variable set, and $\bar{x}^2$ is the square of the mean of the original harmony variable set.

Proof: According to the literature [24], we have

$$E(x_r) = \bar{x} \tag{3}$$

$$E(x_r^2) = \bar{x^2} \tag{4}$$

$$\begin{aligned} E(Y_l) =& HMCR \times (1-PAR) \times E(x_r) + 0.5 \times HMCR \times PAR \times \\ & E(x_r + bw \times R) + 0.5 \times HMCR \times PAR \times E(x_r - bw \times R) \\ & (1-HMCR) \times E(x_{new}) \end{aligned} \tag{5}$$

$$\begin{aligned} E(Y_l^2) =& HMCR \times (1-PAR) \times E(x_r^2) + 0.5 \times HMCR \times PAR \times \\ & E((x_r + bw \times R)^2) + 0.5 \times HMCR \times PAR \times E((x_r - bw \times R)^2) \\ & (1-HMCR) \times E((x_{new})^2) \end{aligned} \tag{6}$$

$$E(\overline{Y^2}) = \frac{1}{HMS} \times \sum_{l=1}^{HMS} E(Y_l^2) \tag{7}$$

$$E(\bar{Y}^2) = \frac{1}{HMS} \times E(\overline{Y^2}) + \frac{HMS-1}{HMS} \times E(Y_l)^2 \tag{8}$$

$$E(Var(Y)) = E(\overline{Y^2}) - E(\bar{Y}^2) \tag{9}$$

According to the step of randomly generating the harmony vector, we have

$$x_{new} = L + (U-L) \times R \tag{10}$$

$$x_{new}^2 = L^2 + 2 \times L \times (U-L) \times R + ((U-L) \times R)^2 \tag{11}$$

where $R$ represents a uniformly distributed stochastic number between 0 and 1. $R$ is uniformly distributed in the interval $[0, 1]$. Then,

$$\varphi(R) = \begin{cases} 1, & R \in [0,1] \\ 0, & \text{others} \end{cases} \tag{12}$$

According to the continuous stochastic variable expectation calculation formula,

$$E(R) = \int_0^1 R \times \varphi(R)dR = \frac{1}{2} \tag{13}$$

$$E(R^2) = \int_0^1 R^2 \times \varphi(R)dR = \frac{1}{3} \tag{14}$$

Therefore,

$$E(x_{new}) = L + (U-L) \times E(R) = \frac{U+L}{2} \tag{15}$$

$$\begin{aligned} E(x_{new}^2) &= L^2 + 2 \times L \times (U-L) \times E(R) + (U-L)^2 \times E(R^2) \\ &= \frac{U^2 + U \times L + L^2}{3} \end{aligned} \tag{16}$$

Then, from Eqs (5) and (6), we get Eqs (17) and (18) as

$$E(Y_l) = HMCR \times \bar{x} + (1-HMCR) \times \frac{U+L}{2} \tag{17}$$

$$\begin{aligned} E(Y_l^2) =& HMCR \times \overline{x^2} + (1-HMCR) \times \frac{U^2 + U \times L + L^2}{3} \\ &+ HMCR \times PAR \times \frac{bw^3}{3} \end{aligned} \tag{18}$$

From Eqs (7) and (8), we get

$$\begin{aligned} E(\overline{Y^2}) =& HMCR \times \overline{x^2} + (1-HMCR) \times \frac{U^2 + U \times L + L^2}{3} \\ &+ HMCR \times PAR \times \frac{bw^3}{3} \end{aligned} \tag{19}$$

$$\begin{aligned} E(\bar{Y}^2) = & \frac{1}{HMS} \times E(\bar{Y}^2) + \frac{HMS-1}{HMS} \times [HMCR \times \bar{x} \\ & + (1-HMCR) \times \frac{U+L}{2}]^2 \end{aligned} \tag{20}$$

Therefore, Eq (9) can be further described as Eq (21)

$$\begin{aligned} E(Var(Y)) = & \frac{HMS-1}{HMS} \times [HMCR \times Var(x) + HMCR \times (1-HMCR) \times \bar{x}^2 \\ & \frac{1}{3} \times HMCR \times PAR \times bw^2 + (1-HMCR) \times (\frac{(U-L)^2}{12} \\ & + HMCR \times \frac{(U+L) \times (U+L-1)}{4})] \end{aligned} \tag{21}$$

and the theorem is proved.

From Theorem 1, *HMS* has a limited impact on $E(Var(Y))$. For example, assuming $HMS = 10$, then $(HMS-1)/HMS = 0.9$; and assuming $HMS = 100$, then $(HMS-1)/HMS = 0.99$. Therefore, choosing a smaller suitable *HMS* helps to reduce the calculation but can obtain similar results with larger ones. The larger the *HMCR* is, the greater the influence of $Var(x)$ and $bw^2$ on $E(Var(Y))$; therefore, the process is most likely to select a harmony in the HM as the basis of new harmony that is generated. The smaller the *HMCR* is, the greater the influence of $(U-L)^2$on $E(Var(Y))$, and the new harmony is most likely to be generated randomly. The larger *PAR* is, the greater the influence of $bw^2$ on $E(Var(Y))$ and the greater the disturbance of the harmony vector. The influences of the parameters on the algorithm are shown in Table 2.

**Table 2.** The influence of the parameters on the algorithm.

| Parameters | Influence |
|---|---|
| *HMS* | *HMS* has a limited impact on $E(Var(Y))$. |
| *HMCR* | A larger *HMCR* is beneficial to the local search, and a smaller *HMCR* is beneficial to increasing the diversity of the HM. |
| *PAR* | A larger *PAR* is beneficial to improving the precision, and a smaller *PAR* makes the algorithm stable. |
| *bw* | A larger *bw* makes the fine-tuning ability lower, and a wider area can be searched; A smaller *bw* helps to precisely fine-tune the algorithm. |
| *U, L* | A wider search range can increase the diversity of the HM, and a smaller search range can improve local search capabilities. |

## 4. Adaptive harmony search utilizing differential evolution and opposition-based learning

Based on the above analysis, this paper improves the HS algorithm from the following three aspects: 1) adaptively adjusts the *HMCR*, *PAR* and search domain with the iterations of the algorithm; 2) let the generated *bw* be related to the harmonies in the HM, which makes the new harmony approach the best harmony in the HM; and 3) to ensure that the algorithm does not fall into the local optimum when the search domain is reduced, the outside domain can also be searched. The corresponding improvement strategies are as follows: (i) dynamic parameter adjustment strategy, including the *HMCR*, *PAR* and search domain; (ii) *bw* generation strategy based on differential evolution; and (iii) new harmony generation strategy based on opposition-based learning. The three strategies are analyzed as follows. The main modifications are shown in Table 3.

