An Improved Harmony Search Algorithm with Segmented Search

Di-Wen Kang, Li-Ping Mo* and Kai-Qing Zhou

College of Information Science and Engineering, Jishou University, Jishou, 416000, Hunan, China

Email: zmx89@jsu.edu.cn

Abstract. Harmony search (HS) algorithm has received widely attention in various fields because of simple operation and outstanding performance. However, there exists some bottlenecks which hinders the further development of HS algorithm, such as easy trapping into local optimum and slow convergence velocity. This paper proposes an improvement HS algorithm by using segmented search strategy (SHS) to overcome these drawbacks. Comparative experiments reveal the proposed SHS algorithm has perfect performance, especially in high dimensions.

1. Introduction
Harmony search (HS) algorithm, proposed by Geem et al. in 2001, is a kind of metaheuristic algorithms based on population and mimics the musician’s improvisational process-improving the pitch of the instrument by finding the perfect harmony [1-2].

Compared to other intelligent optimization algorithms, HS requires minimal mathematical conditions. The search process is a completely random pattern without other restrictions. However, some shortages, like slow convergence velocity and easy trapping into local optimum, hinder the further development of HS [3-5]. Thus, an improved harmony search algorithm by using segmented search (naming SHS) is proposed to overcome the shortcoming above. To verify the performance of SHS algorithm, 3 test functions were selected to test the performance of SHS, and the results are compared with two others improved algorithms.

The outline of the rest of this paper as follow: Section 2 briefly introduces the basic HS algorithm. SHS algorithm steps are described and the main improvement strategies are analyzed in section 3. Section 4 shows the experimental results of algorithm performance comparison. Section 5 presents the conclusions.

2. Basic harmony search algorithm
HS consists of these parts: harmony memory consideration, pitch adjustment and random selection [1-5]. The basic framework of HS algorithm is illustrated as follows:

1. Initialization of parameters ($HMS, HMCR, PAR, bw, NI$);
2. Generate the harmony memory;

\[ X_{i,j} = L_j + (U_j - L_j) \times \text{rand} \quad (1) \]

3. Improvise a new harmony;

\[ X_{\text{new},j} = \begin{cases} 
X_{i,j}, & \text{with probability } \text{HMCR}, \\
X_i + (X_U - X_L) \times \text{rand}, & \text{with probability } (1-\text{HMCR}). 
\end{cases} \quad (2) \]

Pitch adjusting decision:

\[ X'_{\text{new}} = \begin{cases} 
X_{\text{new}}, & \text{with probability } (1-\text{PAR}), \\
X_{\text{new}} \pm \text{rand} \times bw, & \text{with probability } \text{PAR}. 
\end{cases} \quad (3) \]

4. Update the harmony memory;

5. Check the stopping criterion.

Where \( X_{i,j} \) is the j-th value in the i-th harmony vector. \( L_j \) and \( U_j \) are the lower and upper bounds. The parameters are the size of the harmony memory (HMS), the rate of choosing from harmony memory (HMCR), the rate of pitch adjustment (PAR), the bandwidth (bw) which used to adjust the solution vector and the maximum number of iterations (NI).

3. The proposed improved harmony search algorithm

The Global Dynamic Harmony Search (GDHS) algorithm was proposed by Mohammad Khalili et al. (2014), in this algorithm, there is no need to define any parameters in advance, and the domain is changed to dynamic mode to help faster convergence [6]. IHS can find the global optimal area and fine-tune in these areas to improve performance [5]. But they do not perform well on high-dimensional problems. Therefore, we propose some strategies to improve the performance of HS algorithm in high-dimensional problems.

The proposed improved HS algorithm includes three modifications. First, update the harmony memory by an improved harmony search algorithm (IHS) in early iteration [5]. Second, calculate the approximate position of the optimal solution vector and obtain the number of segments based on this value. Third, every few iterations, set the subinterval with the largest weight as the new search area. The steps in the procedure of SHS are shown:

- Step 1. Initialization of parameters (HMS, HMCR, PAR, bw, NI);
- Step 2. Generate the harmony memory;
- Step 3. Estimate the position of the optimal solution vector;
- Step 4. Determine the number of segments;
- Step 5. Improvise a new harmony;
- Step 6. Update the harmony memory;
- Step 7. Reduce search area based on subarea weights;
- Step 8. Check the stopping criterion.

The main improvement steps are steps 3, 4 and 7, which described in the next three subsections.

3.1. Estimate the position of the optimal solution vector

Some experimental results show how the position of the optimal solution vector in search area affects the number is odd or even. In the first few iterations, we use IHS to optimize the
problem and estimate the position, because the performance of IHS in finding the optimal area is good [5]. IHS dynamically adjusts the PAR and \( bw \) by using the following equation:

\[
PAR(gn) = PAR_{\text{min}} + (PAR_{\text{max}} - PAR_{\text{min}}) \times \frac{gn}{NI} (4)
\]

\[
bw(gn) = bw_{\text{max}} \times (c \times gn) (5)
\]

\[
c = \ln\frac{bw_{\text{min}}}{bw_{\text{max}}}/NI (6)
\]

Where \( PAR(gn) \) is the pitch adjusting rate for each generation, \( PAR_{\text{min}} \) is the minimum pitch adjusting rate, \( PAR_{\text{max}} \) is the maximum pitch adjusting rate, \( NI \) is the number of solution vector generations, \( gn \) is the generation number, \( bw_{\text{min}} \) is the minimum bandwidth and \( bw_{\text{max}} \) is the maximum bandwidth [5].

