2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A New Penalty Function Method for Constrained Optimization Using Harmony Search Algorithm

Biao Zhang, Jun-hua Duan, Hong-yan Sang, Jun-qing Li, Hui Yan

***Abstract*—This paper proposes a novel penalty function measure for constrained optimization using a new harmony search algorithm. In the proposed algorithm, a two-stage penalty is applied to the infeasible solutions. In the first stage, the algorithm can search for feasible solutions with better objective values efficiently. In the second stage, the algorithm can take full advantage of the information contained in infeasible individuals and avoid trapping in local optimum. In addition, for adapting to this method, a new harmony search algorithm is presented, which can keep a balance between exploration and exploitation in the evolution process. Numerical results of 13 benchmark problems show that the proposed algorithm performs more effectively than the ordinary methods for constrained optimization problems.**

I. INTRODUCTION

THE harmony search(HS) algorithm, developed by Geem et al. [1] in 2001, is a relatively new population-based meta-heuristic optimization algorithm. It is inspired by the music improvisation process that the musicians improvise their state of harmony to search for a perfect instruments pitch.

The harmony search algorithm has its characteristics of few mathematical requirements, easy implementation and fast convergence speed, so that it has attracted many researchers from various fields especially these working on solving optimization problems. For example, Mahdavi et al. [2] presented an improved HS algorithm (IHS) by introducing a strategy to dynamically adjust the control parameters. Later, Omran and Mahdavi [3] proposed a global best HS algorithm (GHS), taking advantage of the global best solution.

Although the HS algorithm has been successfully applied to optimization problems [4]-[13], there are few about the constrained optimization problems. As we know, most real-world optimization are constrained optimization problems involving constraints of a certain king. Taking a certain production line for an example, the goal is to maximize the profits with the limits of material consumption, labor cost and other factors.

The constrained optimization problems are generally difficult to deal with. Due to the limits of constraints, it can divide the whole search space into some portions . In addition, the interference among constraints, the interrelationship between the constraints and the objective function could be considered. Consider maximizing a function $f(X) = x_1 + x_2$ where two variables are defined by $0 \le x_1, x_2 \le 1$, an optimum value of $f(X) = 2$ can be got when $x_1 = 1$ and $x_2 = 1$. But if there is an equality constraints as $g(X) = x_1 - x_2 = 0.5$, the feasible spaces in only 0.5% of the total search space. And the optimum value in this case would be $f(X) = 1.5$ where $x_1 = 1$ and $x_2 = 0.5$.

In general constrained optimization problem can be formulated as:

$$\text{Optimize } f(X) = f(x_1, x_2, \cdots, x_n), \tag{1}$$

subject to inequality constraints

$$g_i(x) \le 0 \qquad i = 1, \cdots, k,$$

and equality constraints

$$h_i(x) = 0 \qquad i = k+1, \cdots m.$$

The value of inequality constraints that equals "0" at the global optimum solution are called active constraints. And the solutions can be divided into feasible individuals and infeasible individuals. Individuals that satisfy all of the constraints are called feasible individuals while individuals that do not satisfy at least one of the constraints are called infeasible individuals. Without loss of generality, minimization problems are considered in this paper unless specified otherwise.

Generally, due to the presence of constraints, constrained optimization problems are difficult to solve. In this paper, a two-stage penalty strategy is proposed to solve constrained optimization problems using HS algorithm. In the first stage, a higher penalty is added to infeasible individuals, so the HS algorithm could search for feasible individuals efficiently. And in the second stage, a "distance" value [14] is imported. It can encourage the infeasible individuals with lower objective function value and small constraint violation. In addition, for adapting to this method, a new harmony search algorithm is presented, which can keep a balance between exploration and exploitation in the evolutionary process.

The rest of the paper is organized as follows. In Section 2, the classical HS algorithm is introduced. In Section 3, a

Biao Zhang is with the College of Computer Science, Liaocheng University, CO 252059 CHINA (phone: 13863565273; e-mail: zhangbiao1218@ gmail.com).

Jun-hua Duan is with the College of Computer Science, Liaocheng University, CO 252059 CHINA(email: duanjunhua@lcu-cs.com).

Hong-yan Sang is with the College of Mathematic Science, Liaocheng University, CO 252059 CHINA(email: sanghongyanlcu@163.com).