**Table 3.** The main modifications of AHS-DE-OBL.

| Strategy | Measures |
|---|---|
| Dynamic parameter adjustment strategy | As shown in Algorithm 2, the *HMCR* and *PAR* are dynamically adjusted with the iteration; And, add four matrices $x_{maxbound}$, $x_{minbound}$, $x_{U_{new}}$ and $x_{L_{new}}$, that represent the maximum and minimum values in the current HM and the upper and lower boundary of the new search domain, respectively. When the algorithm is iterated, the four equations to calculate each dimension are Eqs (24–27), respectively. |
| *bw* generation strategy | *bw* is replaced by the distance between 3 harmonies in the HM. |
| New harmony generation strategy | After improvisation, switch the best and worst harmony in the HM. |

## 4.1. Analysis of the dynamic parameter adjustment strategy

In the early stage, the search domain should be as wide as possible to avoid becoming stuck in the local optimum, the *HMCR* needs to set to a small value, and the *PAR* needs to be as large as possible. However, in the later iteration process, the probability of fine-tuning should be increased to improve the search precision, and the *HMCR* should be as large as possible. Considering the stability, the *PAR* should be appropriately reduced to gradually make the harmonies in the HM approach the best harmony and increase the probability of fine-tuning the best harmony. The adjustment algorithm is as follows.

**Algorithm:** Dynamic parameter adjustment algorithm

```
if $gn < NI/4$ then
    $HMCR = 0.3 + 0.6 \times \frac{gn}{NI}$
    $PAR = 0.99$
else
    $HMCR = 0.9$
    $PAR = 0.99 - 0.09 \times \frac{gn}{NI}$
end
```

where *gn* is the current iteration number, and *NI* is the maximum number of iterations.

With the iterations, gradually narrowing the search boundary can make the algorithm focus on the optimal search domain. [18] proposes a linear dynamic domain mode that calculates the new boundary by utilizing the maximum and minimum values of the harmony to gradually change the boundary. For each dimension, the new boundary is calculated by the following equations.

The initial equations are:

$$x_{U_{new,j}} = U \tag{22}$$

$$x_{L_{new,j}} = L \tag{23}$$

The equations during iteration are:

$$x_{maxbound,j} = max(HM(:, j)) \tag{24}$$

$$x_{minbound,j} = min(HM(:, j)) \tag{25}$$

$$x_{U_{new,j}} = (1 - \frac{gn}{NI}) \times x_{U_{new,j}} + \frac{gn}{NI} \times x_{maxbound,j} \tag{26}$$

$$x_{L_{new,j}} = (1 - \frac{gn}{NI}) \times x_{L_{new,j}} + \frac{gn}{NI} \times x_{minbound,j} \tag{27}$$

where $x_{maxbound,j}$ and $x_{minbound,j}$ are the maximum and minimum values of the $j$th dimension in the HM, respectively, and $x_{U_{new,j}}$ and $x_{L_{new,j}}$ are the upper and lower boundary values of the new search domain of the $j$th dimension, respectively.

### *4.2. Analysis of the bw generation strategy based on differential evolution*

Differential evolution is an efficient population-based heuristic algorithm used to find the global optimal value through cooperation and competition between individuals in the population [22]. According to this idea, the adaptive ability of bw can be described by Eq (28) as follows.

$$bw = x_{best,j} - x_{i,j} + x_{best,j} - x_{worst,j} \tag{28}$$

where $x_{best,j}$ is the value of the $j$th dimension of the best harmony in the current HM, $x_{i,j}$ is the value of the $j$th dimension of a random harmony in the current HM, and $x_{worst,j}$ is the value of the $j$th dimension of the worst harmony in the current HM. Assuming each harmony is 2-dimensional, the geometric meaning of Eq (28) is shown in Figure 1.

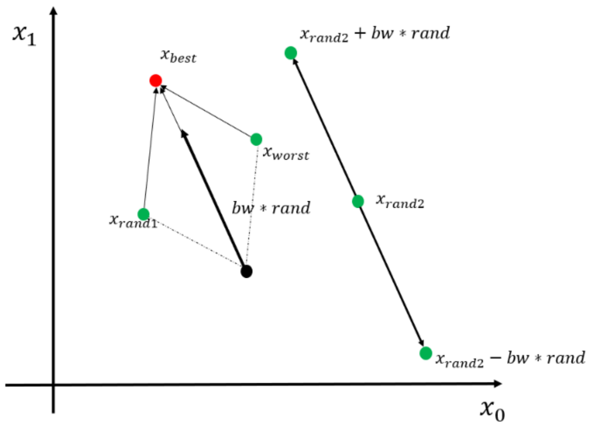


**Figure 1.** The geometric meaning of the bw generation strategy.

where $x_0$ and $x_1$ respectively represent the 1st dimension and the 2nd dimension, $x_{best}$ is the best harmony in the HM, $x_{rand1}$ and $x_{rand2}$ are randomly selected harmonies from the HM, $x_{new} = x_{rand2} \pm bw \times rand$, and $rand \in (0, 1)$. $x_{new}$ has a 50 percent probability of $x_{rand2} + bw \times rand$ and a 50 percent probability of $x_{rand2} - bw \times rand$. In the case of Figure 1, $x_{new}$ has a certain probability of approaching $x_{best}$ or moving away from $x_{best}$. This strategy ensures that $x_{new}$ does not blindly approach $x_{best}$ and is helpful to increasing the diversity of the population.

## *4.3. Analysis of new harmony generation strategy based on opposition-based learning*

Although the above strategies can improve the local search ability of the algorithm, they may fall into the local optimum and cannot jump out. And the random generation of a solution from the current population often leads to revisiting the hopeless areas in the search space [25]. During the search process, generating a random solution and its opposition solution can search areas outside the new search domain and improve the efficiency of the algorithm [23]. In the improvisation stage, two extra opposition solutions are generated by Eqs (29) and (30).

$$x'_{new,j} = L + (U - x_{worst,j}) \tag{29}$$

$$x''_{new,j} = L + (U - x_{best,j}) \tag{30}$$

where $x'_{new,j}$ and $x''_{new,j}$ express the *j*th dimension of the opposition solutions.

Assuming that the harmony has a dimension of two, according to $x_{best,j}$, the opposition solution is generated, as shown in Figure 2. When the new search area is the shaded part of Figure 2, the algorithm can still search for areas outside the new search domain.