This step is to estimate the approximate location of the optimal value and determine the number of segments.

3.2. Determine the number of segments
The area segmentation strategy should avoid optimal values near the boundary between two adjacent subintervals. Some experiments show that search performance is related to the parity of the number of segments and the size has a limited impact on performance. Therefore, determine the number is 3 or 4 is a crucial step. The judgment criterion is as follows:

Step1. Divide the search area into 3 segments;
Step2. Determine in which subspace the optimal solution is and calculate the position of the optimal solution in the subspace;
Step3. Determine number of segments based on the location. If it is lower than the lower quartile or higher than the upper quartile, the number of segments is 4, otherwise it is 3.

3.3. Reduce search area based on subarea weights
Each iteration, solution vector is generated from each subinterval, the weight of the subinterval is increased if it generates the optimal solution vector. Every few iterations, the search area is reduced to the subinterval with the highest weight.

Once the area is found, it will be segmented and searched in parallel. This strategy can improve convergence speed and accuracy.

4. Experiments
In the experiment, we use GDHS and IHS to compare with SHS.

4.1. Test problems
In order to prove the superiority of the SHS algorithm, three classic test functions are used, as shown in table 1, where Opt is the global optimum, D is the dimension. The parameters settings are shown in table 2.
Table 1. Three classic test functions in experiments

| Function | Function expression | Search range | Opt | D  |
|----------|---------------------|--------------|-----|----|
| Sphere   | $f_1(x) = \sum_{i=1}^{D} x_i^2$ | [-100,100]   | 0   | 100|
| Rastrigin| $f_2(x) = \sum_{i=1}^{D} (x_i^2 - 10 \cos (2\pi x_i) + 10)$ | [-5.12,5.12] | 0   | 100|
| Griewank | $f_3(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \sum_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | [-600,600]   | 0   | 100|

Table 2. Setting parameters for various algorithms.

| Algorithm | HMCR | HMS | PAR | PAR min | PAR max | bw min | bw max | NI |
|-----------|------|-----|-----|---------|---------|--------|--------|----|
| IHS       | 0.9  | 5   | 0.35| 0.99    | 5e-5    | 0.05   | 8000   |
| GDHS      |      |     |     |         |         |        |        |    |
| SHS       | 0.9  | 5   | 0.35| 0.99    | 5e-5    | 0.05   | 8000   |

4.2. Comparison of SHS with IHS and GDHS

Each algorithm runs independently 50 times, the mean and standard deviations (Std.) of the optimal values are recorded in table 3, where the best mean is marked in bold and the best standard deviation is underlined.

Table 3. Results of IHS, GDHS and SHS over classic benchmark functions

| F  | IHS | GDHS | SHS |
|----|-----|------|-----|
|    | Mean | Std. | Mean | Std. | Mean | Std. |
| $f_1$ | 35911.53730 | 3827.29598 | 0.16271 | 0.04628 | 1.35241e-10 | 5.43753e-12 |
| $f_2$ | 464.96842 | 23.63408 | 154.21895 | 18.81714 | 9.90951e-04 | 2.01746e-05 |
| $f_3$ | 323.66248 | 33.37684 | 0.17919 | 0.04537 | 8.50249e-11 | 6.53210e-12 |

As shown in table 3, compared with the other two algorithms, whether it is the mean or the standard deviations, SHS can obtain the best value. Therefore, under the same number of iterations, the solution accuracy of SHS is better than the other two algorithms.
Figure 1 shows the convergence curves of IHS, GDHS and SHS on test functions. As shown in figure 1 (a), in the first 500 iterations, the convergence curves of the three algorithms are almost the same, the convergence curve of SHS is even worse than that of IHS, in Rastrigin function. After interval segmentation, the change of the SHS curve shows a cliff-like. As shown in figure 1 (b-c), although GDHS and SHS have similar accuracy in both Sphere and Griewank functions, SHS converges significantly faster than IHS and GDHS.

5. Conclusion
This paper examines a new measure of improvisation in HS based on segmented search. Three test functions are used to verify the performance of the SHS. The result show, SHS has better convergence accuracy and faster convergence speed.

Acknowledgments
The work was supported by the National Natural Science Foundation of Hunan Province (No.2019JJ40234), Research Foundation of Education Bureau of Hunan Province (Nos. 19A414 and 18B317), Special Project on Language Application of Hunan Provincial Language Committee (No. XYJ2019GB09).

References
[1] Geem Z W, Kim J H and Loganathan G V 2001 A new heuristic optimization algorithm: harmony search Simulation 76 pp 60–68.
[2] Alia O M and Mandava R 2011 The variants of the harmony search algorithm: an overview J. Artif Intell Rev 36 pp 49-68
[3] Lee K, Lee S and Geem Z W 2004 A new structural optimization method based on the harmony search algorithm Comput Struct 82 pp 781-798
[4] Zhi Q, Tang X, Li Y and Yeboah M O 2020 An improved differential-based harmony search algorithm with linear dynamic domain Knowl-Based Syst 187
[5] Mahdavi M, Fesanghary M and Damangir E 2006 An improved harmony search algorithm for solving optimization problems J. Appl Math Comput 188 pp 1567-79
[6] Khalili M, Kharrat R, Salahshoor K and Sefat M H 2014 Global Dynamic Harmony Search algorithm: GDHS J. Appl Math Comput 228 pp 195-219