Applied Mechanics and Materials
ISSN: 1662-7482, Vols. 687-691, pp 1367-1372
doi:10.4028/www.scientific.net/AMM.687-691.1367

Submitted: 2014-09-27
Accepted: 2014-09-28
Online: 2014-11-27

An Improved Harmony Search Algorithm with Application

Li Jianping [a], Lu Aiping [b], Wang Haochang [c], Li Xin [d] and Li Panchi [e]

School of Computer and Information Technology, Northeast Petroleum University, Daqing 163318, China

[a]leejp@126.com, [b]luaiping@126.com, [c]wanghc@126.com, [d]lixin_dq@163.com, [e]lipanchi@vip.sina.com

Keywords: intelligent computation; intelligent optimization; harmony search; algorithm design.

Abstract. In classical harmony search algorithm, only one harmony vector is obtained in each of iteration, which affects its search ability. We propose an improve harmony search algorithm in this paper. In our approach, the number of harmony vectors obtained in each of iteration is equivalent to the population size, and all newly generated harmony vectors are put into the harmony memory array. Then, all harmony vectors are sorted by descending order of the fitness, and the first half individuals are served as the next generation of populations. Experimental results show that our approach is obviously superior to the classical one under the same iteration steps and the same running time, which reveals that our approach can effectively generate the excellent individuals approximating the global optimal solution and enhance the optimization ability of classical harmony search algorithm.

Introduction

Many of the existing heuristic algorithms are simulations of natural phenomena. Such as the simulated annealing algorithm is a simulation of physical annealing [1], the tabu search algorithm is a simulation of human intelligence and memory [2], the evolutionary algorithm is a simulation of natural evolution [3], the particle swarm algorithm [4] and the ant colony algorithm [5] is a simulation of swarm intelligence. There is no exception that the harmony search algorithm known as the new heuristic algorithm. It is a simulation of the process in which musicians get the most beautiful harmony through repeated adjustment tones of different instruments. Musician's music is usually composed of the following three steps: (1) Playing a famous songs that is already famous and beautiful harmonies; (2) Playing a similar harmony with their own memories (Namely, fine-tuning harmony in their own harmony memories); (3) Improvising tone to form a new harmony. Geem is inspired by the above three kinds of music playing, proposed the harmony search algorithm in 2001 [6]. As the first practical application after it is proposed, the harmony search algorithm successfully solves the traveling salesman problem. At present the engineering application of harmony search algorithm is mainly focused on the following aspects: Transportation Design [7], Cluster analysis [8], Multi-objective optimization [9], Structural finite element model updating [10], PID control parameter design [11], engineering optimization [12-14], Vehicle Routing [15] etc. The only one harmony vector is obtained in each of iteration in classical harmony search algorithm. Despite higher efficiency, but optimization capabilities is unsatisfactory. Especially for complex high-dimensional nonlinear optimization, it is difficult to obtain global optimal solution. To solve this problem, this paper proposes an improved harmony search algorithm. An equivalent number of harmony vectors with population size are obtained in each of iteration, then preferentially into the next generation harmony memory. Typical function extreme optimization results show that the proposed algorithm is better than the original algorithm.

Harmony Search Algorithm

The harmony search algorithm can be divided into the following five parts: initialize algorithm parameters; initialize harmony memory; create new harmonies; update harmony memory; determine the termination condition.

Parameter Initialization. The parameters of harmony search algorithm include: harmony memory size (HMS), probability of getting value from harmony memory (HMCR), probability of tone fine-tuning (PAR), tone fine-tuning range (bw). In addition, the termination and constraint condition also need to be considered. A non-constrained optimization problem can be described as

$$\begin{cases} \min f(\boldsymbol{X}), \boldsymbol{X} = (x_1, x_2, \cdots, x_n) \\ x_i \in [x_{i\min}, x_{i\max},], i = 1, 2, \cdots, n \end{cases}, \tag{1}$$

where $f(\boldsymbol{X})$ denotes the objective function, $\boldsymbol{X}$ denote the decision variables.

