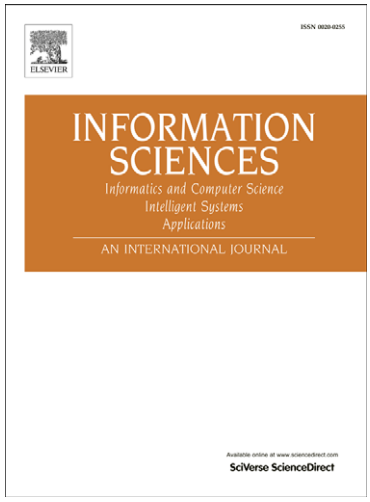


An Improved Adaptive Binary Harmony Search Algorithm

Ling Wang, Ruixin Yang, Yin Xu, Qun Niu, Panos M. Pardalos, Minrui Fei

PII: S0020-0255(13)00035-2
DOI: http://dx.doi.org/10.1016/j.ins.2012.12.043
Reference: INS 9891

To appear in: *Information Sciences*

Received Date: 3 September 2011
Revised Date: 9 December 2012
Accepted Date: 31 December 2012

Please cite this article as: L. Wang, R. Yang, Y. Xu, Q. Niu, P.M. Pardalos, M. Fei, An Improved Adaptive Binary Harmony Search Algorithm, *Information Sciences* (2013), doi: http://dx.doi.org/10.1016/j.ins.2012.12.043

An Improved Adaptive Binary Harmony Search Algorithm

Ling Wang[1,2], Ruixin Yang[1], Yin Xu[1], Qun Niu[1], Panos M. Pardalos[2], Minrui Fei[1]

1. Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronics and Automation, Shanghai University, Shanghai, 200072, China

2. Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida, 32611, USA

wangling@shu.edu.cn

Abstract. Harmony Search (HS), inspired by the music improvisation process, is a new meta-heuristic optimization method and has been successfully used to tackle the optimization problems in discrete or continuous space. Although the standard HS algorithm is able to solve binary-coded optimization problems, the pitch adjustment operator of HS is degenerated in the binary space, which spoils the performance of the algorithm. Based on the analysis of the drawback of the standard HS, an improved adaptive binary Harmony Search (ABHS) algorithm is proposed in this paper to solve the binary-coded problems more effectively. Various adaptive mechanisms are examined and investigated, and a scalable adaptive strategy is developed for ABHS to enhance its search ability and robustness. The experimental results on the benchmark functions and 0-1 knapsack problems demonstrate that the proposed ABHS is efficient and effective, which outperforms the binary Harmony Search, the novel global Harmony Search algorithm and the discrete binary Particle Swarm Optimization in terms of the search accuracy and convergence speed.

Keywords: Harmony Search, binary Harmony Search, meta-heuristic, knapsack problem

1 Introduction

Harmony Search (HS) is a recent meta-heuristic algorithm firstly developed by Geem et al. [13] in 2001. It imitates the musician seeking to find pleasing harmony determined by an aesthetic standard as the optimization method seeks to find the global optimal solution determined by an objective function. Unlike the traditional optimization algorithms based on the Gradient and Newton's Methods, HS uses a stochastic search instead of a gradient-based search, and therefore the derivative information is unnecessary [20].

Due to the characteristics such as easy implementation and quick convergence, HS has drawn more and more attention and dozens of variants have been developed to enhance the optimization ability. Pan et al. [33] proposed an improved Harmony Search algorithm with an ensemble of parameter sets which can self-adaptively choose the best control parameters during the evolution process. Mahdavi et al. [27] developed an adaptive pitch adjustment strategy to improve HS. Hasancebi et al. [18] proposed a new adaptive mechanism for HS in which the harmony memory consideration rate and the pitch adjustment rate change dynamically during the improvisation process. Pan et al. [35] presented a self-adaptive global best harmony search algorithm and proved that it was more effective in finding the better solutions than several other HS algorithms. Li and Li [21] addressed a hybrid HS algorithm combining HS with Particle Swarm Optimization (PSO) to solve high-dimensional optimization problems and achieved better optimization results. Omran and Mahdavi [31] introduced a new global-best HS inspired by the concept of PSO algorithms and validated its efficiency in numerical and integer programming problems. Li et al. [24] also proposed a hybrid PSO-HS algorithm where HS was used to deal with the variable constraints. Li and Wang [23] presented two hybrid algorithms by merging HS with the Differential Evolution algorithm and tested their performance with a set of benchmark functions. The results display that the hybrid algorithms surpass DE. Seok et al. [41] developed a hybrid Simplex Algorithm-Harmony Search in which the Simplex Algorithm was adopted to improve the accuracy and convergence speed. Zhao et al. [50] introduced the HS algorithm into the dynamic multi-swarm particle swarm optimizer (DMS-PSO) and the newly developed approach improved the performance compared with DMS-PSO and HS. Recently, hybridization has been the hotspot in HS research, and more and more hybrid algorithms have been reported, such as HS combined with a genetic algorithm (GA) [22] and a Clonal Selection algorithm [48].

Now HS has been successfully applied to a wide range of discrete and continuous optimization problems in the scientific and engineering fields, such as slope stability analysis [4, 25], structural engineering [38, 39], groundwater management model optimization and identification [1, 2], multiple dam system scheduling [12], energy dispatch problems [5, 30], clustering [8, 26], visual tracking problems [9], sound power level optimization [29] and Sudoku problems[10].

As far as we know, the majority of previous works on HS focused on solving the optimization problems in discrete or continuous space and only a few works investigated binary problems. In 2005, Geem [11] firstly adopted the standard HS with binary-coding (BHS) to solve water pump switching

problems in which the pitch adjustment operator was discarded. Then, Geem and Williams [15] utilized BHS to tackle an ecologic optimization problem and achieved the better results than those using the Simulated Annealing algorithm. Later, Greblicki and Kotowski [17] analyzed the properties of BHS on one dimensional binary knapsack problems and the experimental results indicated that the performance of BHS was not ideal. Wang et al. [47] pointed out that the dysfunction of the pitch adjustment rule degraded the search ability of BHS and redesigned the pitch adjustment operation in the proposed discrete binary harmony search (DBHS) algorithm. Afterwards, they extended DBHS to tackle the Pareto-based multi-objective optimization problems [44]. Recently, Zou et al. [51] modified a novel global HS (NGHS) to solve the 0-1 knapsack problem in which the updating strategy of the real-coded HS was used and binary solutions were generated by replacing the real values with the nearest integers. In summary, the research on binary-coded HS algorithms has only begun, and the performance of the current binary HS algorithms is still not satisfactory. Therefore, in this work we propose an improved adaptive binary harmony search (ABHS) algorithm for solving the binary-coded problems more efficiently and effectively.

The rest of this paper is organized as follows. In section 2, the standard HS is briefly introduced. Then the proposed ABHS algorithm is elaborated in section 3. Section 4 conducts a series of experiments to analyze the properties of ABHS. In section 5, the benchmark functions and 0-1 knapsack problems are used to verify the optimization performance of ABHS. The comparisons with BHS, NGHS and discrete binary PSO (DBPSO) are also given. Finally, the conclusions are drawn in Section 6.

2 Harmony Search Algorithm

HS is developed by imitating the music improvisation process where music players improve the pitch of their instruments to obtain the better harmony. The optimization problem can be described as follows:

$$Minimizing\ f(x)\ subject\ to\ Lx_i \le x_i \le Ux_i, i = 1, 2, \ldots\ldots, M \qquad (1)$$

where $f(x)$ is the objective function; x_i is the decision variable; Lx_i and Ux_i are the lower and upper bounds of the feasible domain; M is the number of decision variables. The implementation of HS is as follows:

Step 1) Initializing the algorithm parameters and the harmony memory. The algorithmic parameters include the harmony memory size (HMS), the harmony memory considering rate (HMCR), the pitch adjusting rate (PAR) and the maximum number of improvisations. The harmony memory $HM = \left[x^1 \; x^2 \ldots x^{HMS} \right]^T$ is randomly initialized within the feasible solution space according to the HMS.

Step 2) Improvising new harmonies. A new harmony denoted as $x' = \left(x_1', x_2' \cdots x_{M-1}', x_M' \right)$ is improvised by using the harmony memory consideration rule, the pitch adjustment rule and randomization, which are determined by the pre-defined HMCR and PAR. The HMCR is the probability of choosing one value from the harmony memory (HM) while (1-HMCR) represents the probability of randomly picking up one value among all the feasible values. For instance, if a uniform random number between 0 and 1 is less than the HMCR, x_i' is chosen from the HM. In addition, the element of *x'* chosen from the HM needs to be judged whether it should be pitch-adjusted with the probability PAR. If the random number is less than PAR, x_i' will be replaced by $x_i' \pm rand() \times bw$ where *bw* is the bandwidth.

Step 3) Updating the harmony memory. If the new generated harmony vector x' performs better than the worst one in the HM, the new one is included in the HM and the corresponding worst candidate is excluded.

Step 4) Checking the termination criterion. If the stopping criterion is satisfied, the iterative search is terminated and the optimal solution is output. Otherwise, Step 2 and step 3 are repeated.

More details on the standard HS algorithm can be found in [27].

3 Adaptive Binary Harmony Search Algorithm

Although the HS with binary-coding can be used to tackle binary-coded problems, the pitch adjustment operator is inferior in binary space, which spoils the performance of the algorithm. To make up for it, the drawback of HS for binary-valued problems is analyzed and discussed, and an adaptive binary HS algorithm with a novel pitch adjustment operator is proposed to improve the optimization ability.

3.1 Initialization of the harmony memory

In ABHS, individuals are represented as binary strings, and the HM, denoted as *H*, is initialized with random binary numbers as Eq. (2):

$$H=\begin{bmatrix} h_{11} & h_{12} \ldots & h_{1j} & \ldots h_{1M} \\ h_{21} & h_{22} \ldots & h_{2j} & \ldots h_{2M} \\ \ldots & & & \\ h_{i1} & h_{i2} \ldots & h_{ij} & \ldots h_{iM} \\ \ldots & & & \\ h_{HMS,1} & h_{HMS,2} \ldots & h_{HMS,j} & \ldots h_{HMS,M} \end{bmatrix} \quad h_{ij}\in\{0,1\}, i\in\{1,2,\ldots,HMS\}, j\in\{1,2,\ldots,M\} \tag{2}$$

where *HMS* is the size of the *HM*; h_{ij} is the element of the *i-th* harmony vector; *M* is the length of harmony vectors, i.e. the dimension of solutions.

3.2 Harmony memory consideration rule

The harmony memory consideration rate indicates whether the element of new candidates is generated from the HM or randomization. Imitating the standard HS, the HMCR is defined as the probability of picking out a value from the HM while (1-HMCR) is the rate of randomly choosing a feasible value not limited to the HM; that is, it is re-initialized stochastically to be "0" or "1" in ABHS. In this work, two strategies are presented to implement harmony memory consideration operation: the bit selection strategy and the individual selection strategy.