Jun-qing Li is with the College of Computer Science, Liaocheng University, CO 252059 CHINA(email: lijunqing.cn@gmail.com).

This research is partially supported by National Science Foundation of China 61174187, Program for New Century Excellent Talents in University (NCET-13-0106), Specialized Research Fund for the Doctoral Program of Higher Education (20130042110035), Science Foundation of Liaoning Province in China (2013020016), Basic scientific research foundation of Northeast University under Grant N110208001, Starting foundation of Northeast University under Grant 29321006, IAPI Fundamental Research Funds (2013ZCX02).

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

classical way dealing with constrained problems named death penalty method is described in detail. In Section 4, the proposed two-stage penalty strategy is proposed and a novel harmony search algorithm is presented for adapting the new strategy . Experimental design and comparisons are presented in Section 5. Finally, Section 6 gives the concluding remarks.

II. The Classical HS Algorithm

In the HS algorithm, each solution is called a "harmony". And it is represented by an n-dimension valued vector. The harmony memory (HM) is initialized by harmony vectors generated randomly. Then a new candidate harmony is generated by using a memory consideration rule, a pitch adjustment rule, and a random re-initialization scheme. Finally, the worst harmony vector is replaced by the new candidate vector if it is better than the worst harmony vector in the HM. The above process is repeated until a certain termination criterion will be met. In summary, the basic HS algorithm consists of three basic phases, initialization, improvisation of a harmony vector, and updating the HM. These process are described below.

A. Initialization

The initialization of the basic HS algorithm is presented in two steps. The control parameters including the harmony search size(HMS), harmony memory consideration rate ($HMCR$), pitch adjustment rate(PAR), distance bandwidth (bw), and termination criterion (NI) are set in the first step. The second step is to fill the HM with HMS number of harmony vectors.Let $X_i = (x_i(1), x_i(2), \ldots, x_i(n))$ represents the i^{th} harmony vector, which is randomly generated as follows:

$$\begin{aligned} & x_i(j) = LB(j) + (UB(j) - LB(j)) \times rand() \\ & s.t.\ \ j = 1,2,\ldots,n \ and \ i = 1,2,\ldots HMS, \end{aligned} \tag{2}$$

where $rand()$ is a uniform random function returning a number between 0 and 1, and $LB(j)$ and $UB(j)$ are the lower and upper bounds, for the design variable $x(j)$ respectively. Then, the HM is filled with the HMS number of randomly generated harmony vectors.

B. Improvise A New Harmony

A new harmony vector X_{new} is generated by applying three rules: a memory consideration, a pitch adjustment and a random selection.

In the memory consideration, each decision variable $x_{new}(j)\ (j \in [1,n])$ is chosen randomly from any harmony vector $x_i(j)\ (i \in [1, HMS])$ with the probability of HMCR so that the historical information obtained from the HM can be well used. With the probability of $1 - HMCR$, $x_{new}(j)$ is generated randomly. Furthermore, every individual obtained by the memory consideration is adjusted by the pitch adjustment rule with the proportion of PAR . The memory consideration, pitch adjustment and random selection are given below, respectively.

$$x_{new}(j) = x_a(j) \ \ where \ a \in (1,2,\ldots,HMS) \tag{3}$$

$$x_{new}(j) = x_{new}(j) \pm rand() \times bw \tag{4}$$

$$x_{new}(j) = LB_j + rand() \times (UB_j - LB_j) \tag{5}$$

C. Update Harmony Memory

The HM will be updated after X_{new} is generated. X_{new} will replace the worst harmony vector X_w and become a new member of the HM, if X_{new} is better than X_w .

D. Computational Procedure

The computational procedure of the basic HS algorithm can summarized as follows:

Step 1: Set the parameters HMS, HMCR, PAR, bw and NI.

Step 2: Initialize the HM and calculate the objective function value of each harmony vector.

Step 3: Improvise a new harmony X_{new} as follows:

```
For(j=1 to n) do
  If( r1 < HMCR ) then
    x_new(j) = x_a(j)  where a ∈ (1,2,…,HMS)
    If( r2 < PAR ) then
      x_new(j) = x_new(j) ± r3 × bw
    Endif
  Else
    x_new(j) = LB_j + r3 × (UB_j − LB_j)
                    Where r1, r2, r3 ∈ (0,1)
  Endif
Endfor
```

Step 4: Update HM as $X_w = X_{new}$ if $f(X_{new}) < f(X_w)$.