Harmony Memory Initialization. The initialization of harmonic memory can be described as follows. All dimension of harmonic vector are initialized to a random number in a given interval, which can be written as follows.

$$x_i^k = x_{i\min} + (x_{i\max} - x_{i\min}) \times rand, \tag{2}$$

where $rand$ denotes a random number in (0, 1).

The form of the harmony memory is shown in Eq.(3).

$$\mathrm{HM} = \begin{bmatrix} \boldsymbol{X}^1, & f(\boldsymbol{X}^1) \\ \boldsymbol{X}^2, & f(\boldsymbol{X}^2) \\ \vdots & \vdots \\ \boldsymbol{X}^{\mathrm{HMS}}, & f(\boldsymbol{X}^{\mathrm{HMS}}) \end{bmatrix} = \begin{bmatrix} x_1^1, & \cdots, & x_n^1, & f(\boldsymbol{X}^1) \\ x_1^2, & \cdots, & x_n^2, & f(\boldsymbol{X}^2) \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{\mathrm{HMS}}, & \cdots, & x_n^{\mathrm{HMS}}, & f(\boldsymbol{X}^{\mathrm{HMS}}) \end{bmatrix}. \tag{3}$$

Generate a New Harmonic Vector. This is the most crucial part of the Harmony Search Algorithm. The new harmonic vector will be produced in the following three ways.

1) Learning from the harmony memory.

2) Implement tone fine-tuning based on learning.

3) Generate randomly a new tone.

For a new harmony vector $\boldsymbol{X}^{\mathrm{new}} = [x_1^{\mathrm{new}}, x_2^{\mathrm{new}}, \ldots, x_n^{\mathrm{new}}]$, If a component x_i^{new} is picked from the memory, it can be randomly selected any one value in the ith column of the memory. First of all, generate a random number rand_1 in the interval (0,1), and then perform the following Equation.

$$\begin{cases} x_i^{\mathrm{new}} = x_i \in \{x_i^1, x_i^2, \cdots, x_i^{\mathrm{HMS}}\}, & \mathrm{rand}_1 < \mathrm{HMCR} \\ x_i^{\mathrm{new}} = x_{i\min} + (x_{i\max} - x_{i\min})\mathrm{rand}, & \mathrm{rand}_1 \geq \mathrm{HMCR} \end{cases}. \tag{4}$$

The above equation indicates that any one tone of the new vector can get values from the memory with HMCR of probability, and from the domain of definition with 1-HMCR of probability.

When the ith tone is picked from memory, it probability of this tone PAR can be fine-tuned with PAR of probability by the following equation.

$$\begin{cases} x_i^{\mathrm{new}} = x_i^{\mathrm{new}} \pm \mathrm{rand}(0,1) \times \mathrm{bw}, & \mathrm{rand}_2 < \mathrm{PAR} \\ x_i^{\mathrm{new}} = x_i^{\mathrm{new}}, & \mathrm{rand}_2 \geq \mathrm{PAR} \end{cases}. \tag{5}$$

It can be seen from Eqs.(4,5) that the fine-tuning is based on removing the tone from memory, and then plus a random adjustment between $\pm \mathrm{bw}$.

Updated Harmony Memory. The new harmonic vector is evaluated by target function after it generated. If the new vector is superior to the worst one in harmony memory, the worst one is replaced with this new vector.

Termination Condition. Check whether the termination condition is satisfied or not. If it not is satisfied go back to 2.3, otherwise ends. The termination condition may take many forms, and the

precision threshold is a kind of commonly used indicators. For the minimum optimization problem, when the optimization result is less than this threshold, the algorithm terminates. Of course, the maximum iteration steps also can be considered. The basic flow of harmony search algorithm is shown in Fig.1.