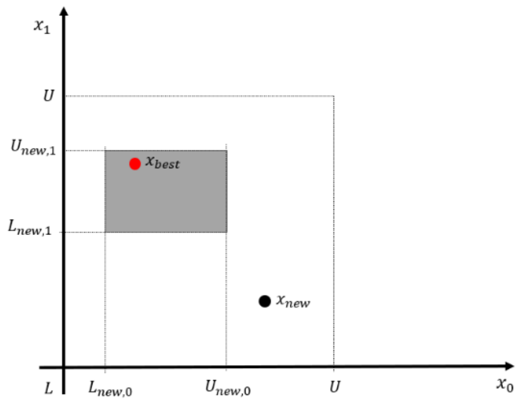


**Figure 2.** The geometric meaning of the new harmony generation strategy.

## *4.4. Description of the AHS-DE-OBL algorithm*

In the early stage, the search domain should be as wide as possible The AHS-DE-OBL algorithm can be described by the following steps.

Step 1: Initialize the parameters. The pseudocode is as follows.

**Initialization:** Initialize the parameters

```
for $j \leftarrow 1$ to $D$ do
    $x_{maxbound,j} = U$
    $x_{minbound,j} = L$
    $x_{U_{new},j} = U$
    $x_{L_{new},j} = L$
end
```

Step 2: Initialize the HM. Randomly generate *HMS* harmonies and store them in the HM.

Step 3: Improvisation. Generate three harmonies randomly. The pseudocode is as follows. In the code, $x_{U_{new},j}$ and $x_{L_{new},j}$ are the upper and lower bounds of the *j*th dimension, respectively. *HMCR* and *PAR* are dynamically adjusted with the number of iterations; and *bw* is generated by the best, the worst, and a random harmony.

**Algorithm:** Generate three harmonies

```
if $gn < NI/4$ then
    $HMCR = 0.3 + 0.6 \times \frac{gn}{NI}$
    $PAR = 0.99$
else
    $HMCR = 0.9$
    $PAR = 0.99 - 0.09 \times \frac{gn}{NI}$
end
for $j \leftarrow 1$ to $D$ do
    if $rand < HMCR$ then
        $x_{new,j} = x_{i,j} \quad i \in [1, \ldots, HMS]$
        if $rand < PAR$ then
            $bw = x_{best,j} - x_{rand,j} + x_{best,j} - x_{worst,j}$
            if $rand < 0.5$ then
                $x_{new,j} = x_{new,\ j} + rand \times bw$
            else
                $x_{new,j} = x_{new,\ j} - rand \times bw$
            end
        end
    else
        $x_{new,j} = x_{L_{new},j} + (x_{U_{new},j} - x_{L_{new},j}) \times rand$
    end
end
for $j \leftarrow 1$ to $D$ do
    $x'_{new,j} = L + (U - x_{worst,j})$
    $x''_{new,j} = L + (U - x_{best,j})$
end
```

Step 4: Update the HM. If the objective function value of the new harmony is better than that of the worst harmony, the worst harmony is replaced by the new harmony.

Step 5: Dynamically adjust the parameters. Update the parameters based on the current iteration number and harmony in the HM. The pseudocode is as follows.

**Algorithm:** Dynamic adjustment of the parameters

```
for $j \leftarrow 1$ to $D$ do
    $x_{maxbound,j} = max(HM(:, j))$
    $x_{minbound,j} = min(HM(:, j))$
    $x_{U_{new,j}} = (1 - \frac{gn}{NI}) \times x_{U_{new,j}} + \frac{gn}{NI} \times x_{maxbound,j}$
    $x_{L_{new,j}} = (1 - \frac{gn}{NI}) \times x_{L_{new,j}} + \frac{gn}{NI} \times x_{minbound,j}$
end
```

Step 6: Check the termination conditions. If the condition is not satisfied, repeat Step 3 to Step 6; otherwise, the algorithm ends.

### 4.5. *Time complexity analysis*

According to the steps of the algorithms, the time complexity of the HS algorithm and the AHS-DE-OBL algorithm is discussed below, where only one iteration is considered. In the HS algorithm, the initialization assignment of $HMS$ harmonies in the $N$-dimensional search space requires $N \times HMS$ operations, and $N$ operations are needed to calculate the fitness function of an individual. Thus, this stage's time complexity is O($N \times HMS$). The improvisation stage's time complexity is O($N$). The other stages' time complexity is O(1). In the AHS-DE-OBL, Step 5's time complexity is O($N \times HMS$), the other stages are the same as the HS algorithm. Therefore, in $T$ iterations, the time complexity of the HS algorithm is O($T \times HMS \times N$), and the AHS-DE-OBL is also. Theoretically, the improved algorithm does not increase the time complexity compared with the original algorithm.

**Table 4.** Ten benchmark functions used in the experiment.

| Function | Formula | D | Interval | GOV |
|---|---|---|---|---|
| Sphere | $F_1 = \sum_{i=1}^{D} x_i^2$ | 10,30 | [-100, 100] | 0 |
| Schwefel 2.21 | $F_2 = max\{\lvert x_i \rvert, 1 \leq i \leq D\}$ | 10,30 | [-100, 100] | 0 |
| Step | $F_3 = \sum_{i=1}^{D}(x_i + 0.5)^2$ | 10, 30 | [-100, 100] | 0 |
| Rastrigin | $F_4 = \sum_{i=1}^{D}(x_i^2 - 10 \times \cos(2\pi x_i) + 10)$ | 10,30 | [-5.12, 5.12] | 0 |
| Ackley | $F_5 = -20 \times exp(-0.2 \times \sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i^2)}) - exp(-0.2 \times \sqrt{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}) + 20 + e$ | 10,30 | [-32, 32] | 0 |
| Ackley shift | $F_6 = -20 \times exp(-0.2 \times \sqrt{\frac{1}{D}\sum_{i=1}^{D}((x_i - 1)^2)}) - exp(-0.2 \times \sqrt{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi(x_i - 1))} + 20 + e$ | 10, 30 | [-31, 33] | 0 |
| Griewank | $F_7 = \frac{1}{4000}\sum_{i=1}^{D}(x_i^2) - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$ | 10,30 | [-600, 600] | 0 |
| Matyas | $F_8 = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$ | 2 | [-10, 10] | 0 |
| Three Hump Camel | $F_9 = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$ | 2 | [-5, 5] | 0 |
| Drop Wave | $F_{10} = -\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5(x_1^2+x_2^2)+2}$ | 2 | [-5.12, 5.12] | -1 |

*Note: GOV is the global optimal value.*

## 5. Experiments

### 5.1. *Test functions used in the experiment*

The experiment is conducted using a computer with an Intel ® Core ™ i5 – 3470 CPU @ 3.20 GHz, 4 GB memory, the Windows 10 operating system, and the Python 3.7 programming environment. The ten benchmark functions used in the experiment are listed in Table 4.