Step 5: If NI is completed, return the best harmony vector X_B in the HM; otherwise go back to step 3.

III. Death Penalty Method

How to deal with the infeasible individuals throughout the search process is the major issue for constrained optimization. One way for basic HS algorithm to deal with the infeasible individuals is to completely disregard them and go on with feasible individuals only. And the step 2 of the basic HS algorithm may be modified as follows:

Step 2: initialize the HM strictly and all members of the HM must satisfy all constraints and calculate the objective function value.

The flow chart of the modified step 4 is shown in Fig.1. But this has a drawback because HS is a probabilistic search method and some of the information contained in the infeasible individuals could be unutilized and this may lead to the algorithm being stuck in a local optima. In addition, in some highly constrained problems, initializing the HM with all feasible individuals might be difficult.

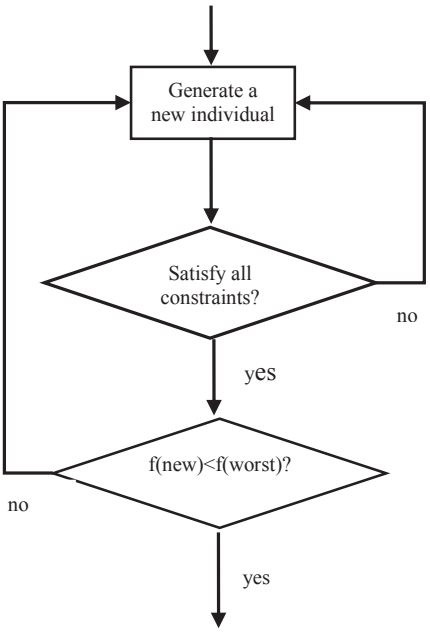

Fig. 1. The modified step 4.

IV. PROPOSED ALGORITHM

Static penalty function method is one way for constrained problems to exploit the information contained in infeasible individuals. This method is simple and easy to be programmed. In this method, some penalty value is added to the objective value of each infeasible individuals, so that it will be penalized for violating the constraints. Generally, penalty functions have the following form:

$$F(x) = f(x) + R \times c_j(x) \tag{6}$$

$$c_j(x) = \sum_{j=1}^{k} \max(0, g_j(x)) + \sum_{j=k+1}^{m} |h_j(x) - \delta| \tag{7}$$

where $F(x)$ is the updated fitness value, $f(x)$ is the original fitness value, R is the penalty coefficient and $c_j(x)$ is the j^{th} constraint violation (δ is the tolerance value).

Here, R is a large value, so the algorithm will encourage the feasible individuals, so HS algorithm using the static penalty function method can search for feasible individuals with better objective function value efficiently. The major shortcoming of this method is to determine the penalty coefficients. Usually, a prior knowledge of the problem is needed or repeated experiments should be done to determine the proper coefficients. To reduce the influence of the penalty coefficient and exploit more information contained in the infeasible individuals in later stage, this paper presents a two-stage penalty function method described in detail as follows.

To reduce the influence of the penalty coefficient in the static penalty function method and exploit more information contained in the infeasible individuals. This paper presents a two-stage penalty function.

A. The First Stage

The static penalty method is applied in the first stage. Because this measure has a large penalty value added to infeasible individuals, so that the algorithm could search for feasible solutions with better objective values efficiently and narrow the search space. The fitness value in this stage can formulated as (1) when $g \le \frac{2}{5} NI$ (g is the current generation criterion).

B. The Second Stage

In the second stage, the fitness formulation modified here is based on the method that was proposed by Wright and Farmani [15]. Constraint violation of each infeasible individual is then calculated as the sum of the normalized violation of each constraint divided by the total number of constraints.

$$v(x) = \frac{1}{m} \sum_{j=1}^{m} \frac{c_j(x)}{c\max_j} \tag{8}$$

where $v(x)$ is constraint violation function, m is total number of constraints. The infeasibility measure has the properties that it increases in value with both the number of active constraints and the maximum of each constraint violation. This measure has a benefit for solving highly constrained problems that have solutions on one or more of the constraint bounds. On the other hand, solutions that is farther from the constraint bounds are more likely to be penalized.