(1) Bit selection strategy

For the bit selection strategy, each element of harmony vectors is independently chosen from the HM; that is, each bit of new solutions is chosen from the different harmony memory vectors randomly as Eq. (3-4)

$$x_{ij}=\begin{cases} h_{pj} & \text{if } r_1 < HMCR \\ R & else \end{cases} \tag{3}$$

$$R=\begin{cases} 0 & \text{if } r_2 < 0.5 \\ 1 & else \end{cases} \tag{4}$$

where x_{ij} is the *j-th* bit of the new improvised harmony vector x_i; $p\in\{1,\ldots,HMS\}$ is a random integer decided by $\lceil r_3\times HMS\rceil$ and generated for each bit independently; r_1, r_2 and r_3 are three independent random numbers between 0 and 1.

(2) Individual selection strategy

In the individual selection strategy, all the elements of solutions are only selected from one HM vector. The individual strategy operation can be defined as Eq. (5-6)

$$x_{ij} = \begin{cases} h_{tj},\ t \in \{1,2,...,HMS\} & r_1 < HMCR \\ R & else \end{cases} \quad (5)$$

$$R = \begin{cases} 0 & \text{if } r_2 < 0.5 \\ 1 & else \end{cases} \quad (6)$$

where x_{ij} is the *j-th* element of the new candidate x_i ; t is a random integer which does not change during generating x_i ; r_1 and r_2 are two independent random numbers between 0 and 1.

The difference between these two strategies can be depicted as Fig. 1; that is, the elements of the new harmony vector can be picked up from the different vectors in the HM for the bit selection strategy while the individual strategy only chooses the value from one harmony vector in the HM for a new candidate.

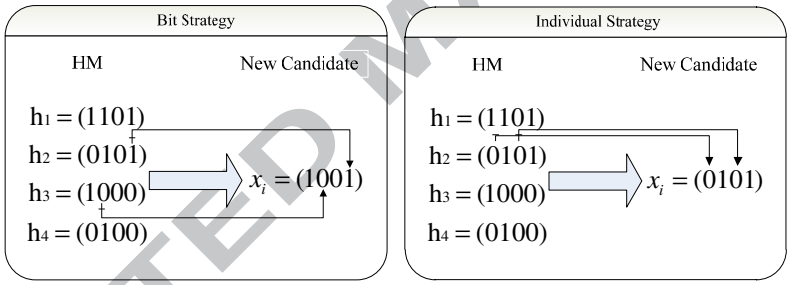


Fig. 1 Bit and individual selection strategies

3.3 Pitch adjustment rule

The pitch adjustment operator chooses an adjacent value with the probability PAR in the standard HS. For binary optimization problems, the solution has only two values, i.e. “0” or “1”. Thus the standard pitch adjustment operator is a NOT gate for binary-coded problems as shown in Eq. (7-8)

$$x_{ij} = \begin{cases} \overline{x_{ij}} & r < PAR \\ x_{ij} & else \end{cases} \quad (7)$$

$$\overline{x_{ij}} = \begin{cases} 1 & if\ h_{ij} = 0 \\ 0 & if\ h_{ij} = 1 \end{cases} \quad (8)$$

where r is a random number between 0 and 1. According to Eq. (7-8), we can find that the pitch adjustment operator is transformed into a mutation operator for binary optimization problems. It should be noted that the randomization operator has introduced a mutation operation in the process of the harmony memory consideration with the probability (1-*HMCR*)/2, which means that the total mutation probability of the standard BHS is p_m as Eq.(9)

$$p_m = \frac{2 \times HMCR \times PAR + (1 - HMCR)}{2} \tag{9}$$

The pitch adjustment operator is utilized to find locally improved solutions, but it is degraded to be the mutation operator in the standard BHS. Therefore, it is reasonable that the optimization ability of BHS is spoiled. Due to this fact, Geem [11] abandoned the pitch adjustment rule in his work while Greblicki [17] retained the pitch adjustment operator. Although the mutation operation can perform a local search, the efficiency and effectiveness are not satisfactory. To improve it, we adopt a new pitch adjustment rule which chooses the adjacent value from its structural neighborhood rather than an adjacent value in the harmony memory. For easy implementation, the neighbor of each HS vector is defined as the global optimal harmony vector in the HM. Thus, the pitch adjustment rule in ABHS is improved as Eq. (10)

$$x_{ij} = \begin{cases} h_{bj} & r < PAR \\ x_{ij} & else \end{cases} \tag{10}$$

where h_{bj} denotes the corresponding element of the global optimal harmony vector in the HM.

3.4 Adaptive mechanism

The optimization ability of meta-heuristic algorithms, including HS, relies on the parameter setting. To improve the robustness and flexibility of algorithms, adaptive parameter strategies [37] are adopted in evolutionary algorithms such as PSO [7, 28, 49], Differential Evolution [3] and Ant Colony Optimization [46]. As mentioned above, adaptive strategies are also introduced into the HS algorithms to enhance the performance [6, 18, 27, 40, 43]. However, it is difficult to determine which adaptive method is the best choice for the binary-coded HS. On the one hand, various adaptive methods, even conflicted ones [27, 42], were proposed for the standard HS. On the other hand, the characteristics of the binary-coded HS are different from those of the standard HS, and therefore the performance of adaptive methods may change. Thus, a variety of adaptive HMCR and PAR mechanisms including the

linear increment, linear decrease, nonlinear increment and random increment as Eq. (11-22) are studied and analyzed to improve the optimization ability and the flexibility of the proposed binary HS.

$$HMCR = HMCR_{\min} + \frac{HMCR_{\max} - HMCR_{\min}}{iter_{\max}} \times iter \tag{11}$$

$$HMCR = HMCR_{\max} - \frac{HMCR_{\max} - HMCR_{\min}}{iter_{\max}} \times iter \tag{12}$$

$$\begin{cases} HMCR = HMCR_{\min} e^{(c \cdot iter)} \\ c = \dfrac{Ln\left(\dfrac{HMCR_{\max}}{HMCR_{\min}}\right)}{iter_{\max}} \end{cases} \tag{13}$$

$$HMCR = HMCR_{\min} + \left(HMCR_{\max} - HMCR_{\min}\right) \frac{1}{\left(\ln\left(iter\right)+1\right)} \tag{14}$$

$$HMCR = HMCR_{\min} + \frac{e}{e^{-\frac{iter}{iter_{\max}}}} \times \Delta HMCR \quad \Delta HMCR = 0.02 \tag{15}$$

$$HMCR = HMCR_{\min} + rand_1 \times \left(HMCR_{\max} - HMCR_{\min}\right) \tag{16}$$

$$\mathrm{PAR} = \mathrm{PAR}_{\min} + \frac{\mathrm{PAR}_{\max} - \mathrm{PAR}_{\min}}{iter_{\max}} \times iter \tag{17}$$

$$\mathrm{PAR} = \mathrm{PAR}_{\max} - \frac{\mathrm{PAR}_{\max} - \mathrm{PAR}_{\min}}{iter_{\max}} \times iter \tag{18}$$

$$\begin{cases} PAR = PAR_{\min} e^{(c \cdot iter)} \\ c = \dfrac{Ln\left(\dfrac{PAR_{\max}}{PAR_{\min}}\right)}{iter_{\max}} \end{cases} \tag{19}$$

$$PAR = PAR_{\min} + \left(PAR_{\max} - PAR_{\min}\right) \frac{1}{\left(\ln\left(iter\right)+1\right)} \tag{20}$$

$$PAR = PAR_{\min} + \frac{e}{e^{-\frac{iter}{iter_{\max}}}} \times \Delta PAR \tag{21}$$

$$PAR = PAR_{\min} + rand_1 \times \left(PAR_{\max} - PAR_{\min}\right) \tag{22}$$

$HMCR_{\min}$ / $PAR_{\min}$ denotes the lower boundary of the *HMCR/PAR*; $HMCR_{\max}$ / $PAR_{\max}$ is the upper boundary of the *HMCR/PAR*; *Iter* and $Iter_{\max}$ represent the current iteration number and the maximum iteration number, respectively. Fig. 2 depicts the dynamical changes of the *HMCR* with these

six adaptive mechanisms which are totally different, and thus it is vital to choose a proper adaptive method to improve the performance of the algorithm.

In summary, the procedure of ABHS can be described as follows.

Step1: Set the parameters such as the HMS, the HMCR and the PAR.

Step2: Initialize the harmony memory.

Step3: Improvise new harmonies by conducting the adaptive harmony memory consideration, the pitch adjustment and randomization.

Step4: Update the harmony memory and the global best harmony vector.

Step5: Repeat step 3 and step 4 until the termination criteria are satisfied.

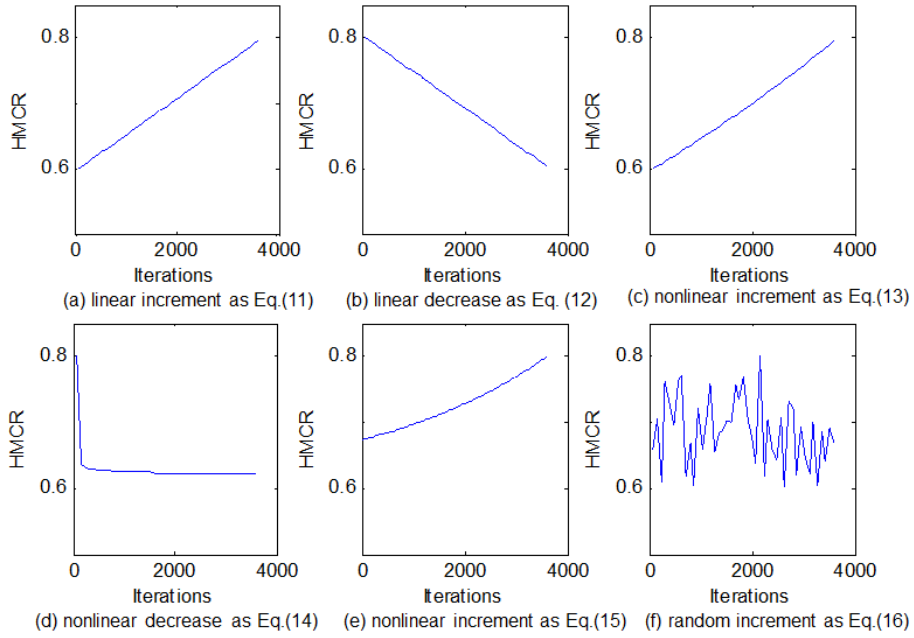


Fig. 2 Variation of the HMCR versus the iteration number

3.5 Algorithm complexity

ABHS consists of two phases, i.e. the initialization and the iterative search. The generation and sorting of the initial HM has time complexity $O(HMS\times M+HMS\times log(HMS))$ where M is the dimension of solutions. During the iterative search, harmony vectors are generated to search for the optimal solution. The generation of each new harmony vector costs time $O(M\times(1+HMCR\times PAR))$, and the new candidate needs to be checked and replaces the worst vector in the HM which has complexity $O(log(HMS)+M)$. However, one or multiple candidates can be generated in each generation of ABHS.