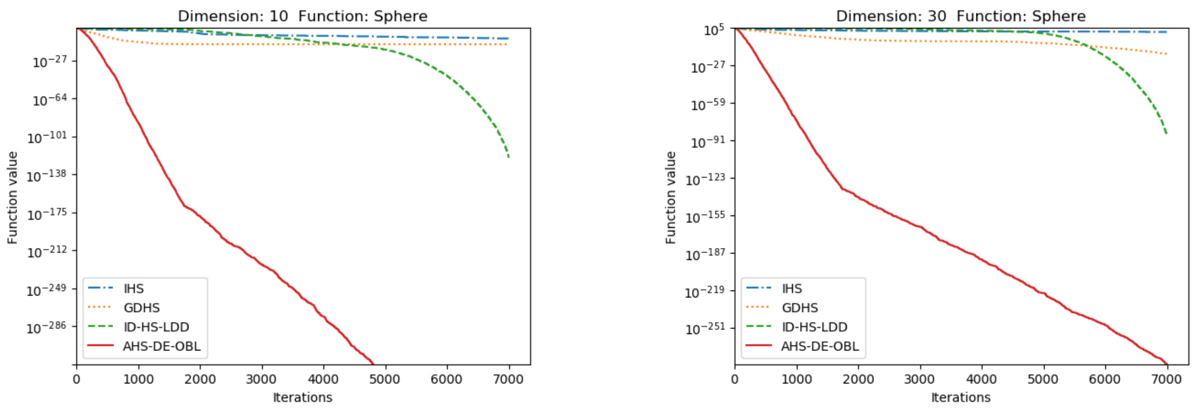
### 5.2. *Experiment results and discussions*

To verify the performance of AHS-DE-OBL, it was compared with IHS, GDHS, ID-HS-LDD using ten benchmark functions. Each algorithm was run 30 times independently with 7000 iterations per run. By referring to the relevant literature [15, 17, 18] of the selected algorithms, the parameter settings are shown in Table 5.

**Table 5.** Parameter setting for each HS variant.

| Algorithm | Parameters |
|---|---|
| IHS | $HMS = 5, HMCR = 0.95, PAR_{min} = 0.01, PAR_{max} = 0.99$ $bw_{min} = 0.001$,and $bw_{max} = (U - L)/20$ |
| GDHS | $HMS = 5$ |
| ID-HS-LDD | $HMS = 30, HMCR_{max} = 0.99, HMCR_{min} = 0.3,$ $PAR_{min} = 0.9, PAR_{max} = 0.1$ |
| AHS-DE-OBL | $HMS = 5$ |

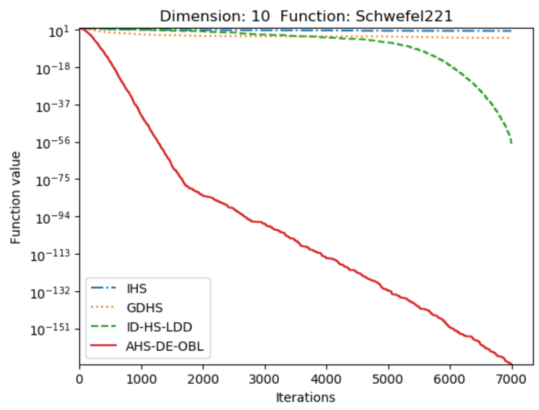
Figures 3–10 show the optimization curves of the ten test functions.