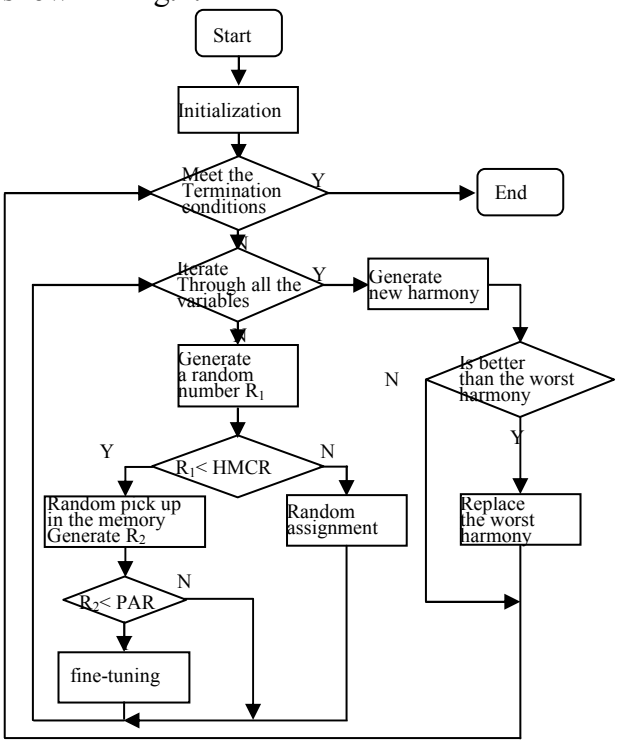

Figure 1. The flow chart of general harmonic search algorithm.

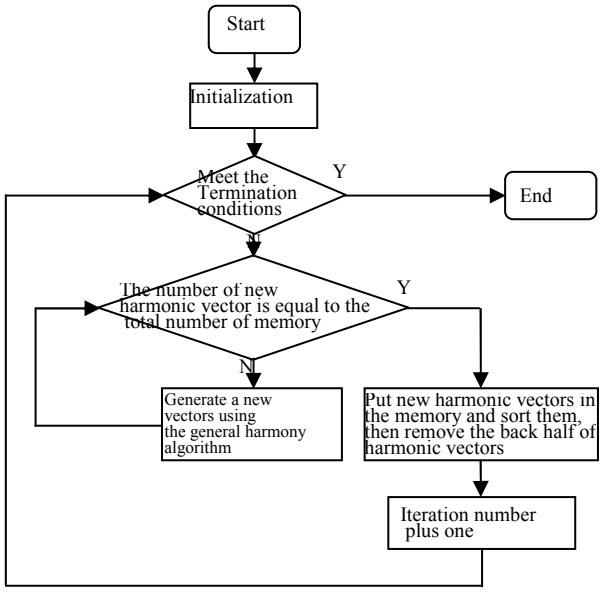

Figure 2. The flow chart of improved harmonic search algorithm

Improved Harmony Search Algorithm

It can be seen from Fig.1 that in step (C) and step (D) of the basic harmony search algorithm, each of iteration only generate a new harmonic vector, and replace the worst memory harmony with this vector. Although this approach is efficient, the search results are not satisfactory. To solve this problem, taking the optimization of the minimum for an example, we propose the improvement measures as follows. Let the harmony memory size equal M. At each of iteration, M new harmonic vectors are generated, and then put them into harmony memory. Sort all 2M harmonic vectors according to the objective function values from low to high. Finally pick the first M harmonic vectors with low objective function values into the next generation of harmony memory. The improved algorithm flow is shown in Fig.2.

Comparative Experiment

Test Function. To verify the superiority of the improved algorithm, in this section the following nine typical high-dimensional functions are applied to compare with the basic harmony search algorithm.

$$f_1(\boldsymbol{X}) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2 ,\ x_i \in [-100, 100], x_i^* = 0 ,\ f_1(\boldsymbol{X}^*) = 0 .$$

$$f_2(\boldsymbol{X}) = 418.982887 - \frac{1}{D}\sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|}) , x_i \in [-500, 500],\ x_i^* = 420.968746 ,\ f_2(\boldsymbol{X}^*) = 0 .$$

$$f_3(\boldsymbol{X}) = \sum_{i=1}^{D} u(x_i,10,100,4) + \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\},\ \ y_i = 1 + \frac{1}{4}(x_i + 1) ,$$