Thus, one iteration takes $O(NGC\times(M\times(1+HMCR\times PAR)+log(HMS)+M))$ where NGC stands for the number of the new candidates. NGC=1 means that ABHS adopts the serial updating strategy as the standard HS algorithm while NGC>1 indicates that the parallel updating mechanism is used, which denotes that multiple candidates are generated and used to update the HM in each iteration. Usually, the iterative search process is repeated till the pre-defined fitness calculation number (FCN) is reached; that is, the iteration repeats FCN/NGC times. Hence, the iterative search phase has time complexity $O(FCN/NGC\times(NGC\times(M\times(1+HMCR\times PAR)+log(HMS)+M))$. In summary, the overall complexity of the proposed ABHS is $O(HMS\times(M+log(HMS))+FCN\times(M\times(2+HMCR\times PAR)+log(HMS)))$.

4 Parameter analysis of ABHS

Like other meta-heuristic algorithms, ABHS is sensitive to the parameter setting. Although we hardly expect to find the general optimal parameter values for ABHS as they are problem-dependent, it is possible to set the fair parameter values to achieve the satisfactory optimization performance. Thus, two unimodal functions and two multimodal functions of 36 benchmark functions [16, 32, 34, 49], i.e. F2, F3, F11 and F17, are used for the parameter analysis to avoid the improper parameter setting of ABHS. The characteristics of all the functions are listed in Table 1-2.

4.1 The number of new harmony vectors generated

The original HS adopts the serial updating strategy; that is, it improvises one new candidate and then updates the HM immediately. However, the parallel updating strategy [36] has been reported to have the better performance for various optimization problems. To verify the performance clearly and exactly, ABHS based on the bit selection and the individual selection with the different number of new generating candidates (NGC) were studied, and no adaptive parameter was employed to eliminate the disturbance from the adaptive mechanism. The HMCR and PAR of ABHS were set as the values recommended in [14, 17], i.e. HMCR=0.74 and PAR=0.1; the HMS was 30. For a fair comparison, the maximum number of fitness calculating was set as a constant, i.e. 90000. Four criteria, i.e. the best value (BV), the mean best value (MBV), the rate of finding the global optima (RGO) and the fitness calculation number (FCN), were adopted to evaluate the optimization performance. All the tests were run 50 times independently. The results are given in Table 3 and Table 4.

Table 1. Characteristics of Benchmark Functions

	Name	dimension	Optima	Type
F1	Sphere	2/30	0.0(min)	unimodal
F2	Rosenbrock	2/30	0.0(min)	unimodal
F3	Goldstein&Price	2	3.0(min)	unimodal
F4	Glankwahdee	2	0.0(min)	unimodal
F5	Griewangk	2/30	0.0(min)	unimodal
F6	Freudenstein-Roth	2	0.0(min)	unimodal
F7	Beale	2	0.0(min)	unimodal
F8	Sum of different power	2/30	0(min)	unimodal
F9	Schwefel 2.22	2/30	0(min)	unimodal
F10	Rastrigin	2/30	0(min)	multimodal
F11	Schwefel 2.26	2/30	-837.96577(min) /-12569.5(min)	multimodal
F12	Ackley's path	2/30	0.0(min)	multimodal
F13	Branin	2	0.3978874(min)	multimodal
F14	Schaffer F6	2	1.0(max)	multimodal
F15	Levy F3	2	-176.5417(min)	multimodal
F16	Camel-6	2	-1.031628(min)	multimodal
F17	Alpine	2	7.8856(max)	multimodal
F18	schaffer F7	2	0(min)	multimodal
F19	Levy F5	2	-176.1375 (min)	multimodal
F20	Bohachevsky 2	2	0(min)	multimodal

Table 2**.** Characteristics of CEC05 Benchmark Functions

	Name	Dimension	Optima	Type
F21	Shifted Sphere Function	2/30	-450	unimodal
F22	Shifted Schwefel's Problem 1.2	2/30	-450	unimodal
F23	Shifted Rotated High Conditioned Elliptic Function	2/30	-450	unimodal
F24	Shifted Schwefel's Problem 1.2 with Noise in Fitness	2	-450	unimodal
F25	Schwefel's Problem 2.6 with Global Optimum on Bounds	2	-310	unimodal
F26	Shifted Rosenbrock's Function	2/30	390	multimodal
F27	Shifted Rotated Griewank's Function without Bounds	2/30	-180	multimodal
F28	Shifted Rotated Ackley's Function with Global Optimum on Bounds	2	-140	multimodal
F29	Shifted Rastrigin's Function	2/30	-330	multimodal
F30	Shifted Rotated Rastrigin's Function	2	-330	multimodal
F31	Shifted Rotated Weierstrass Function	2	90	multimodal
F32	Schwefel's Problem 2.13	2	-460	multimodal
F33	Expanded Extended Griewank's plus Rosenbrock's Function	2	-130	multimodal
F34	Shifted Rotated Expanded Scaffer's F6	2/30	-300	multimodal
F35	Hybrid Composition Function	2/30	120	multimodal
F36	Rotated Hybrid Composition Function	2	120	multimodal

Table 3 and Table 4 show that the parallel updating strategy is beneficial to the global optimization ability of ABHS, especially on the multimodal functions. In the serial updating, only one new harmony vector is yielded based on the current HM, and the HM will be updated instantly. But in the parallel updating, multiple candidates are generated in each generation, which means that the present worse vectors in the HM have more chance to be chosen to create new solutions. Obviously, the selection pressure in the HM is effectively alleviated in the parallel updating method. Thus, on the one hand, the parallel updating strategy can maintain the diversity of the HM better, which helps the algorithm escape from the local optima and improves the global search ability efficiently. On the other hand, compared with the original serial updating strategy, the convergence speed of the parallel updating is

slowed on the unimodal problems in which there are no local optima. However, a small NGC, for instance less than 15, cannot present this advantage, and too big an NGC may decrease the efficiency of ABHS, which can be observed both in the individual and bit selection methods.

Obviously, the difference between the bit selection strategy and the individual selection strategy affects the optimization performance. For the serial updating strategy, the bit selection method can maintain the better diversity and achieves the better optimization results. But the individual selection method is prior for the parallel updating mechanism. According to the results in Table 3-4, ABHS based on the individual selection with NGC=20 achieves the best optimization performance, which is adopted in the following sections.

Table 3**.** The results of ABHS with the different NGC based on the individual selection strategy

	NGC	1	5	10	15	20	30
F2	BV	0.0	0.0	0.0	0.0	**0.0**	0.0
	MBV	3.25E-9	9.74E-8	1.74E-9	1.98E-10	**1.75E-11**	2.48E-8
	RGO	40%	22%	30%	40%	**44%**	42%
	FCN	66095	45560	77040	59790	**71600**	69960
F3	BV	3.0	3.0	3.0	3.0	**3.0**	3.0
	MBV	3+1.91E-08	3+1.99E-08	3+1.61E-08	3+1.50E-08	**3+1.45E-08**	3+1.66E-08
	RGO	24%	32%	36%	40%	**42%**	34%
	FCN	71286	65700	62802	46155	**58320**	64950
F11	BV	-837.96577	-837.96577	-837.96577	-837.96577	**-837.96577**	-837.96577
	MBV	-837.62955	-837.96568	-837.96570	-837.96571	**-837.96571**	-837.96570
	RGO	40%	42%	48%	60%	**64%**	54%
	FCN	62053	15540	27209	33030	**38640**	67590
F17	BV	7.885601	7.885601	7.885601	7.885601	**7.885601**	7.885601
	MBV	7.865510	7.867054	7.869099	7.876906	**7.877155**	7.873681
	RGO	44%	40%	58%	64%	**66%**	60%
	FCN	56202	51724	43812	50472	**34853**	35642

Table 4. The results of ABHS with the different NGC based on the bit selection strategy

	NGC	1	5	10	15	20	30
F2	BV	0.0	0.0	0.0	0.0	0.0	**0.0**
	MBV	5.35E-11	2.31E-10	1.44E-10	4.19E-10	7.45E-11	**5.23E-11**
	RGO	44%	24%	32%	38%	42%	**50%**
	FCN	71951	81305	78979	73854	73891	**72610**
F3	BV	3.0	3.0	3.0	**3.0**	3.0	3.0
	MBV	3+1.68E-08	3+2.16E-08	3+2.06E-08	**3+1.61E-08**	3+1.96E-08	3+1.96E-08
	RGO	26%	14%	18%	**30%**	22%	22%
	FCN	71286	78775	80430	**63588**	73073	72667
F11	BV	-837.96577	-837.96577	-837.96577	-837.96577	-837.96577	**-837.96577**
	MBV	-837.96568	-837.96568	-837.96568	-837.96571	-837.96570	**-837.96573**
	RGO	36%	40%	38%	40%	42%	**48%**
	FCN	60370	57450	59004	58627	57658	**51897**
F17	BV	7.885601	7.885601	7.885601	7.885601	**7.885601**	7.885601
	MBV	7.870760	7.867254	7.863590	7.866346	**7.871849**	7.870010
	RGO	50%	40%	44%	46%	**62%**	52%
	FCN	40041	50486	63687	51376	**39401**	44985

4.2 HMS

ABHS yields new solutions based on the HM, and therefore the HMS influences the performance of the algorithm. A set of the HMS values, i.e. 1, 5, 20, 30, 40, 60, 100, 140 and 180, were studied to evaluate its influence on the optimization ability. HMCR=0.74 and PAR=0.1 were adopted and NGC was set as 20. The adaptive mechanisms were not introduced to evade the disturbance in the performance analysis. The experimental results of the 50 independent runs are presented in Table 5.