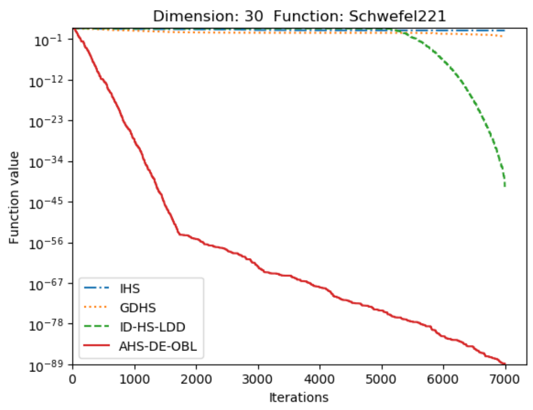
**(a)** 10-dimension situation

**(b)** 30-dimension situation

**Figure 3.** Performance comparison on F1.

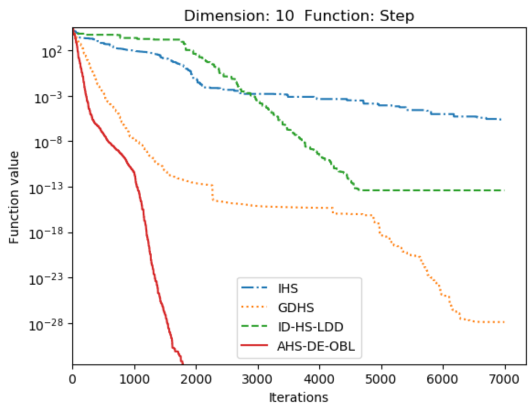


**(a)** 10-dimension situation

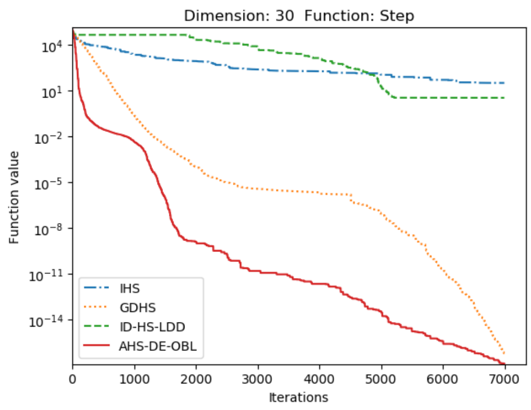


**(b)** 30-dimension situation

**Figure 4.** Performance comparison on F2.

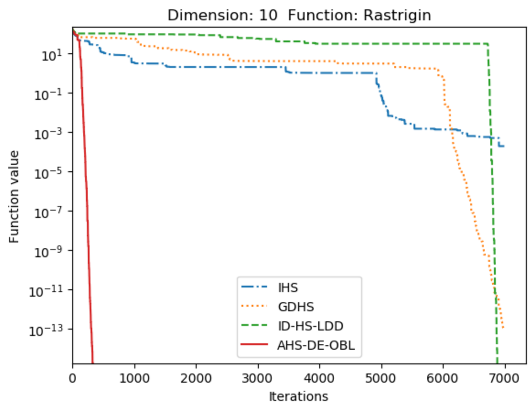


**(a)** 10-dimension situation

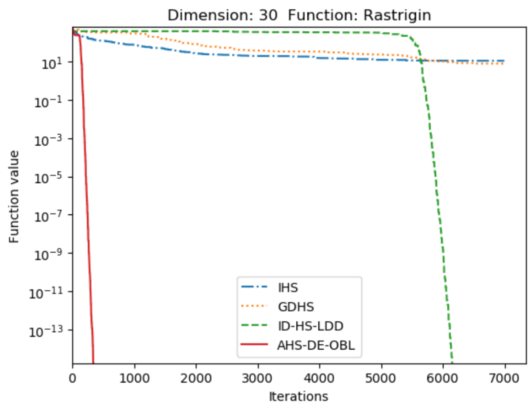


**(b)** 30-dimension situation

**Figure 5.** Performance comparison on F3.

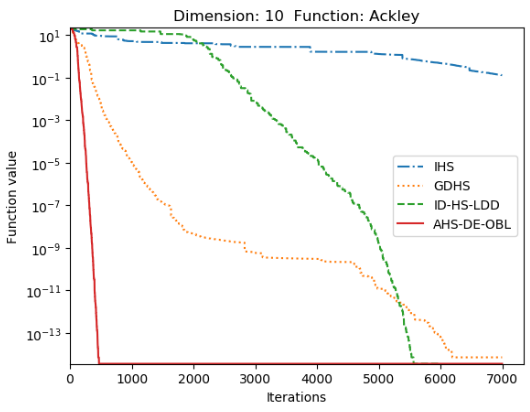


**(a)** 10-dimension situation

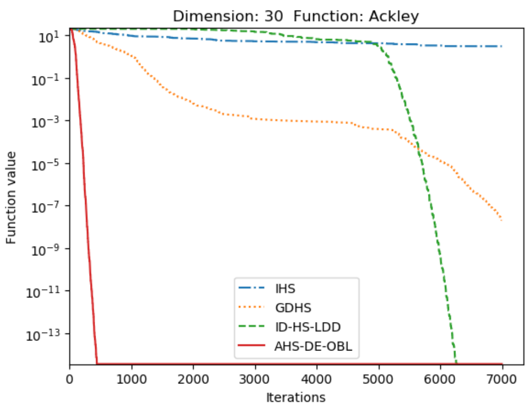


**(b)** 30-dimension situation

**Figure 6.** Performance comparison on F4.

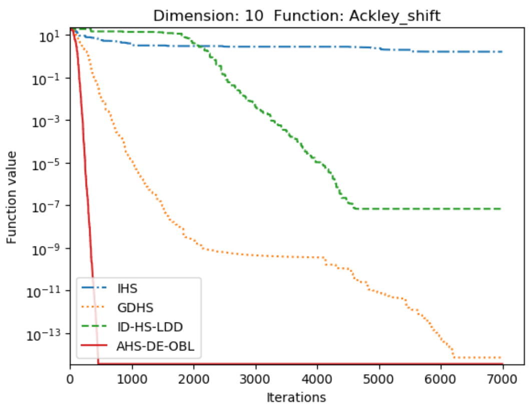


**(a)** 10-dimension situation

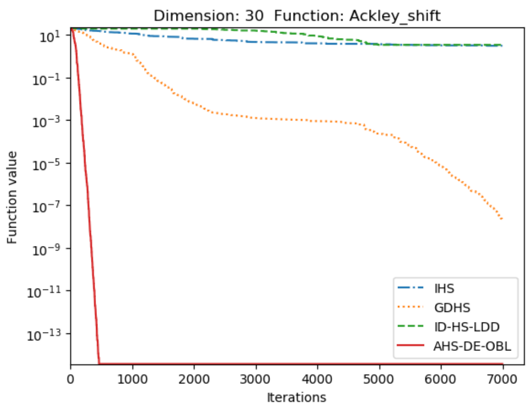


**(b)** 30-dimension situation

**Figure 7.** Performance comparison on F5.

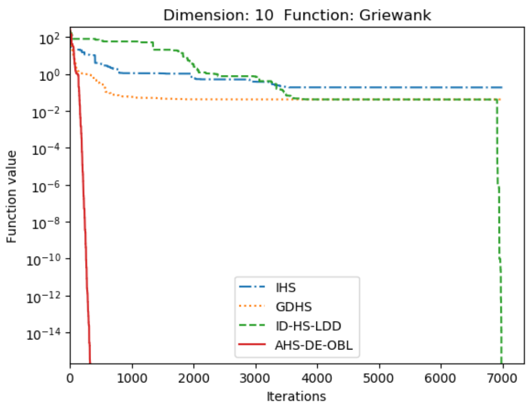


**(a)** 10-dimension situation

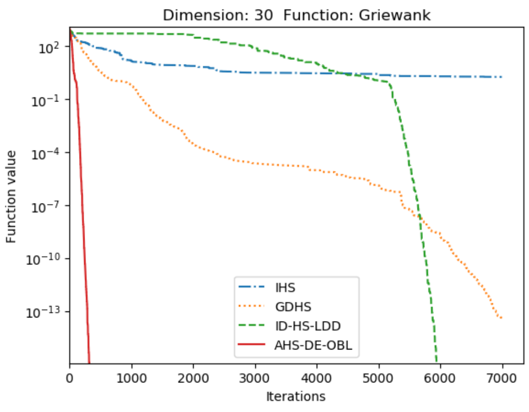


**(b)** 30-dimension situation

**Figure 8.** Performance comparison on F6.

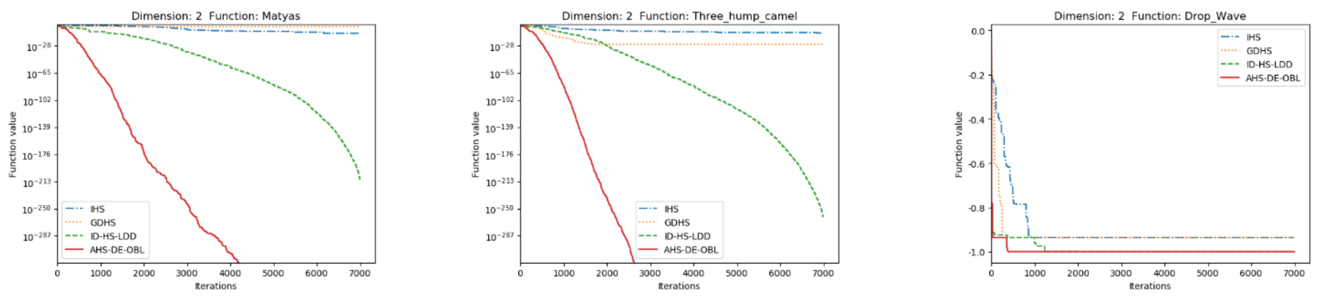


**(a)** 10-dimension situation



**(b)** 30-dimension situation

**Figure 9.** Performance comparison on F7.



**(a)** F8 **(b)** F9 **(c)** F10

**Figure 10.** Performance comparison on F8, F9, and F10.

Table 6 shows the results of the four HS variants running 30 times for 10 functions. The values in bold font in Table 6 show the optimal values among all algorithms.