To exploit more information contained in the infeasible individuals, the "distance" value [14] is imported here. It can be formulated as follows:

$$d(x) = \sqrt{f'(x)^2 + v(x)^2} \tag{9}$$

$$f'(x) = \frac{f(x) - f_{\min}}{f_{\max} - f_{\min}} \tag{10}$$

where $f(x)$ is the objective function value , and $f_{\max}$ and $f_{\min}$ are the largest and smallest objective function values of the current population.

Fig.2 shows that the two different types of infeasible individuals may encounter during the search process. All of the two individuals shown in the figure are important but at different stages during the search process. Individual "A" has very low constraint violation but high objective value, while individual "B" has very low objective value but high constraint violation. Obviously, "A" can help in finding more feasible individuals, while "B" will help in finding the global optimum value. Through the first stage of search process, there will be feasible individuals(at least infeasible individuals with low constraint violation). So in the second stage, the algorithm will encourage "B" more likely and in the

first stage, individual "A" has a high selection probability.

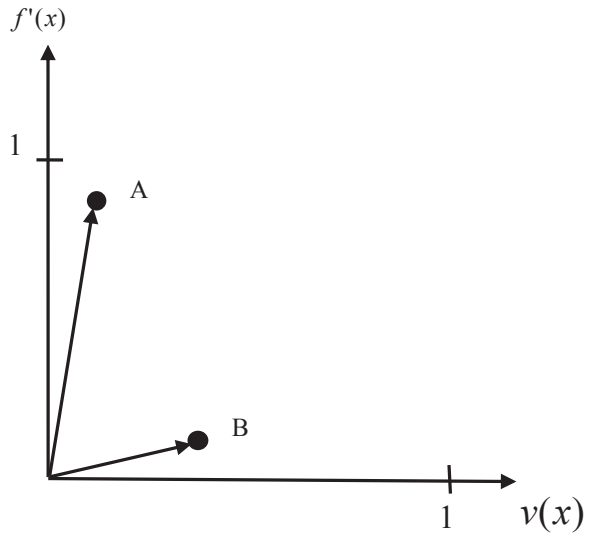

Fig. 2. Distance value for "A" and "B".

On the other hand, if two feasible individuals encounter in the second stage, the distance is equal to $f'(x)$. Then the individual with small objective function value will have small distance value, so the individuals with small distance value will be selected. And if we compare the distance value of a feasible individual with an infeasible individual, then either one can have smaller value. But if the two individuals have the same objective value, the feasible individual will have smaller distance value.

C. The Final Fitness Formulation

To sum up, the final fitness formulation can be given as (11).

$$F(x)=\begin{cases} f(x)+R\times c_j(x) & when\ g\le \dfrac{2}{5}NI \\ d(x) & when\ g>\dfrac{2}{5}NI \end{cases} \tag{11}$$

This fitness formulation is flexible, here are some properties of this fitness formulation:

1) In the early stage of the search process, there may be not feasible individuals in the HM, then individuals will be compared based on their objective function value adding a large penalty value, so the algorithm could find feasible individuals with better objective value, efficiently.

2) Through the first stage, there may be feasible individuals or infeasible individuals with low constraint violation at least. So, in the second stage, the individuals will be compared based on the distance value and informations contained in infeasible individuals could be exploited. This will help the algorithm search for global optimum.

D. The New HS Algorithm

For adapting to the properties of the two-stage penalty measure, a new HS algorithm with dynamic control parameters is presented in this section. Here are the modifications when the new algorithm is compared with the basic HS algorithm.

1) A novel memory consideration rule is presented here. In the second stage of the search process, when a new harmony with small distance value and low constraint violation is appeared in the HM, it should have a higher selection probability to find feasible individuals. So, a tournament rule [17] is to used to select a harmony with better fitness. The tournament rule here can be divided into two steps. Firstly, two harmonies are chosen randomly from the HM. Then, among them the best harmony x_{ts} is selected. The memory consideration is accordingly updated as follows:

$$x_{new}(j)=x_{ts}(j),\ \ j\in[1,n] \tag{12}$$

where x_{ts} is the j^{th} element of the harmony X_{ts} selected by the tournament rule. In addition, [8] also shows that improvising a harmony from a group of harmonies with higher qualities can effectively help the algorithm speed up its convergence.