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, x_i>a\\ 0, \quad -a\le x_i\le a\\ k(-x_i-a)^m, x_i<-a\end{cases}, \ x_i\in[-100,100], \ x_i^*=-1, \ f_3(\boldsymbol{X}^*)=0.$$

$$f_4(\boldsymbol{X})=\frac{1}{10}\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_D-1)^2[1+\sin^2(2\pi x_D)]\}+\sum_{i=1}^{D}u(x_i,5,100,4),$$

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, x_i>a\\ 0, \quad -a\le x_i\le a\\ k(-x_i-a)^m, x_i<-a\end{cases}, \ x_i\in[-100,100], \ x_i^*=1, \ f_4(\boldsymbol{X}^*)=0.$$

$$f_5(\boldsymbol{X})=-29.63088385032440-\sum_{i=1}^{D}\sin(x_i)\sin^{20}(\frac{ix_i^2}{\pi}), \ x_i\in[0,\pi], f_5(\boldsymbol{X}^*)=0.$$

$$f_6(\boldsymbol{X})=\sum_{i=1}^{D}|x_i|^{(i+1)}, \ x_i\in[-100,100], x_i^*=0, \ f_6(\boldsymbol{X}^*)=0.$$

$$f_7(\boldsymbol{X})=\sum_{i=1}^{D-1}(0.5+\frac{\sin^2\sqrt{100x_i^2+x_{i+1}^2}-0.5}{1+0.001(x_i^2-2x_ix_{i+1}+x_{i+1}^2)^2}), \ x_i\in[-100,100], \ x_i^*=0, \ f_7(\boldsymbol{X}^*)=0.$$

$$f_8(\boldsymbol{X})=1-\exp(-0.5\sum_{i=1}^{D}x_i^2), x_i\in[-1,1], x_i^*=0, \ f_8(\boldsymbol{X}^*)=0.$$

$$f_9(\boldsymbol{X})=-[A\prod_{i=1}^{D}\sin(x_i-z)+\prod_{i=1}^{D}\sin(B(x_i-z))]+A+1, x_i\in[30,30+\pi], A=2.5, B=5, z=30,$$
$$x_i^*=\pi/2+z, \ f_9(\boldsymbol{X}^*)=0.$$

The above nine functions are minimum optimization problem, where D denote the dimension of optimize space, $\boldsymbol{X}^*$ denotes the exact global optimal solution，$f(\boldsymbol{X}^*)$ is the objective function value of $\boldsymbol{X}^*$. Some related definitions are below.

1) Accuracy threshold ε: If $|f(X)-f(X^*)|<\varepsilon$ before algorithm reach the Maximum Iterative Steps (MIS), it is convergent; otherwise it is not considered converge.

2) Error E: The error of an optimized solution X is defined as $E=|f(X)-f(X^*)|$。

3) Iteration Steps (IS): The number of iterations when algorithm converges, if it is not converged, the Iteration Steps are set to MIS.

4) Run time: The average time to conduct an iteration step.

5) Termination condition: When the maximum number of iterations is reached, the algorithm terminates.

Parameter Settings. To make a fair comparison, the improved algorithm employs the same parameters as the original algorithm, and these parameters are defined as follows. HMS = 30; HMCR = 0.9; PAR = 0.3; bw = 0.001. The parameters of target function are below: D=30; $\varepsilon=0.1$. After several simulations, we found that the running time is about 4-9 times long than the basic algorithm. To enhance the credibility of good performance of the improved algorithm, it is necessary to implement the contrast at the same running time. Therefore, in the improved algorithm, the maximum number of iterations MIS = 104; and in the basic algorithm, MIS = 104 and MIS = 105 respectively. To reflect the advantages of the improved algorithm and reduces the randomness of the algorithm, each function is independently optimized 50 times by the improved algorithm and the original one respectively. The convergence times, the mean and variance of error, the mean and variance of iteration steps, and the time mean, as the comparison indexes.