**Table 6.** Experimental results.

| | | IHS | GDHS | ID-HS-LDD | Ours |
|---|---|---|---|---|---|
| | D | Mean±std | Mean±std | Mean±std | Mean±std |
| F1 | 10 | 1.06E-02±5.47E-02 | 4.47E-28±1.84E-27 | 2.04E-124±4.37E-124 | **0.00±0.00** |
| | 30 | 8.52E+01±3.32E+01 | 5.95E-12±8.22E-12 | 8.24E-85±1.56E-84 | **6.51E-255±0.00** |
| F2 | 10 | 3.55±1.32 | 6.35E-04±2.98E-03 | 7.75E-57±2.64E-57 | **6.86E-161±3.69E-160** |
| | 30 | 1.77E+01±2.72 | 1.28±7.11E-01 | 6.60E-40±9.78E-40 | **7.77E-83±4.03E-82** |
| F3 | 10 | 1.62E-06±3.99E-07 | 2.93E-18±1.57E-17 | 3.64E-17±9.10E-16 | **1.64E-33±1.90E-33** |
| | 30 | 8.91E+01±4.25E+01 | 5.14E-12±6.12E-12 | 2.45±6.04E-01 | **1.94E-14±1.01E-13** |
| F4 | 10 | 3.17E-01±4.92E-01 | 3.48E-02±1.85E-01 | **0.00±0.00** | **0.00±0.00** |
| | 30 | 8.78±2.74 | 6.77±2.77 | **0.00±0.00** | **0.00±0.00** |
| F5 | 10 | 1.30±8.14E-01 | 1.50E-10±8.09E-10 | **0.00±0.00** | 3.52E-15±1.21E-15 |
| | 30 | 3.09±4.16E-01 | 1.76E-08±2.79E-08 | **0.00±0.00** | 4.23E-15±8.86E-16 |
| F6 | 10 | 1.13±6.60E-01 | 3.61E-10±1.94E-09 | 2.53E-09±3.04E-09 | **2.93E-15±1.63E-15** |
| | 30 | 3.18±4.23E-01 | 2.14E-08±2.07E-08 | 3.03±4.20E-01 | **4.24E-15±1.21E-15** |
| F7 | 10 | 4.08E-01±2.07E-01 | 4.10E-02±2.35E-02 | **0.00±0.00** | **0.00±0.00** |
| | 30 | 1.80±2.39E-01 | 4.51E-03±6.79E-03 | **0.00±0.00** | **0.00±0.00** |
| F8 | 2 | 3.85E-03±1.02E-02 | 3.32E-02±7.22E-02 | 4.36E-204±0.00 | **0.00±0.00** |
| F9 | 2 | 8.96E-02±1.37E-01 | 1.04E-01±3.27E-01 | 1.88E-261±0.00 | **0.00±0.00** |
| F10 | 2 | -9.27E-01±6.01E-02 | -9.19E-01±7.10E-02 | **-1.00±0.00** | **-1.00±0.00** |

As seen from Figures 3–10 and Table 6, AHS-DE-OBL has better global search abilities and a faster convergence speed than the other three HS variants.

In AHS-DE-OBL, three strategies are adopted. The strategy of opposition-based learning can improve the search efficiency of the solution space, the bw generation strategy based on differential evolution, and the adaptive parameter adjustment strategy can improve the precision. Therefore, for most of the selected test functions, AHS-DE-OBL has a faster convergence speed and higher accuracy than the others. For unimodal functions F1 (a convex and bowl-shaped unimodal function), F2 (a tapering-shaped function), F3 (a bowl-shaped function with the optimal solution not at position zero), and F8 (a plate-shaped function), regardless of whether there are 2, 10 or 30 dimensions, AHS-DE-OBL has better performance than the other three methods. For multimodal functions F4 (a multimodal function with a regular distribution of the local minima), F6 (a nearly flat outer region and a large hole at the center multimodal function), F7 (a multimodal function with widely distributed local minima), F9 (a valley-shaped multimodal function), and F10 (a multimodal and highly complex function), which can detect the ability of the algorithm to jump out of the local optimum, AHS-DE-OBL obtains all of the best results. For F5, because ID-HS-LDD uses a forced convergence strategy, which forces the value of the harmony to approach zero in a later iteration stage, it can find the optimal value of F5. However, for F6, which is obtained by moving F5 one unit in the positive direction, the performance of ID-HS-LDD was significantly reduced, and AHS-DE-OBL still maintained fairly good performance. For the 30-dimensional F3, ID-HS-LDD also has a bad optimization result.

To verify the improved algorithm does not increase the time complexity compared with the original algorithm, the execution time of the four algorithms in the same experimental environment is recorded. As shown in Tables 6 and 7, although it takes less time to execute IHS and GDHS than the proposed algorithm, their optimization accuracy is lower than the proposed algorithm. The accuracy of ID-HS-LDD is the same as the proposed algorithm, however, it takes more time to execute than the proposed algorithm.

**Table 7.** Execution time of the four algorithms.

| | | IHS | GDHS | ID-HS-LDD | Ours |
|---|---|---|---|---|---|
| | D | Time | Time | Time | Time |
| F1 | 10 | 0.52 s | 0.48 s | 1.43 s | 0.87 s |
| | 30 | 0.98 s | 0.83 s | 4.47 s | 2.02 s |
| F2 | 10 | 0.50 s | 0.47 s | 1.43 s | 0.72 s |
| | 30 | 0.73 s | 0.76 s | 3.83 s | 1.78 s |
| F3 | 10 | 0.53 s | 0.49 s | 1.45 s | 0.79 s |
| | 30 | 0.77 s | 0.79 s | 3.99 s | 1.82 s |
| F4 | 10 | 0.57 s | 0.57 s | 1.51 s | 0.94 s |
| | 30 | 0.97 s | 0.95 s | 4.36 s | 2.45 s |
| F5 | 10 | 0.74 s | 0.65 s | 1.68 s | 1.28 s |
| | 30 | 1.08 s | 1.13 s | 4.55 s | 2.90 s |
| F6 | 10 | 0.67 s | 0.65 s | 1.69 s | 1.22 s |
| | 30 | 1.13 s | 1.27 s | 4.98 s | 2.95 s |
| F7 | 10 | 0.57 s | 0.55 s | 1.53 s | 0.93 s |
| | 30 | 0.90 s | 0.96 s | 4.15 s | 2.33 s |
| F8 | 2 | 0.39 s | 0.36 s | 0.40 s | 0.29 s |
| F9 | 2 | 0.40 s | 0.35 s | 0.41 s | 0.32 s |
| F10 | 2 | 0.41 s | 0.36 s | 0.42 s | 0.31 s |

To further verify the performance of AHS-DE-OBL, it was compared with Sine Cosine Algorithm (SCA), self-adaptive DE (SaDE), and multi-scale cooperative mutatingly self-adaptive escape PSO (MAEPSO) using ten benchmark functions. Each algorithm was run 30 times independently with 7000 iterations per run. By referring to the relevant literature [18, 26, 27] of the selected algorithms, the parameter settings are shown in Table 8.