Table 5. The results of ABHS with the different HMS

	HMS	1	5	20	30	40	60	100	140	180
F2	BV	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0	0.0	0.0
	MBV	3.20E-9	1.54E-9	4.35E-10	**1.75E-11**	8.13E-10	1.10E-9	3.51E-9	1.18E-8	7.33E-8
	RGO	24%	26%	36%	**44%**	32%	28%	20%	14%	6%
	FCN	69419	65768	70727	**71600**	73932	79402	86792	88709	89018
F3	BV	3.0	3.0	3.0	**3.0**	3.0	3.0	3.0	3.0	3.0
	MBV	3+2.24E-08	3+2.05E-08	3+2.13E-08	**3+1.45E-08**	3+2.03E-08	3+2.38E-08	3+2.42E-08	3+2.43E-08	3+2.35E-08
	RGO	30%	34%	34%	**42%**	36%	26%	26%	24%	24%
	FCN	74396	69236	71503	**58320**	65377	81248	81520	72419	80040
F11	BV	-837.96577	-837.96577	-837.96577	-837.96577	-837.96577	-837.96577	-837.96577	-837.96577	**-837.96577**
	MBV	-837.96566	-837.96567	-837.96569	-837.96571	-837.96572	-837.96572	-837.96572	-837.96573	**-837.96573**
	RGO	32%	40%	46%	64%	66%	64%	66%	70%	**72%**
	FCN	64919	59592	51300	38640	36991	40422	29801	35190	**34126**
F17	BV	7.885601	7.885601	7.885601	**7.885601**	7.885601	7.885601	7.885601	7.885601	7.885601
	MBV	7.872386	7.871289	7.872440	**7.877155**	7.874240	7.870639	7.871928	7.870972	7.871035
	RGO	52%	52%	54%	**66%**	58%	52%	52%	50%	52%
	FCN	56202	46118	51937	**34853**	33953	492981	45624	49681	45304

According to the results in Table 5, it is obvious that the optimal HMS is problem-dependent. The optimization performance of F2 is degraded with the increment of the HMS when it is more than 30, while the contrary result, i.e. the enhanced performance, is observed on F11. However, the HMS should not be too small as the results of all the functions become worse with the decreasing of the HMS when it is less than 30. The convergence curves of ABHS with HMS=1, 30 and 180 on F2 and F11 are depicted in Fig. 3 and Fig. 4, respectively. As expected, we can find that the better initial fitness values are obtained with the increasing of the HMS because more initial solutions are generated. Meanwhile,

Fig. 3 and Fig. 4 show that the size of the HM does not significantly influence the convergence speed of the algorithm. Therefore, HMS=30 is adopted in this work based on the comprehensive results of the four functions.

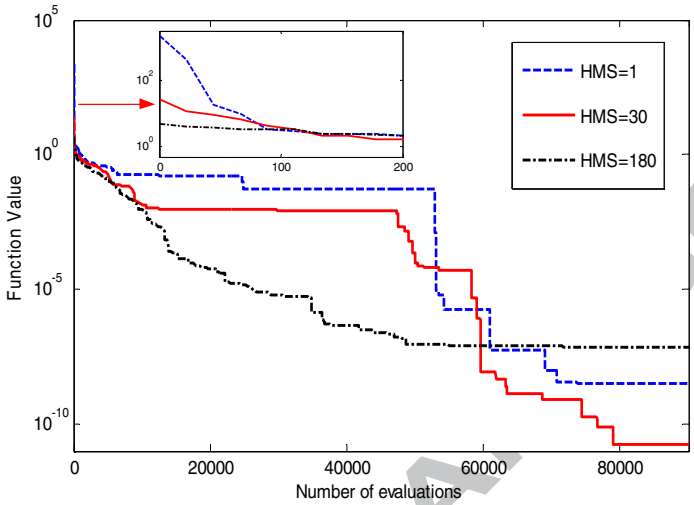


Fig.3 Convergence graph of F2 with different HMS

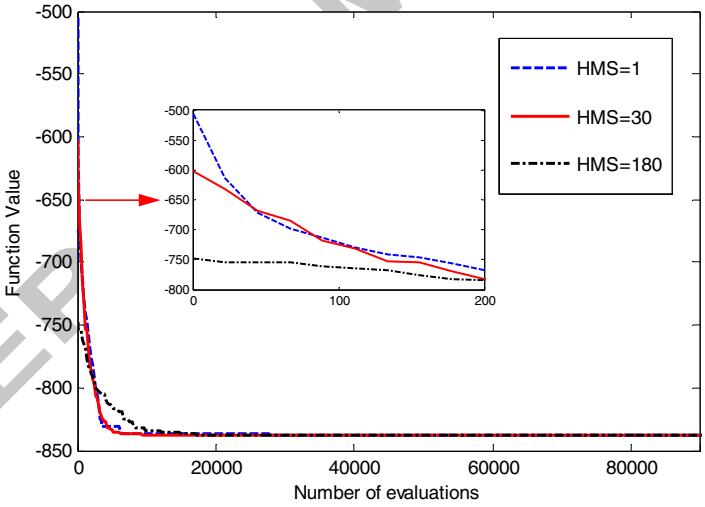


Fig.4 Convergence graph of F11 with different HMS

4.3 HMCR and PAR

The HMCR indicates that the new candidate is generated from the HM or randomization, and the PAR determines whether the new candidate needs to be refined. Thus, the HMCR and the PAR are of great importance as they determine the capabilities of the global search and the local search. Apparently, there is a coupling between the HMCR and the PAR on the optimization ability. To investigate their

influence on ABHS, the parameter studies were performed with the HMCR and the PAR tuned from 0.1 to 0.9 and 0.05 to 0.9, respectively. The adaptive mechanisms were not introduced to avoid the disturbance in the analysis. The optimization results of F2, F3, F11 and F17 are illustrated in Fig. 5-8 where the three axes represent the HMCR, the PAR and the success rate (SR) of finding the global optima in the 50 independent runs, respectively.

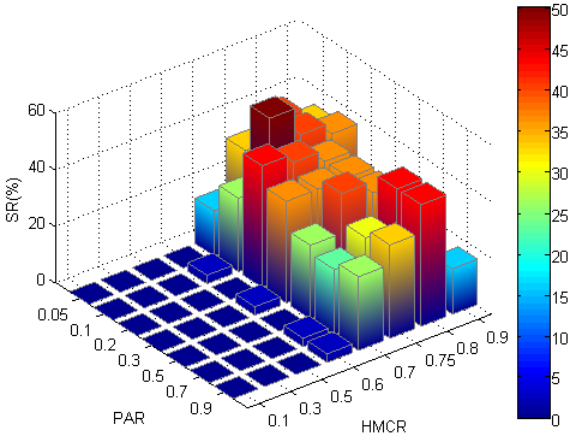


Fig. 5 The influence of PAR and HMCR on F2

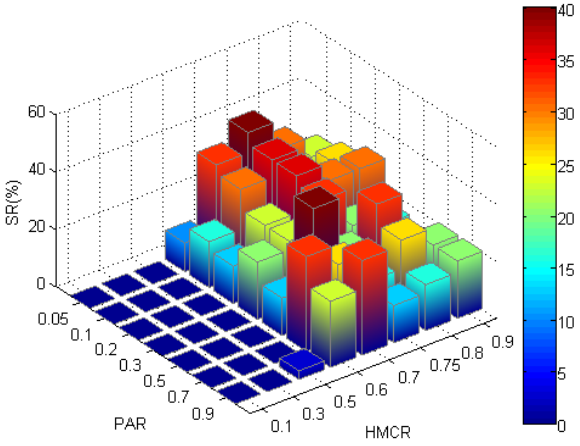


Fig. 6 The influence of PAR and HMCR on F3

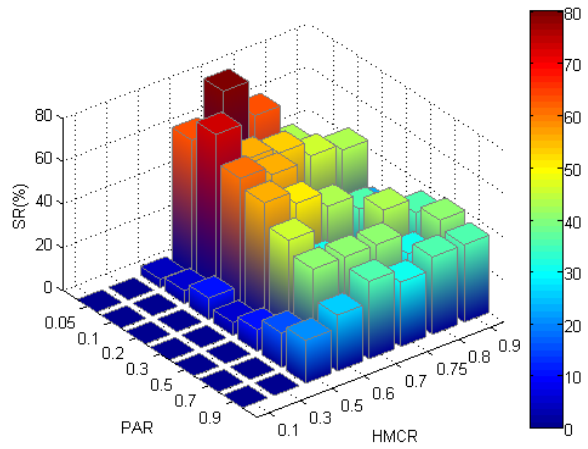


Fig. 7 The influence of PAR and HMCR on F11

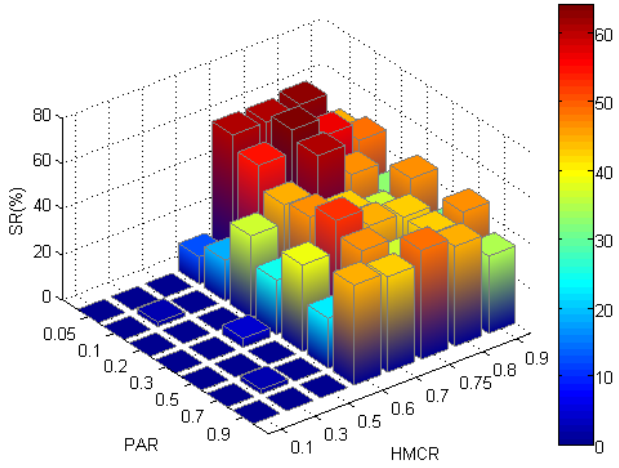


Fig. 8 The influence of PAR and HMCR on F17

Fig. 5-8 show that the HMCR and the PAR play important roles in the algorithm performance, and ABHS is more sensitive to the HMCR. ABHS gains the better performance when the HMCR is between 0.6 and 0.8, while its optimization ability is severely spoiled if the HMCR is out of this range. Compared with the unimodal functions, the multimodal optimization problems prefer to a small HMCR, which means that a higher randomization rate can keep the diversity more efficiently to help the algorithm escape from the local optima. Although ABHS is less sensitive to the PAR, an improper PAR can degrade ABHS as well. As depicted in the Fig. 5-8, a small PAR, less than 0.3, is recommended especially for the multimodal functions. Overall, the results reveal that the optimal HMCR and PAR of ABHS are different for various optimization problems, and it is hard to set the optimal parameters without a priori knowledge. However, it is not difficult to choose the proper

parameter values to achieve the fair performance for ABHS. Based on the results of four functions, the recommended values of the HMCR and the PAR are from 0.6 to 0.8 and from 0.05 to 0.3, respectively.

4.4 Adaptive mechanism

ABHS is sensitive to the parameter setting. Thus, the adaptive mechanisms presented in section 3.4 are investigated to improve the flexibility and optimization ability. The success rate of the different adaptive strategies and the constant parameters are plotted in the Fig. 9-12. The definitions of the axes in the figures are listed in Table 6 and the parameter values adopted are given in Table 7.