2) The control parameters vary dynamically in the new algorithm. Since the control parameters have a profound effect on the performance of the HS algorithm, fine-tuning these parameters can help the method enhance the convergence rate and the precision of final solutions [10]. The details are presented below.

The parameter bw is a distance bandwidth and bw is closely related to the problem being solved as well. So the bw is presented as follows:

$$bw=0.01\times\frac{UB_j-LB_j}{50} \tag{13}$$

Another two control parameters are $HMCR$ and PAR , which determine the probability of fine-tuning the selected pitch or retaining its original value. A small $HMCR$ value and large PAR value are able to help the algorithm search for global optimums in the second stage. So, in the new algorithm, $HMCR$ and PAR is changed dynamically with the generation number as follows:

$$PAR(t)=PARMIN+\frac{PARMAX-PARMIN}{NI}\times g \tag{14}$$

$$HMCR(t)=HMAX-\frac{(HMAX-HMIN)\times g}{NI} \tag{15}$$

where $HMCR(t)$ and $PAR(t)$ are the value in generation t, $PARMAX$ and $HMAX$ are the maximum values, while $PARMIN$ and $HMIN$ are the minimum values.

V. Experimental Results

The performance of the proposed algorithm is investigated

TABLE I
SUMMARY OF MAIN CHARACTERISTICS OF THE BENCHMARK PROBLEMS

Test function	Max/Min	n	Type of function	Feasibility ratio	LI	NE	NI	a
g01	Min	13	Quadratic	0.011%	9	0	0	6
g02	Max	20	Nonlinear	99.990%	1	0	1	1
g03	Max	10	Nonlinear	0.002%	0	1	0	1
g04	Min	5	Quadratic	52.123%	0	0	6	2
g05	Min	4	Nonlinear	0.000%	2	3	0	3
g06	Min	2	Nonlinear	0.006%	0	0	2	2
g07	Min	10	Quadratic	0.000%	3	0	5	6
g08	Max	2	Nonlinear	0.856%	0	0	2	0
g09	Min	7	Nonlinear	0.521%	0	0	4	2
g10	Min	8	Linear	0.001%	3	0	3	3
g11	Min	2	Quadratic	0.000%	0	1	0	1
g12	Max	3	Quadratic	4.779%	0	0	9	0
g13	Min	5	Nonlinear	0.000%	0	3	0	3

TABLE II
RESULTS FOUND USING THE PROPOSED ALGORITHM

Test function	Optimum value	Best	Worst	Mean	Standard Deviation	Infeasible runs
g01	-15.000	**-14.999**	-14.893	-14.959	0.022914	0
g02	0.803619	0.725457	0.654311	0.700954	0.039653	0
g03	1.000	**1.000**	0.951	0.988	0.013714	0
g04	-30665.539	-30665.377	-30405.339	-30581.909	24.256724	0
g05	5126.498	5125,324	5112.324	5115.246	1.252652	0
g06	-6961.814	-6961.660	-6960.897	-6961.268	0.244345	0
g07	24.306	24.552	31.231	27.612	1.654452	0
g08	0.095825	**0.095825**	0.076144	0.080709	0.013589	0
g09	680.630	680.656	680.779	680.742	0.072523	0
g10	7049.331	7082.562	7110.263	7010.156	2.085432	0
g11	0.750	**0.749**	0.749	0.749	0.000003	0
g12	1.00000	**0.990860**	0.891323	0.95251	0.088769	0
g13	0.053950	0.057111	0.070276	0.059453	0.072566	0

TABLE III
RANK SUM TEST RESULTS

Test Function	g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11	g12	g13
P1	1	1	1	1	1	1	1	1	1	1	0	1	0
P2	1	1	1	1	1	1	1	1	1	1	0	1	1