**Table 8.** Parameter Setting for selected metaheuristic algorithms.

| Algorithm | Parameters |
|---|---|
| SCA | $populationsize = 50$ |
| SaDE | $populaitonsize = 20, crm = 0.5, learningPeriod = 50, crPeriod = 5,$ $crmUpdatePeriod = 25, p = 0.5$ |
| MAEPSO | $M = 5, subP = 20, c1 = 1.4, c2 = 1.4,$ $k1 = 5, k2 = 10$ |

where *populaitonsize* is population size, *crm* is the initial value of adaptive crossover rate, *learningPeriod* is the update period of parameter $p$, $p$ is the selection probability of mutation strategy, *crPeriod* controls the update frequency of the crossover rate for each individual, *crmUpdatePeriod* controls the update frequency of the parameter *crm*, $M$ is the number of multi-scale Gaussian mutation operators, *subP* is the number of individuals in a subgroup, $c1$ and $c2$ is the learning factor, $k1$ and $k2$ is the escape threshold. Table 9 shows the experimental reuslts.

**Table 9.** Experimental results.

| | | SCA | SaDE | MAEPSO | Ours |
|---|---|---|---|---|---|
| | D | Mean±std | Mean±std | Mean±std | Mean±std |
| F1 | 10 | 7.88E-230±0.00 | 5.00E-324±0.00 | 7.58E-82±1.50E-81 | **0.00±0.00** |
| | 30 | 6.27E-42±1.88E-41 | 6.26E-86±1.88E-85 | 3.28E-28±6.54E-28 | **6.51E-255±0.00** |
| F2 | 10 | 1.67E-72±4.99E-72 | 6.47E-35±1.15E-34 | 2.12E-06±6.03E-06 | **6.86E-161±3.69E-160** |
| | 30 | 8.21E-02±2.39E-01 | 5.63E-04±2.87E-04 | 1.09±6.56E-01 | **7.77E-83±4.03E-82** |
| F3 | 10 | 1.36E-01±8.22E-02 | **0.00±0.00** | **0.00±0.00** | 1.64E-33±1.90E-33 |
| | 30 | 3.50±2.52E-01 | **0.00±0.00** | 6.37E-26±1.35E-25 | 1.94E-14±1.01E-13 |
| F4 | 10 | **0.00±0.00** | **0.00±0.00** | **0.00±0.00** | **0.00±0.00** |
| | 30 | **0.00±0.00** | **0.00±0.00** | 2.44E-08±4.86E-08 | **0.00±0.00** |
| F5 | 10 | **1.42E-15±1.74E-15** | 3.55E-15±0.00 | 6.04E-15±1.63E-15 | 3.52E-15±1.21E-15 |
| | 30 | 1.31E+01±7.89 | 6.75E-15±1.07E-15 | 7.65E-12±1.48E-11 | **4.23E-15±8.86E-16** |
| F6 | 10 | 2.52±2.36E-01 | 3.20E-15±1.07E-15 | 5.33E-15±1.78E-15 | **2.93E-15±1.63E-15** |
| | 30 | 1.17E+01±8.02 | 7.12E-15±0.00 | 8.72E-12±2.60E-11 | **4.24E-15±1.21E-15** |
| F7 | 10 | **0.00±0.00** | 6.65E-03±4.93E-03 | 9.69E-02±6.18E-02 | **0.00±0.00** |
| | 30 | 4.61E-03±1.38E-02 | **0.00±0.00** | 1.18E-02±8.15E-03 | **0.00±0.00** |
| F8 | 2 | **0.00±0.00** | **0.00±0.00** | 4.88E-221±0.00 | **0.00±0.00** |
| F9 | 2 | **0.00±0.00** | **0.00±0.00** | 2.35E-215±0.00 | **0.00±0.00** |
| F10 | 2 | **-1.00 ± 0.00** | **-1.00±0.00** | **-1.00±0.00** | **-1.00±0.00** |

As shown in Table 9, except for F3 and F5 of 10 dimensions, the proposed algorithm has achieved better results than other methods. About the SCA, it has obtained better values in seven cases, among which the F5 with 10-dimension is better than the proposed algorithm, but there is little difference in accuracy between them. For the SaDE, it has obtained better values in eight cases, among which the F3 is better than the proposed algorithm. However, it performs better in F7 with 30-dimension than in F7 with 10-dimension, showing its instability. While for the MAEPSO, it has obtained better values in three cases, among which the F3 with 10-dimension is better than the. But it performs relatively worse for the other cases.

To further analyze the experimental results, the Wilcoxon test is used, and Table 10 shows the results. Where the Wilcoxon column represents the comparison result of the selected algorithms and the proposed algorithm, '+' means the proposed algorithm is better than the selected algorithms in this case, '-' is the opposite. '≈' means the two algorithms achieve the same results. the Rank column is the ranking of their mean solution accuracy. For the selected test function, it can be seen that the performance of IHS and GDHS is not good, therefore SCA, SaDE, MAEPOS, and ID-HS-LDD are chosen to compare with the proposed algorithm.

**Table 10.** The Wilcoxon test results.

| | | SCA | | SaDE | | MAEPSO | | ID-HS-LDD | | Ours |
|---|---|---|---|---|---|---|---|---|---|---|
| | D | Wilcoxon | Rank | Wilcoxon | Rank | Wilcoxon | Rank | Wilcoxon | Rank | Rank |
| F1 | 10 | + | 3 | + | 2 | + | 5 | + | 4 | 1 |
| | 30 | + | 4 | + | 2 | + | 5 | + | 3 | 1 |
| F2 | 10 | + | 2 | + | 4 | + | 5 | + | 3 | 1 |
| | 30 | + | 5 | + | 3 | + | 4 | + | 2 | 1 |
| F3 | 10 | + | 5 | - | 1 | - | 1 | + | 4 | 3 |
| | 30 | + | 5 | - | 2 | - | 1 | + | 4 | 3 |
| F4 | 10 | ≈ | 1 | ≈ | 1 | ≈ | 1 | ≈ | 1 | 1 |
| | 30 | ≈ | 1 | ≈ | 1 | + | 5 | ≈ | 1 | 1 |
| F5 | 10 | - | 1 | + | 4 | + | 5 | - | 2 | 3 |
| | 30 | + | 5 | + | 3 | + | 4 | - | 1 | 2 |
| F6 | 10 | + | 5 | + | 2 | + | 3 | + | 4 | 1 |
| | 30 | + | 5 | + | 2 | + | 3 | + | 4 | 1 |
| F7 | 10 | ≈ | 1 | + | 4 | + | 5 | ≈ | 1 | 1 |
| | 30 | + | 4 | ≈ | 1 | + | 5 | ≈ | 1 | 1 |
| F8 | 2 | ≈ | 1 | ≈ | 1 | + | 4 | + | 5 | 1 |
| F9 | 2 | ≈ | 1 | ≈ | 1 | + | 5 | + | 4 | 1 |
| F10 | 2 | ≈ | 1 | ≈ | 1 | ≈ | 1 | ≈ | 1 | 1 |
| **Ave** | | | 2.94 | | 2.01 | | 3.65 | | 2.65 | 1.41 |
| **Final** | | | 4 | | 2 | | 5 | | 3 | 1 |