Table 6. The definition of the adaptive mechanism combinations

HMCR Axis		PAR Axis	
Label	Strategy	Label	Strategy
1	Constant (0.7)	1	Constant(0.2)
2	Eq. (11)	2	Eq. (17)
3	Eq. (12)	3	Eq. (18)
4	Eq. (13)	4	Eq. (19)
5	Eq. (14)	5	Eq. (20)
6	Eq. (15)	6	Eq. (21)
7	Eq. (16)	7	Eq. (22)

Table 7. The parameter values of adaptive methods

Parameter	Value
$HMCR_{min}$	0.6
$HMCR_{max}$	0.8
$\Delta HMCR$	0.02
PAR_{min}	0.05
PAR_{max}	0.3
ΔPAR	0.03
$iter_{max}$	4500

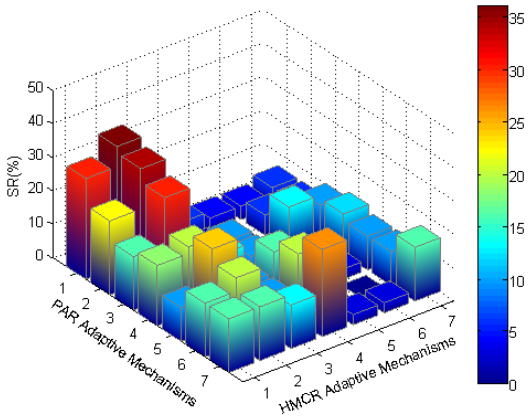


Fig. 9 The optimization performance of the adaptive mechanisms on F2

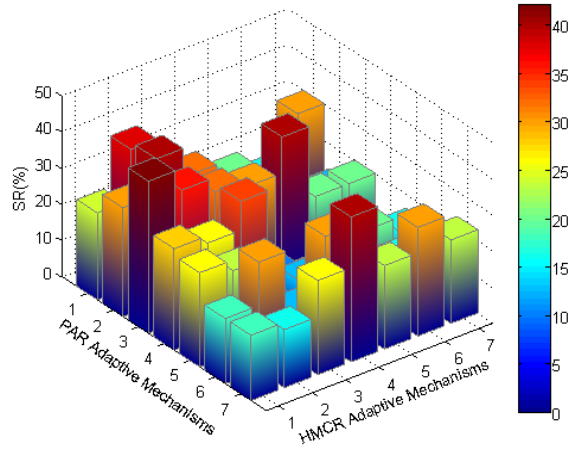


Fig. 10 The optimization performance of the adaptive mechanisms on F3

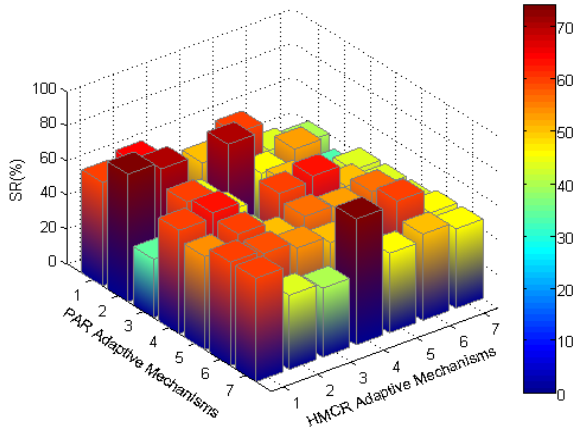


Fig. 11 The optimization performance of the adaptive mechanisms on F11

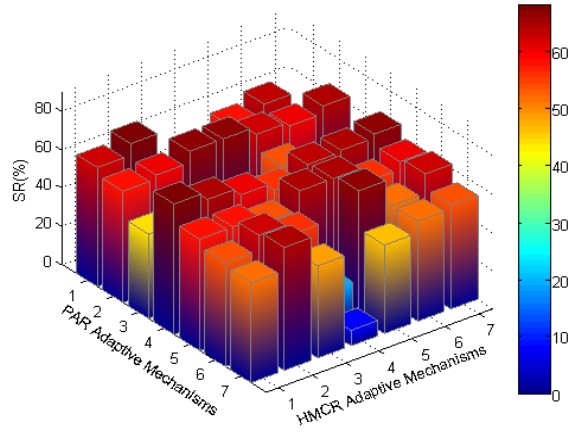


Fig. 12 The optimization performance of the adaptive mechanisms on F17

It is obvious that a proper adaptive strategy can efficiently improve the optimization ability of the algorithm while an improper adaptive method spoils the performance. On various functions, the optimal adaptive combination strategy is different. It is reasonable as the best parameter values are

problem-dependent. Based on the results displayed in Fig. 9-12, the overall best performance of ABHS is achieved when (a) the HMCR linearly increases as Eq. (11) and the PAR is a constant, (b) the HMCR and the PAR both linearly increase using Eq. (11) and Eq. (17). To reduce the computation cost and simplify the algorithm, the former adaptive strategy, i.e. the linearly incremental HMCR and the constant PAR, is adopted in this paper. Considering the scalability of the algorithm, the adaptive linearly increasing HMCR finally used is defined as Eq. (23) based on the overall results of the parameter analysis,

$$HMCR = \left(1 - \frac{C}{M}\right) + \frac{\lfloor \ln M \rfloor}{M} + \frac{\ln M}{M} \times \frac{iter}{iter_{\max}} \tag{23}$$

where M is the dimension of solutions and C is a constant. The first term on the right of Eq. (23) ensures that certain elements are generated by the randomization operator, which can effectively prevent the algorithm from trapping in the local optima regardless of the change of problem dimensionality. The value of the second term on the right of Eq. (23) varies following the change of the dimension which slightly tunes the HMCR. These two parts correspond to the first item of Eq. (11). The third term on the right of Eq. (23) is the adaptive factor that dynamically and linearly adjusts HMCR based on the dimension and the number of the iteration.

5 Experimental results and discussions

To verify the optimization ability and scalability of the algorithm, ABHS was applied to optimize the benchmark functions listed in Table 1 and Table 2. All the functions were tested in 2 dimensions, and sixteen of them which could be extended to the high dimension were also evaluated in 30 dimensions. Furthermore, 10 low-dimensional and 10 high-dimensional 0-1 knapsack problems were adopted as benchmarks as well. For comparisons, the HS with binary-coding (BHS) [11], the binary-coded novel global harmony search (NGHS) [51] and the discrete binary PSO (DBPSO) [19, 45] with the recommended parameter settings were also used to solve these benchmark problems. The parameter values of all the algorithms are listed in Table 8.

Table 8**.** The parameters of ABHS, BHS, NGHS and DBPSO

Parameter setting	Parameter setting

ABHS	C=15;PAR=0.2;HMS=30; NGC=20;	BHS [11]	HMCR=0.971;HMS=19; NGC = 1;
NGHS [51]	Pm=2/M; NGC = 1; HMS=5(low-dimensional problems); HMS=30(high-dimensional problems);	DBPSO [19, 45]	c_1=2;c_2=2;ω=0.8;vmax=6;vmin=-6; population = 30;

5.1 Benchmark functions

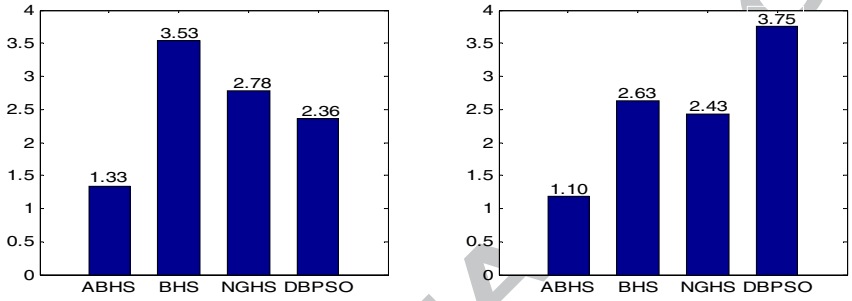
The fitness calculation number of all the algorithms for all the benchmark functions was set to 90000. Table 9-11 show the optimization results of the 36 low-dimensional functions and the 16 high-dimensional functions. The t-test results of all functions are also given in these three tables where "1" or "-1" denotes that the results obtained by the proposed method, i.e. ABHS, are significantly better or worse than those of the compared algorithm at the 95% confidence, respectively, while "0" means that the achieved results are not statistically different. To show the performance clearly, Fig. 13 draws the average rank based on the RGO, MBV and FCN sorted by the descending priority. It is obvious that the proposed ABHS outperforms BHS, NGHS and DBPSO.

On the 36 low-dimensional functions, ABHS is only inferior to BHS, NGHS and DBPSO on 3, 3 and 6 functions, respectively. In Table 9, we can find that ABHS is poorer than BHS and NGHS on F1, F8 and F9 just due to the slower convergence speed. F1, F8 and F9 are all the unimodal functions and easy for optimizing, thus four algorithms reach the global optima with 100% success rate. As BHS and NGHS both adopt the serial updating strategy, it is not surprising that they are superior to ABHS and DBPSO on the convergence speed. However, the serial updating method increases the risk of sticking in the local optima, thus BHS and NGHS perform worse than ABHS on the other functions. DBPSO obtains the best results on six functions and outperforms BHS and NGHS on the most functions. Fig.14-25 draw the average convergence curves of all the algorithms in 50 runs where we can observe that ABHS surpasses BHS, NGHS and DBPSO in terms of the accuracy and convergence speed.

The search space of the 30-dimensional functions is expanded to 2^{600} as each decision variable is coded by 20 bits, which is a challenge for ABHS, NGHS, BHS and DBPSO. Thus, BHS and DBPSO fail to reach any global optima of all the high-dimensional functions while ABHS and NGHS find out the global best values of 5 and 4 functions, respectively. The results in Table 11 show that ABHS outperforms BHS and DBPSO on all the functions and is only inferior to NGHS on three functions, i.e. F1, F8 and F9 due to the slow convergence speed. As mentioned above, F1, F8 and F9 are relatively easy for optimizing, and therefore NGHS as well as ABHS searches out the global optima with a 100%

success rate. NGHS adopts the serial updating strategy, thus it achieves the fastest convergence on these three functions. However, this advantage of NGHS becomes the disadvantage for the complicated optimization functions as NGHS is likely to stick in the local optima. The average convergence graphs of ABHS, NGHS, BHS and DBPSO on the high-dimensional functions in 50 runs are drawn in Fig.26-37 where we can find that the convergence speed of NGHS is obviously poorer than those of ABHS on the high-dimensional F2, F5, F11 and F12.

Based on the results of Table 9-11 and Fig.14-37, it is fair to claim that ABHS has a better optimization ability and scalability compared with NGHS, BHS and DBPSO.