TABLE IV
COMPARISON OF MEAN RESULTS

Test Function	Two-stage penalty method	Static penalty method	Death penalty method
g01	-14.959	-13.003	-9.253
g02	0.700954	0.699820	0.620031
g03	0.988	0.897	0.786
g04	-30581.909	-30482.347	-29992.920
g05	5115.246	4871.352	Invalid
g06	-6961.268	-6930.104	-6882.324
g07	27.612	29.527	30.286
g08	0.080709	0.075825	0.073825
g09	680.742	680.222	680.187
g10	7010.156	7002.453	7005.235
g11	0.749	0.749	0.749
g12	0.95251	0.850441	0.951548
g13	0.059453	0.098837	Invalid

TABLE V
COMPARISON OF BEST RESULTS

Test Function	Two-stage penalty method	Static penalty method	Death penalty method
g01	-14.999	-14.998	-10.746
g02	0.725457	0.700109	0.62618 6
g03	1.000	0.997	0.670540
g04	-30665.656	-30665.347	-30618.920
g05	5125.324	4971.758	Invalid
g06	-6961.660	-6961.304	-6960.727
g07	24.552	26.557	30.286
g08	0.095825	0.095825	0.095825
g09	680.656	680.650	680.789
g10	7082.562	7100.657	7104.895
g11	0.749	0.749	0.749
g12	0.990860	0.950451	0.951548
g13	0.057111	0.128437	Invalid

in this section by comparing with the common algorithms described in section 3. These three algorithm was tested on 13 test functions from [18]. We can observe that the test functions involve various types of problems. Some are maximization problems while others are minimization problems. On the other hand, the functions also vary from linear, nonlinear, quadratic, cubic, to polynomial. The number and the type of constraints(LI- linear inequality, NE-nonlinear equality, and NI- nonlinear inequality) are also different. The feasibility ratio is an estimate of the ratio of the feasible space to that of the entire of the ratio of the feasible space to that of the entire search space. a and n represent the number of active constraints and decision variable involved, respectively.

The basic HS algorithm is applied to the common measures. The parameter settings for these two algorithms are set as those in [1]: $HMCR = 0.9$, $PAR = 0.3$ and $bw = 0.01$; For the proposed algorithm, the parameters are set as these: $HMAX = 0.99$, $HMIN = 0.85$, $PARMAX = 0.99$, $PARMIN = 0.35$. The $HMS = 5$ is same for all the algorithms. Thirty independent replications are carried out for each function and the NI is set to 50000. The best, the worst, the mean results of each test function are reported in Table II. The numbers of infeasible runs from the 30 trials for each test function are also recorded in the same table. In addition, the best values that are identical to the already know optimum values are highlighted.

From Table II, we notice that the proposed algorithm could find feasible solutions for all the 30 runs of all the 13 test functions. And it was able to find very good results. The proposed algorithm found the known optimum in 5 of the 13 functions. In g01, g03, g08, g11 and g12, the values are exactly equal to the already known optimum values. In the rest of the test functions, the optimum values could not be found but the best results are very close to. In addition, all the "mean" and "standard deviation" values show the algorithm has good stability.

Table V compares the best results of the proposed algorithm with other two algorithms. Comparing the static penalty method and the death penalty method, the proposed algorithm generated better results for most of the test functions except g08 and g11 in which the values are equal. And the death penalty method for g05 and g13 is invalid. When the proposed algorithm is compared to the static penalty function, the proposed algorithm performed outstandingly for most of the test functions except g01, g08 and g11 for which the value of the best results are equal. And for g09, the proposed algorithm generated worse results but exactly equal to the result generated by the static penalty method. In general, we can say that the proposed algorithm can perform better than the two common algorithms for constrained optimization problems.

From Tables I, II, IV and V, it can be observed that the new algorithm generates better results than the other two algorithms directly. On the other hand, to investigate whether the results perform better than the compared algorithms in a statistically significant way, the Wilconxion's rank tests on the mean values showed in Table IV is conducted for the two-stage penalty method against static penalty method and death penalty method as $p1$ and $p2$, respectively. The results of the rank sum tests are expressed as the h value. The h value equals 1 or 0 when the new algorithm performs significantly better or not than the compared algorithms. When the h value comes to -1, it illustrates that the results generated by the new algorithm is worse significantly than the other two algorithms. From Table III, it can be observed that the proposed algorithm yields 11 and 12 significantly better results when comparing with the other two algorithms, respectively.