Comparison Results. Two kinds of algorithm adopt Matlab7.0 programming and simulate in a computer with 2.0GHz frequency and 1.0G memory. In 50 times optimization results, the

convergence times, the mean and variance of error, the mean and variance of iteration steps, and the time mean are shown in Table 1.

Table 1. The contrasts of convergence times, mean and variance of error, mean and variance of iterations, and time mean between improved algorithm and original one

Fun.	Alg.	Convergence times	Error mean	Error variance	Iterations mean	Iterations variance	Time mean
	Improved (10^4)	**48**	**0.0400**	0.0392	**6319**	3.2e+006	**2.0103**
f_1	Original (10^4)	0	398.76	9.9e+003	10000	0	0.5019
	Original (10^5)	0	3.9400	2.1392	100000	0	24.5425
	Improved (10^4)	**50**	**0.0348**	2.25e-004	**5754**	1.2e+006	**2.7153**
f_2	Original (10^4)	0	15.197	13.5280	10000	0	0.5500
	Original (10^5)	0	0.3758	0.0194	100000	0	25.2228
	Improved (10^4)	**50**	**0.0031**	1.25e-005	**3245**	8.4e+005	**7.6106**
f_3	Original (10^4)	0	1.0e+004	4.2e+008	10000	0	0.8738
	Original (10^5)	29	0.1034	0.0050	90174	1.48e+008	28.2250
	Improved (10^4)	**49**	**0.0526**	4.21e-004	**7886**	9.6e+005	**7.1753**
f_4	Original (10^4)	0	3.8e+005	1.1e+011	10000	0	0.8622
	Original (10^5)	0	0.8604	0.1152	100000	0	28.2003
	Improved (10^4)	**50**	**0.0446**	2.64e-004	**5067**	1.9e+006	**4.1006**
f_5	Original (10^4)	0	2.1591	0.1476	10000	0	0.6728
	Original (10^5)	4	0.1622	0.0026	98670	2.99e+007	26.5228
	Improved (10^4)	**47**	**0.0194**	0.0017	**6351**	2.7e+006	**3.8872**
f_6	Original (10^4)	0	7.7e+018	2.6e+039	10000	0	0.6769
	Original (10^5)	0	125.53	1.04e+005	100000	0	26.4403
	Improved (10^4)	**22**	**0.1564**	0.0148	**8426**	4.5e+006	**6.8584**
f_7	Original (10^4)	0	0.4994	1.0e-007	10000	0	0.7431
	Original (10^5)	0	0.3624	0.0107	100000	0	27.3881
	Improved (10^4)	**50**	**4.73e-007**	4.75e-015	**1024**	328.3673	**1.8972**
f_8	Original (10^4)	50	0.0195	3.3e-005	3017	2.5e+005	0.5291
	Original (10^5)	50	2.06e-006	1.31e-012	30652	3.23e+005	28.2378
	Improved (10^4)	**50**	**1.35e-005**	4.09e-012	**1224**	1.69e+004	**2.6444**
f_9	Original (10^4)	0	0.8716	0.0144	10000	0	0.5566
	Original (10^5)	50	4.53e-004	3.53e-007	36847	2.38e+007	36.0288

Experimental results show that on the indicators contrast, not only when the two algorithms use the same number of iterations, the improved algorithm is significantly superior to the original one for all nine standard test functions, and when the iterative steps of basic algorithm increased 9 times, the improved algorithm is also significantly better than the basic algorithm.

For the above experimental results, we present the following analysis. The core of the mechanism of harmony search algorithm is to generate new individuals with excellent performance by sharing the entire population information. These new individuals play the role of road signs to guide the optimization process gradually advancing toward the optimal solution. Obviously the number of new individual decides the degree of update of population, also determines the rate of approximate global optimal solution. However, in the basic algorithm, each generation construct only one new individual. Because the number of new individuals is too little, thereby the efficiency of optimization is severely reduced. On the other hand, due to the small difference between the two generations of adjacent populations, it is easy to fall into premature convergence such as in the function f7.