(a) the low-dimensional functions (b) the high-dimensional functions

Fig. 13 The average rank of the four algorithms on the benchmark functions

Table 9**.** The low-dimensional function optimization results of ABHS, BHS, NGHS and DBPSO

		ABHS	BHS	NGHS	DBPSO			ABHS	BHS	NGHS	DBPSO
F1	BV	0	**0**	0	0	F11	BV	**-837.96577**	-837.96577	-837.96577	-837.96577
	RGO	100%	**100%**	100%	100%		RGO	**54%**	14%	28%	46%
	MBV	0	**0**	0	0		MBV	**-837.96570**	-836.83069	-837.40648	-837.96571
	FCN	14540	**388**	415.02	33090		FCN	**46142**	79462	68520	25306
	t-test	/	**0**	0	0		t-test	/	1	1	0
F2	BV	**0**	5.82E-09	0	0	F12	BV	**0%**	0	0	0
	RGO	**32%**	0%	4%	6%		RGO	**100%**	96%	92%	100%
	MBV	**5.05E-10**	4.12E-01	1.90E-01	1.02E-08		MBV	**0**	7.31E-01	1.56E-01	0
	FCN	**73682**	90000	88144	47439		FCN	**16275**	11368	19742	31888
	t-test	/	1	1	1		t-test	/	0	0	0
F3	BV	**3**	3	3	3	F13	BV	**0.3978874**	0.3978927	0.3978874	0.3978874
	RGO	**48%**	18%	18%	32%		RGO	**90%**	0%	28%	74 %
	MBV	**3**	28.1636	19.3637	3		MBV	**0.3979258**	1.3631266	0.4756296	0.3979479
	FCN	**54592**	74116	74465	38670		FCN	**31810**	90000	77419	37403
	t-test	/	1	0	0		t-test	/	1	1	0
F4	BV	**0**	0	0	0	F14	BV	-1	-1	-1	**-1**
	RGO	**14%**	6%	8%	2%		RGO	56%	8%	6%	**78%**
	MBV	**8.70E-06**	2.49E+00	5.85E-01	3.59E-04		MBV	-0.995725	-0.984775	-0.989766	**-0.998445**
	FCN	**79600**	86610	82960	43215		FCN	54221	87132.56	86814	**49291**
	t-test	/	1	1	0		t-test	/	1	1	**-1**
F5	BV	0	0	0	**0**	F15	BV	**-176.5417**	-176.5417	-176.5418	-176.5417
	RGO	60%	14%	48%	**76%**		RGO	**96%**	24%	42%	94%
	MBV	0.002958	0.027353	0.010937	**0.001482**		MBV	**-176.541593**	-164.647753	-173.763723	-176.541561
	FCN	47537	77964.76	56971	**40568**		FCN	**15113**	77576	60672	25165
	t-test	/	1	0	**0**		t-test	/	1	1	0
F6	BV	**0**	0	0	0	F16	BV	**-1.031628**	-1.031628	-1.0316285	-1.031628
	RGO	**18%**	12%	2%	10%		RGO	**96%**	22%	46%	64%
	MBV	**9.36E-07**	2.39E+01	1.10E+01	9.01E-07		MBV	**-1.031628**	-1.026981	-1.030869	-1.031621
	FCN	**76743**	80672	89258	39270		FCN	**27605**	74443	61121	43850
	t-test	/	1	0	0		t-test	/	1	1	0
F7	BV	**0**	0	0	0	F17	BV	**7.8856**	7.8856	7.8856	7.8856

	RGO	**48%**	20%	20%	18%		RGO	**46%**	8%	22%	42%
	MBV	**5.01E-10**	3.25E-01	1.22E-02	1.29E-09		MBV	**7.86854**	5.9817	6.6875	7.8679
	FCN	**59247**	80914	76797	44208		FCN	**52454**	85744	76601	19317
	t-test	**/**	1	0	0		t-test	**/**	1	1	0
F8	BV	-1	-1	**-1**	-1	F18	BV	**0**	0	0	0
	RGO	100%	100%	**100%**	100%		RGO	**100%**	98%	100%	92%
	MBV	-1	-1	**-1**	-1		MBV	**0.00**	0.002557	0	0.000724
	FCN	15194	414	**407**	32876		FCN	**19561**	11811	23180	41602
	t-test	/	0	**0**	0		t-test	**/**	0	0	0
F9	BV	0	**0**	0	0	F19	BV	**-176.1375**	-176.1375	-176.1376	-176.1375
	RGO	100%	**100%**	100%	100%		RGO	**98%**	2%	30%	96%
	MBV	0.00	**0**	0	0.00E+00		MBV	**-176.1370**	-146.4581	-159.2101	-176.1372
	FCN	14265	**388**	12920	31020		FCN	**16163**	89564	76063.62	29581
	t-test	/	**0**	0	0		t-test	**/**	1	0	0
F10	BV	**0**	0	0	0	F20	BV	**0**	0	0	0
	RGO	**100%**	24%	36%	100%		RGO	**100%**	94%	100%	100%
	MBV	**0**	1.014860	0.736270	0		MBV	**0**	0.013105	0	0
	FCN	**18306**	68492	58249	35356		FCN	**14708**	7949	15282	32786
	t-test	**/**	1	1	0		t-test	**/**	0	0	0

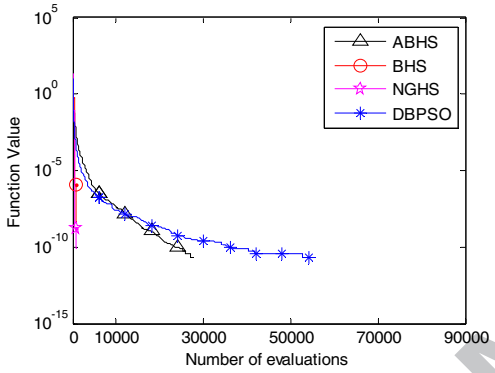


Fig.14 Convergence graph of the low-dimensional F1

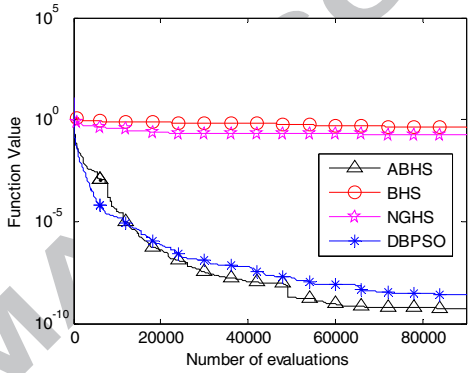


Fig.15 Convergence graph of the low-dimensional F2

Table 10**.** The low-dimensional CEC05 function optimization results of ABHS, BHS, NGHS and DBPSO

		ABHS	BHS	NGHS	DBPSO			**ABHS**	BHS	NGHS	DBPSO
F21	BV	**1.23E-08**	1.23E-08	1.23E-08	1.23E-08	F29	BV	**1.20E-08**	4.27E-08	1.20E-08	4.27E-08
	RGO	**0%**	0%	0%	0%		RGO	**0%**	0%	0%	0%
	MBV	**1.98E-07**	1.46E+01	6.93E+00	2.80E-07		MBV	**1.71E-04**	1.39E+00	7.11E-02	2.92E-04
	FCN	**90000**	90000	90000	90000		FCN	**90000**	90000	90000	90000
	t-test	**/**	1	1	0		t-test	**/**	1	0	0
F22	BV	2.06E-09	6.61E-08	6.61E-08	**2.06E-09**	F30	BV	1.21E-08	8.87E-06	9.24E-08	**1.21E-08**
	RGO	0%	0%	0%	**0%**		RGO	0	0	0	**0%**
	MBV	2.83E-01	9.18E+01	6.21E+00	**2.34E-04**		MBV	2.43E-01	2.24E+00	9.44E-01	**1.46E-01**
	FCN	90000	90000	90000	**90000**		FCN	90000	90000	90000	**90000**
	t-test	/	1	1	**0**		t-test	/	1	1	**0**
F23	BV	**7.23E-03**	2.19E+00	1.31E+01	3.09E+00	F31	BV	**9.58E-04**	9.58E-04	3.22E-03	5.16E-03
	RGO	**0%**	0%	0%	0%		RGO	**0%**	0%	0%	0%
	MBV	**6.16E+01**	4.72E+05	6.17E+03	1.07E+02		MBV	**2.63E-02**	4.17E-01	1.59E-01	3.23E-02
	FCN	**90000**	90000	90000	90000		FCN	**90000**	90000	90000	90000
	t-test	**/**	1	1	0		t-test	**/**	1	1	0
F24	BV	2.06E-09	6.80E-03	6.61E-08	**2.10E-08**	F32	BV	**1.85E-08**	4.86E-08	1.65E-07	1.31E+00
	RGO	0%	0%	0%	**0%**		RGO	**0**	0%	0%	0%
	MBV	2.89E-01	1.45E+02	1.34E+01	**5.09E-03**		MBV	**9.96E-03**	9.26E+00	7.21E+00	7.77E+01
	FCN	90000	90000	90000	**90000**		FCN	**90000**	90000	90000	90000
	t-test	/	1	1	**-1**		t-test	**/**	0	1	1
F25	BV	**0**	0	0	0	F33	BV	**0**	6.77E-07	9.95E-13	0
	RGO	**32%**	6%	14%	23%		RGO	**12%**	0%	0%	4%
	MBV	**9.43E-03**	1.51E+02	8.95E+01	1.15E-02		MBV	**5.84E-03**	5.03E-02	3.37E-02	6.01E-03
	FCN	**71536**	86612	85632	81352		FCN	**82696**	90000	90000	89041
	t-test	**/**	1	1	0		t-test	**/**	1	1	0
F26	BV	**3.95E-09**	2.23E-02	3.95E-09	3.95E-09	F34	BV	**4.96E-08**	1.94E-02	3.59E-05	7.24E-08
	RGO	**0%**	0%	0%	0%		RGO	**0%**	0%	0%	0%
	MBV	**2.84E-01**	9.97E+01	4.29E+01	3.75E-01		MBV	**1.55E-02**	2.16E-01	1.51E-01	9.99E-01
	FCN	**90000**	90000	90000	90000		FCN	**90000**	90000	90000	90000

	t-test	/	0	0	0		t-test	/	1	1	1
F27	BV	**1.53E-06**	3.04E-06	7.40E-03	1.53E-06	F35	BV	4.26E-07	4.26E-07	4.26E-07	**4.26E-07**
	RGO	**0%**	0%	0%	0%		RGO	0%	0%	0%	**0%**
	MBV	**3.12E-02**	1.30E-01	9.63E-02	3.24E-02		MBV	2.00E+00	1.42E+02	4.89E+01	**6.00E-01**
	FCN	**90000**	90000	90000	90000		FCN	90000	90000	90000	**90000**
	t-test	/	1	0	0		t-test	/	1	1	**0**
F28	BV	**1.83E-03**	3.22E-03	2.03E-03	1.61E-02	F36	BV	**5.49E-07**	5.13E-02	4.63E-05	5.49E-07
	RGO	**0%**	0%	0%	0%		RGO	**0%**	0%	0%	0%
	MBV	**1.52E+00**	1.73E+01	1.38E+01	2.02E+00		MBV	**1.33E+01**	1.48E+02	6.28E+01	1.58E+01
	FCN	**90000**	90000	90000	90000		FCN	**90000**	90000	90000	90000
	t-test	/	0	0	0		t-test	/	1	1	0