*Note:* ***Ave*** *is average rank, and* ***Final*** *is finally rank.*

As shown in Table 10, for low-dimensional functions, the proposed algorithm can obtain the same results as some improved algorithms. As the dimensionality increases, in most cases, the performance of the proposed algorithm is significantly better than the selected improved algorithms.

## 6. Conclusions

Aiming to address the inherent shortcomings of the HS algorithm, such as its slow convergence speed and low search precision, an improved HS algorithm, named AHS-DE-OBL, was proposed. AHS-DE-OBL is based on differential evolution, opposition-based learning, and a search domain adaptive adjustment strategy. In the improvisation stage, which uses the best and worst harmonies to affect bw, an opposition-based learning strategy is used to increase the diversity of the harmony in HM. After improvisation is completed, the range of the search domain is dynamically adjusted to increase the search efficiency. The experimental results also show that AHS-DE-OBL has better robustness and a better adaptive ability than the three selected HS variants and the selected metaheuristic algorithms.

## Acknowledgments

This work was supported by the Natural Science Foundation of Hunan Province, China (Nos.2019JJ40234 and 2020JJ5458), the Research Foundation of Education Bureau of Hunan Province, China (No.19A414 and 18B317), the Special Project of Language and Writing Application Research of Hunan Provincial Language Commission (No. XYJ2019GB09), the National Natural Science Foundation of China (No.62066016), and Jishou University Graduate Research and Innovation Project (No. JGY202032).

## Conflict of interests

The authors declare there is no conflict of interest.

## References

1. D. E. Goldberg, *Genetic Algorithm in Search Optimization and Machine Learning*, Addison-Wesley Professional, 1989.
2. G. C. Chen, J. S. Yu, Particle swarm optimization algorithm, *Inf. Control*, **186** (2005), 454–458.
3. Z. W. Geem, J. H. Kim, G. V. Loganathan, A new heuristic optimization algorithm: harmony search, *Simulation*, **76** (2001), 60–68.
4. O. M. Alia, R. Mandava, The variants of the harmony search algorithm: an overview, *Artif. Intell. Rev.*, **36** (2011), 49-68.
5. T. Zhang, Z. W. Geem, Review of harmony search with respect to algorithm structur, *Swarm Evol. Comput.*, **48** (2019), 31–43.
6. M. Shaqfa, Z. Orbán, Modified parameter-setting-free harmony search (PSFHS) algorithm for optimizing the design of reinforced concrete beams, *Struct. Multidiplinary Optim.*, **60** (2019), 999–1019.
7. Y. Song, Q. Pan, L. Gao, B. Zhang, Improved non-maximum suppression for object detection using harmony search algorithm, *Appl. Soft Comput.*, **81** (2019), 105478.

8. A. A. Vasebi, B. M. Fesanghary, A. S. M. T. Bathaee, Combined heat and power economic dispatch by harmony search algorithm, *Int. J. Electr. Power Energy Syst.*, **29** (2007), 713–719.

9. Z. W. Geem, K. S. Lee, Y. Park, Application of harmony search to vehicle routing, *Am. J. Appl. Sci.*, **2** (2005), 1552–1557.

10. C. A. Christodoulou, V. Vita, G. C. Seritan, L. Ekonomou, A harmony search method for the estimation of the optimum number of wind turbines in a wind farm, *Energies*, **13** (2020), 2777.

11. M. Z. Mistarihi, R. A. Okour, G. M. Magableh, H. B. Salameh, Integrating advanced harmony search with fuzzy logic for solving buffer allocation problems, *Arabian J. Sci. Eng.*, **45** (2020), 3233–3244.

12. H. C. Li, K. Q. Zhou, L. P. Mo, A. M. Zain, F. Qin, Weighted fuzzy production rule extraction using modified harmony search algorithm and BP neural network framework, *IEEE Access*, **8** (2020), 186620–186637.

13. A. Soumen, S. P. Ranjan, M. Anirban, Solving tool indexing problem using harmony search algorithm with harmony refinement, *Soft Comput.*, **23** (2019), 7407–7423.

14. J. H. Yoon, Z. W. Geem, Empirical convergence theory of harmony search algorithm for box-constrained discrete optimization of convex function, *Mathematics*, **9** (2021), 545.

15. M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, *Appl. Math. Comput.*, **188** (2007), 1567–1579.

16. C. M. Wang, Y. F. Huang, Self-adaptive harmony search algorithm for optimization, *Expert Syst. Appl.*, **37** (2010), 2826–2837.

17. M. Khalili, R. Kharrat, K. Salahshoor, M. H. Sefat, Global dynamic harmony search algorithm: GDHS, *Appl. Math. Comput.*, **228** (2014), 195–219.

18. Q. Zhu, X. Tang, Y. Li, M. O. Yeboah, An improved differential-based harmony search algorithm with linear dynamic domain, *Knowl.-Based Syst.*, **187** (2020), 104809.

19. M. A. Al-Betar, A. T. A. Khader, F. Nadi, Selection mechanisms in memory consideration for examination timetabling with harmony search, in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, (2010), 1203–1210.

20. P. Chakraborty, G. G. Roy, S. Das, An improved harmony search algorithm with differential mutation operator, *Fundam. Informaticae*, **95** (2004), 401–426.

21. N. Taherinejad, Highly reliable harmony search algorithm, in *2009 European Conference on Circuit Theory and Design*, IEEE, (2009), 818–822.

22. R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997) 341–359.

23. H. R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, **1** (2005), 695–701.

24. S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B. K. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)*, **41** (2010), 89–106.

25. X. Ma, Q. Zhang, G. Tian, J. Yang, Z. Zhu, On Tchebycheff decomposition approaches for multiobjective evolutionary optimization, *IEEE Trans. Evol. Comput.*, **22** (2017), 226–244.

26. S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowl.-Based Syst.*, **96** (2016), 120–133.

27. X. M. Tao, F. R. Li, Z. J. Tong, Multi-Scale cooperative mutation particle swarm optimization algorithm, *J. Software*, **23** (2012), 1805–1815.