For the improved algorithm, the number of new harmonic vectors in each of iteration is equal to population size. This approach greatly increases the number of new individuals in the current population, and improves the degree of updating population between two adjacent iterations. Under the guidance of many new individual, the improved algorithm significantly enhances the optimization efficiency, but also avoids the premature convergence. It is also worth pointing out that the improved algorithm increases the amount of computation, which has been extended running time. The optimization ability of the improved algorithm has been enhanced at the expense of running time, which is consistent to the conclusion of no free lunch theorem.

Conclusions

This paper presents an improved harmony search algorithm, the difference between the proposed algorithm and the basic harmony search algorithm is that, in each of iteration, the number of new individuals is equal to the population size. The benchmark function extreme optimization results show that the proposed method can greatly enhance the optimization ability of the basic harmony search algorithm.

Acknowledgment

This paper was supported by Scientific Research Foundations of Heilongjiang Provincial Education Department (Grant No. 12511012).

References

[1] Yang Yanxia. “A hybrid differential evolution algorithm based on simulated annealing operation”. CAAI Transactions on Intelligent Systems, 2014, 9(1): 1-6.

[2] Ye Xiaolong, Lan Julong, Guo Tong. “Algorithm of Network Traffic Feature Selection Based on PAC and Tabu Search”. Computer Science, 2014, 41(1): 187-191.

[3] Dong Ning, Wang Yuping. “Multi-objective evolutionary algorithm based on preference for constrained optimization problems”. Journal of XIDIAN University, 2014, 41(1): 121-129.

[4] Li Xinpeng, Zhang Chaoyong, Gao Liang. “NC cutting parameter optimization based on cellular particle swarm optimization algorithm”. Computer Engineering and Applications, 2014, 50(2): 252-257.

[5] Lai Zhiming, Guo Gongde. “Ant Colony Optimization Based on Self-Adaption Threshold for Path Planning”. Computer Systems & Applications, 2014, 23(2): 113-118, 59.

[6] Geem Z M, Kim J H, Loganathan G V. “A new heuristic optimization algorithm: harmony search”. Simulation, 2001, 76(2): 60-68.

[7] Lai Zhizhu. “Harmony search algorithm for solving selection of multimodal transportation scheme with several time windows”. Journal of Computer Applications, 2013, 33(9): 2640-2642.

[8] Yi Yufeng, Gao Liqun, Guo Li. “The Application of Harmony Search Algorithm in Clustering Analysis”. Journal of Northeastern University(Natural Science), 2012, 33(1): 47-51.

[9] Hao bing，Ren Xianhua，Gao Yuelin. “Hvbrid harmonv search and estimation of distribution algorithm for multi-objective optimization problems”. Application Research of Cornputers, 2012, 29(5): 1659-1661.

[10]Du Yongfeng, Li Wangrun, Li Hui. “Application of harmony search algorithm in structural finite element model updating”. Journal of Lanzhou University of Technology, 2013, 39(5): 106-110.

[11]Zou Dexuan, Gao Liqun, Wu Peifeng. “A Global Harmony Search Algorithm and Its Application to PID Control”. Journal of Northeastern University (Natural Science), 2010, 31(11): 1534-1537.

[12]Fesangharya M, Mahdavib M, Jolandanc M M, et al. “Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems”. Computer Methods in Applied Mechanics and Engineering, 2008,197(33-40), 3080-3091.

[13]Majid J, Esmaile K. “Two improved harmony search algorithms for solving engineering optimization problems”. Communications in Nonlinear Science and Numerical Simulation, 2010, 15(1): 3316-3331.

[14]Mahdav I M, Fesangharym D E. “An improved harmony search algorithm for solving optimization problems”. Applied Mathematics and Computation, 2007, 188(2): 1567-1579.

[15]Geem Z W, Lee K S, Park Y. “Application of harmony search to vehicle routing”. American Journal of Applied Sciences, 2005, 2(12): 1552-1557.