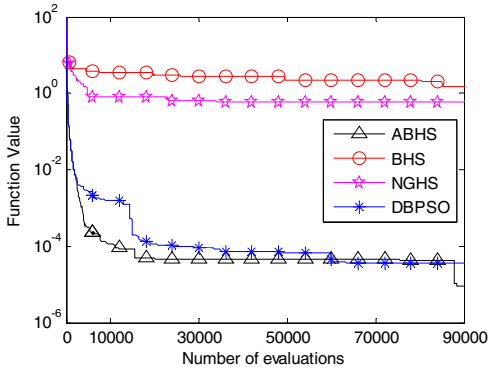


Fig.16 Convergence graph of the low-dimensional F4

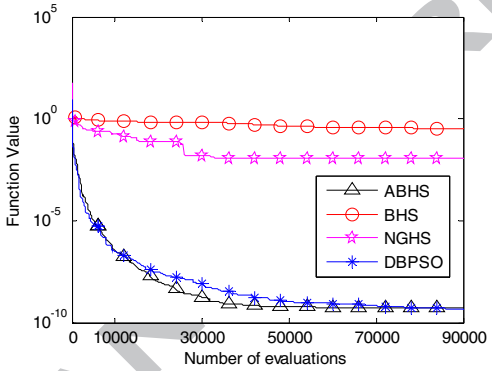


Fig.17 Convergence graph of the low-dimensional F7

Table 11**.** The high-dimensional function optimization results of ABHS, BHS, NGHS and DBPSO

		ABHS	BHS	NGHS	DBPSO			ABHS	BHS	NGHS	DBPSO
F1	BV	0	1.60E-06	**0**	1.35E+02	F21	BV	**-278.383977**	-253.178301	-247.205974	10233.71184
	RGO	100%	0%	**100%**	0%		RGO	**0%**	0%	0%	0%
	MBV	0	7.72E-06	**0**	1.86E+02		MBV	**100.033756**	343.866819	410.293866	16549.86863
	FCN	62234	90000	**16535**	90000		FCN	**90000**	90000	90000	90000
	t-test	/	1	0	1		t-test	/	1	1	1
F2	BV	**7.82162**	15.31827	45.53963	1210.60138	F22	BV	**1970.91705**	5687.88402	4314.02218	31917.39486
	RGO	**0%**	0%	0%	0%		RGO	**0%**	0%	0%	0%
	MBV	**679.92240**	817.69248	1063.4419	2422.02256		MBV	**6373.8376**	18240.2003	11912.2755	44480.9849
	FCN	**90000**	90000	90000	90000		FCN	**90000**	90000	90000	90000
	t-test	/	1	1	1		t-test	/	1	1	1
F5	BV	**0**	1.25E-07	0	1.04E+00	F23	BV	**7.59E+06**	4.14E+07	3.37E+07	1.83E+08
	RGO	**38%**	0%	10%	0%		RGO	**0%**	0%	0%	0%
	MBV	**0.033037**	0.056922	0.137763	1.046698		MBV	**7.98E+07**	1.23E+08	1.26E+08	3.31E+08
	FCN	**79758**	90000	82710.66	90000		FCN	**90000**	90000	90000	90000
	t-test	/	1	0	1		t-test	/	1	1	1
F8	BV	0	5.19E-47	**0**	1.79E-02	F26	BV	**8.19E+05**	1.06E+06	1.13E+06	1.41E+09
	RGO	100%	0%	**100%**	0%		RGO	**0%**	0%	0%	0%
	MBV	0	2.53E-31	**0**	8.72E-02		MBV	**1.42E+07**	1.53E+07	2.94E+07	2.49E+09
	FCN	80371	90000	**33583**	90000		FCN	**90000**	90000	90000	90000
	t-test	/	0	**0**	1		t-test	/	1	1	1
F9	BV	0	0.004456	**0**	49.121522	F27	BV	**-164.708843**	-162.982001	-162.212717	437.8451747
	RGO	100%	0%	**100%**	0 %		RGO	**0%**	0%	0%	0%
	MBV	0	0.008674	**0**	60.948356		MBV	**-117.310561**	-74.904286	-93.683347	900.790971
	FCN	59870	90000	**13953**	90000		FCN	**90000**	90000	90000	90000
	t-test	/	1	0	1		t-test	/	1	1	1
F10	BV	**6.965127**	7.962070	10.949991	221.388412	F29	BV	**-3.23E+02**	-3.12E+02	-3.10E+02	-9.58E+01
	RGO	**0%**	0%	0%	0%		RGO	**0%**	0%	0%	0%
	MBV	**13.156419**	13.257949	13.934090	256.261875		MBV	**-3.09E+02**	-2.87E+02	-2.86E+02	-4.82E+01
	FCN	**90000**	90000	90000	90000		FCN	**90000**	90000	90000	90000
	t-test	/	0	0	1		t-test	/	1	1	1

F11	BV	**-12442.114**	-12134.971	-11677.452	-6878.777	F34	BV	**-2.88E+02**	-2.87E+02	-2.86E+02	-2.87E+02
	RGO	**0%**	0%	0%	0 %		RGO	**0%**	0%	0%	0%
	MBV	**-11946.553**	-11729.410	-10767.115	-6277.884		MBV	**-2.87E+02**	-2.86E+02	-2.86E+02	-2.86E+02
	FCN	**90000**	90000	90000	90000		FCN	**90000**	90000	90000	90000
	t-test	**/**	0	0	1		t-test	**/**	1	1	1
F12	BV	**0**	0.005828	1.900188	19.942523	F35	BV	**2.24E+02**	4.01E+02	4.71E+02	7.64E+02
	RGO	**90%**	0%	0%	0%		RGO	**0%**	0%	0%	0%
	MBV	**0.156398**	0.398168	2.274827	19.951260		MBV	**5.13E+02**	6.17E+02	5.83E+02	9.39E+02
	FCN	**62350**	90000	90000	90000		FCN	**90000**	90000	90000	90000
	t-test	**/**	0	1	1		t-test	**/**	1	0	1