VI. Conclusion

In this paper, we have developed a two-stage penalty method for constrained optimization problem. In the first stage, the algorithm can find feasible individuals or at least infeasible individuals with low constraints. And in the second stage, the main objective of the algorithm is to exploit the information hidden in infeasible individuals. In addition, for adapting to the properties of the measure, a new HS algorithm is also presented in this paper. The performance of our algorithm was tested on 13 benchmark functions. The algorithm can find feasible solutions in every run for problems and can find competitive results with the two common algorithms.

Acknowledgment

This research is partially supported by National Science Foundation of China 61174187, Program for New Century Excellent Talents in University (NCET-13-0106), Specialized Research Fund for the Doctoral Program of Higher Education (20130042110035), Science Foundation of Liaoning Province in China (2013020016), Basic scientific research foundation of Northeast University under Grant N110208001, Starting foundation of Northeast University under Grant 29321006, IAPI Fundamental Research Funds (2013ZCX02).

References

[1] Z . W. Geem, J. H. Kim and G. V. Loganathan, "A new heuristic optimization algorithm:harmony search," Simulation, vol.76, pp. 60-68, 2001.

[2] M. Mahdavi, M. Fesanghary and E. Damangir, "An improved harmony search algorithm for solving optimization problems, " Appl. Math. Comput, vol.188, pp.2687-2699, 2010.

[3] M. G. H. Orman and M. Mahdavi, "Global-best harmony search," *Appl. Math. Comput*, vol.198, pp.643-656, 2008.

[4] B. Alatas, "Chaotic harmony search algorithms," Appl. Math. Comput, vol.216, no.9, pp.2687-2699, 2010.

[5] D. X. Zou, L. Q. Gao, J. H. Wu and S. Li, "Novel global harmony search algorithm fro unconstrained problems," *Neurocomputing, v*ol.73, pp.3308-3318, 2010.

[6] Y. M. Cheng, L. Li, T. Lansivaara, S. C. Chi and Y. J. Sun, "An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis," *Eng. Optim, v*ol.40, pp.95-115, 2008.

[7] Z. W. Geem, "Optimal cost design of water distribution networks using harmony search," *Eng. Optim,* vol.38, pp.259-280, 2006.

[8] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Appl. Math. Comput*, vol.199, no.1, pp.223-230, 2008.

[9] H. Ceylan and S. Haldenbilen, "Transport energy modeling with meta-heuristic harmony search algorithm, " *An Application to Turkey, Energy Policy,* vol.36, no.7, pp.2527-2535, 2008.

[10] Z. W. Geem and K. S. Lee, "Application of harmony search to vehicle routing, " *Am. J. Appl. Sci*, vol.2, pp.1552-1557, 2005.

[11] S. Das, A. Mukhopadhyay and A. Roy, "Exploratory power of the haromony search algorithm: analysis and improvements for global numerical optimization," *IEEE T, Syst. Man. Cybernet. B Cybernet*, vol.99, pp.1-18, 2010.

[12] M. Fesanghary, M. Mahdavi and Y. Alizadeh, "Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems, " *Comput.Methods Appl. Mech. Eng,* vol.197, pp.3080-3091, 2008.

[13] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Appl. Math. Comput,* vol.199, no.1, pp.223-230, 2008.

[14] Biruk Tessema and Gary G. Yen, "A self Adaptive Penalty Function Based Algorithm for Constrained Optimization,." *IEEE Congress On Evolutionary Computation*, pp.246-253, 2006.

[15] J. A. Wright and R. Farmani, "Genetic algorithm: A fitness formulation for constrained minimization," in *Proc. Genetic and Evolutionary Computation Conf*, pp. 725-732, 2001.

[16] Raziyeh Farmani and Jonathan A. Wright, "Self-Adaptive Fitness Formulation for Constrained Optimization," *IEEE Transactions On Evolutionary Computation*, vol.7, no.5, pp.445-455,2008.

[17] Jing Chen, Quan-ke Pan, Jun-qing Li, "harmony search algorithm with dynamic control parameters," *Applied Mathematics and Computation,*vol.219*,* pp.*5*92-604, 2012.

[18] T. P. Runarsson and X. Yao, "Search biases in constrained evolutioanry optimization," IEEE Transaction on Systems, and Cybernetics, Part C, vol.35, no.2, pp.233-244, 2005.