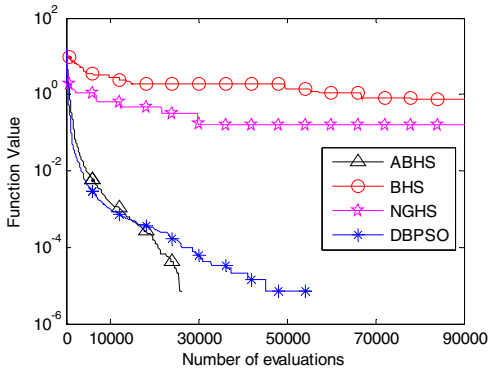


Fig.18 Convergence graph of the low-dimensional F12

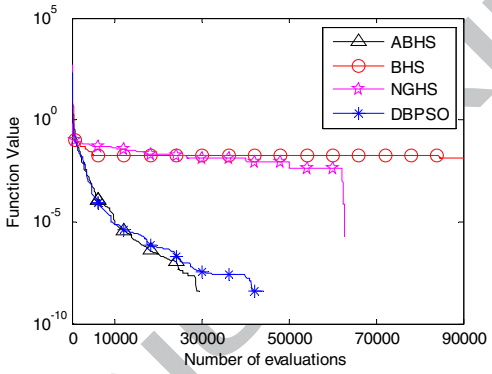


Fig.19 Convergence graph of the low-dimensional F20

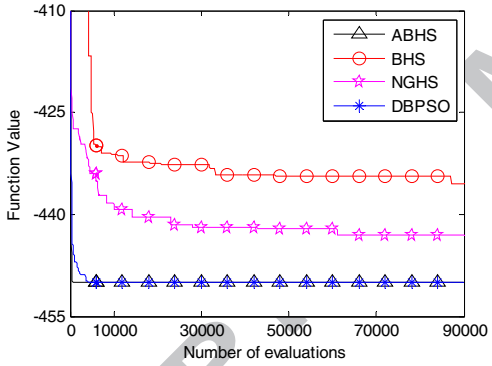


Fig.20 Convergence graph of the low-dimensional F21

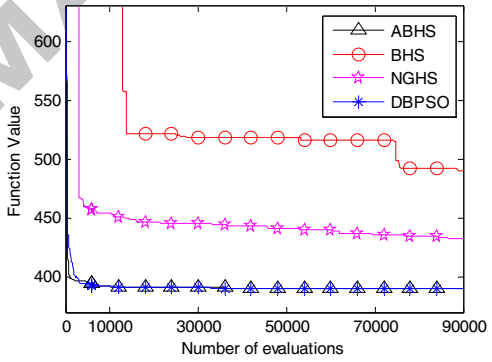


Fig.21 Convergence graph of the low-dimensional F26

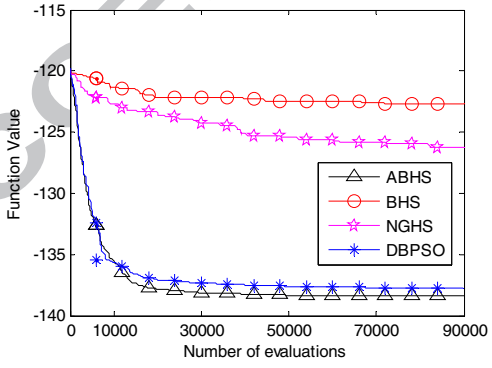


Fig.22 Convergence graph of the low-dimensional F28

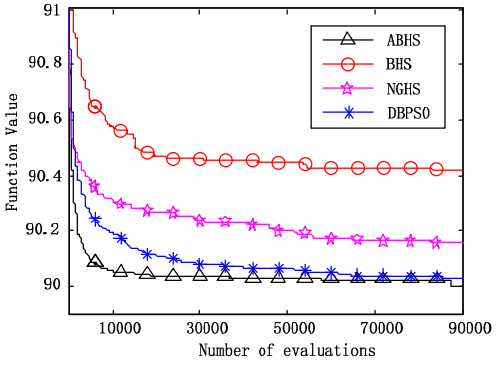


Fig.23 Convergence graph of the low-dimensional F31

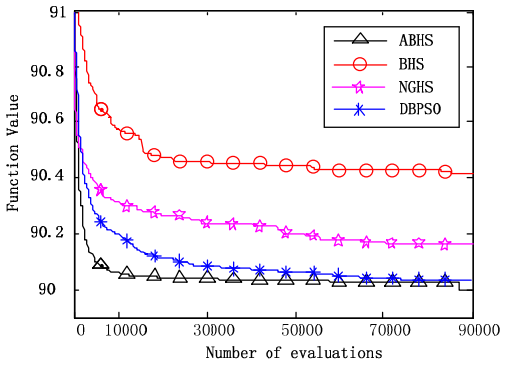


Fig.24 Convergence graph of the low-dimensional F34

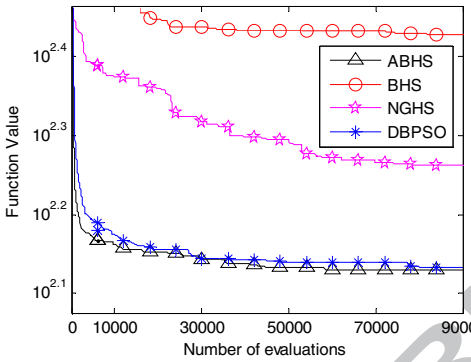


Fig.25 Convergence graph of the low-dimensional F36

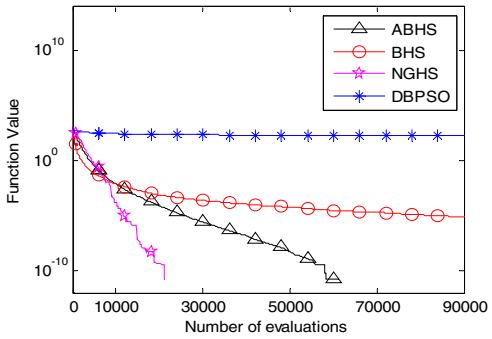


Fig.26 Convergence graph of the high-dimensional F1

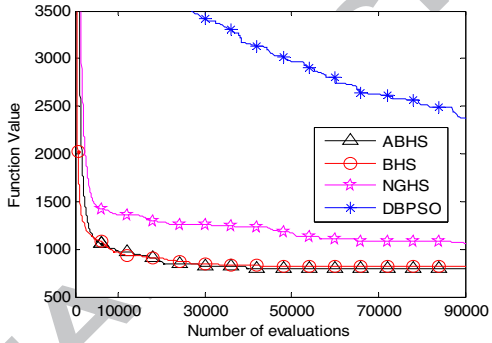


Fig.27 Convergence graph of the high-dimensional F2

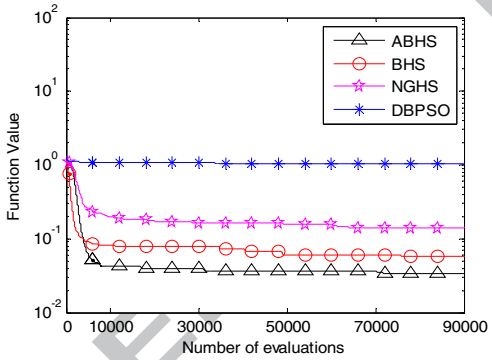


Fig.28 Convergence graph of the high-dimensional F5

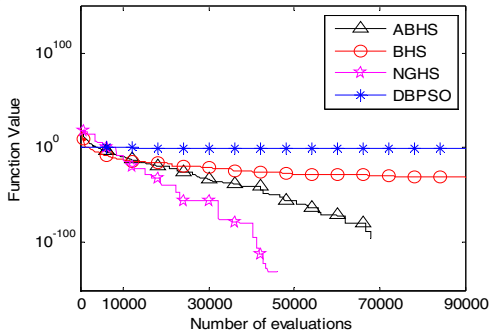


Fig.29 Convergence graph of the high-dimensional F8

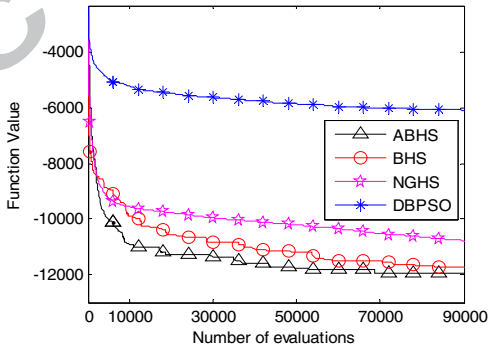


Fig.30 Convergence graph of the high-dimensional F11

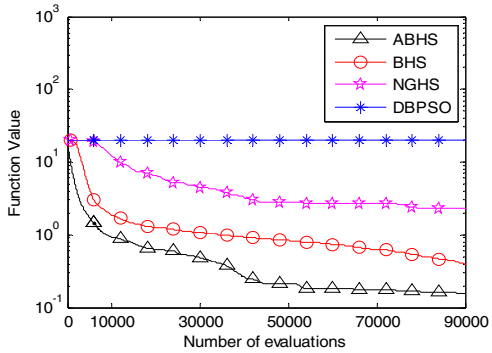


Fig.31 Convergence graph of the high-dimensional F12

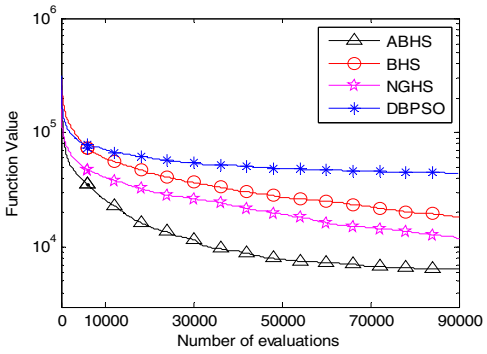


Fig.32 Convergence graph of the high-dimensional F22

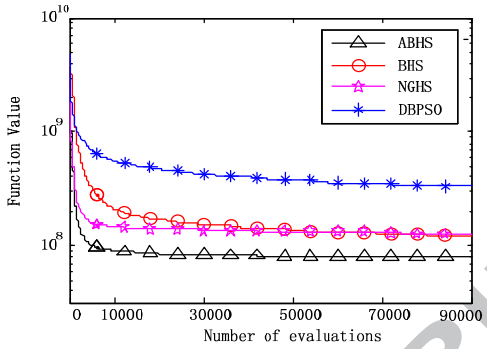


Fig.33 Convergence graph of the high-dimensional F23

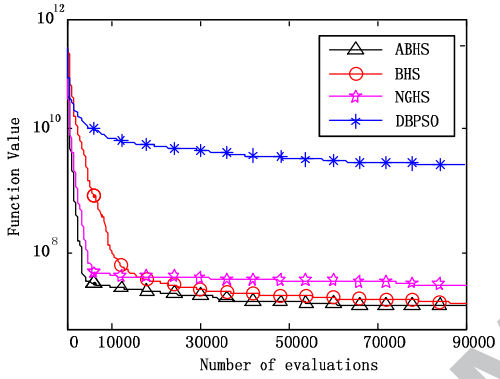


Fig.34 Convergence graph of the high-dimensional F26

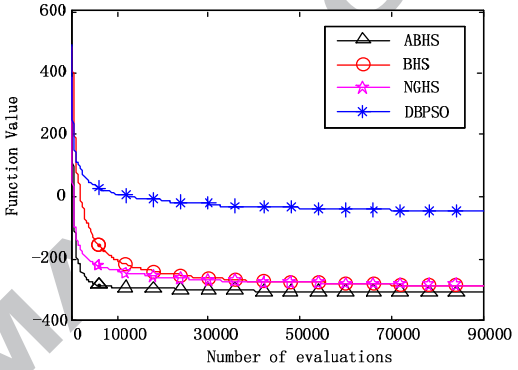


Fig.35 Convergence graph of the high-dimensional F29

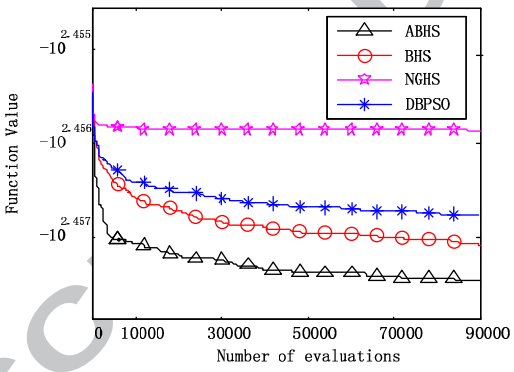


Fig.36 Convergence graph of the high-dimensional F34

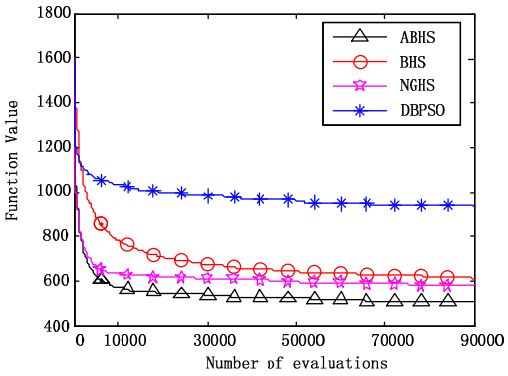


Fig.37 Convergence graph of the high-dimensional F35

5.2 0-1 knapsack problem

The 0-1 knapsack problem (0-1 KP) is one of the classical NP-complete problems, which can be described as Eq. (24):

$$Max\ f(x) = \sum_{i=1}^{n} p_i x_i$$
$$s.t \begin{cases} \sum_{i=1}^{n} w_i x_i \le C \\ x_i = 0\ or\ 1 \quad i = 1, 2, \ldots, N \end{cases} \tag{24}$$

where x_i indicates whether item i is included in the knapsack or not; N is the number of items; p_i is the profit of item i; w_i is the weight of item i ; C is the knapsack capacity. When the total weight exceeds the limit C, the penalty function is introduced to fix the fitness and lead the algorithm to search in the feasible area effectively as Eq. (25):

$$MaxF(x) = f(x) - \lambda \times \max(0, g)$$
$$g = \sum_{i=1}^{n} w_i x_i - C \tag{25}$$

where λ , called the penalty coefficient, is a big constant which guarantees that the fitness of the best infeasible solution is poorer than that of the worst feasible solution.

ABHS was verified and compared with BHS, NGHS and DBPSO on the low-dimensional and high-dimensional 0-1 KPs. Ten low-dimensional 0-1 KP instances used in [51] are adopted as the low-dimensional benchmarks. As there is no high-dimensional 0-1 KP benchmark, ten high-dimensional cases are generated following the instruction in [49]; that is, the weight w_i is the integer between 5 and 20 and the profit p_i is randomly set between 50 and 100. The details of the low-dimensional and high-dimensional 0-1 KPs used in this paper are described in Table 12 and Table 13, respectively. ABHS, BHS, NGHS and DBPSO ran on each 0-1 KP instance 50 times independently. The optimization results are given in Table 14 and Table 15.

Table 12**.** The parameters of the low-dimensional 0-1 knapsack problems

Instance	D	Maximum FCN	Parameter
Kp1	10	10000	w=(95,4,60,32,23,72,80,62,65,4,6),C=269,p=(55,10,47,5,4,50,8,61,85,87)
Kp2	20	10000	w= (92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58), C = 878, p = (44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29,75,
Kp3	4	10000	w=(6,5,9,7),C =20, p=(9,11,13,15)
Kp4	4	10000	w=(2,4,6,7),C =11, p=(6,10,12,13)
Kp5	15	10000	w= (56.358531, 80.874050, 47.987304,89.596240, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204,16.589862, 44.569231, 0.466933,37.788018, 57.118442, 60.716575), C = 375, p = (0.125126, 19.330424,58.500931, 35.029145, 82.284005, 17.410810, 71.050142, 30.399487,9.140294, 14.731285, 98.852504, 11.908322, 0.891140, 53.166295, 60.176397)
Kp6	10	10000	w=(30,25,20,18,17,11,5,2,1,1), C=60, p=(20,18,17,15,15,10,5,3,1,1)

Kp7	7	10000	w=(31,10,20,19,4,3,6),C=50, p=(70,20,39,37,7,5,10)
Kp8	23	10000	w= (983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959), C = 10000, p = (981, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857)
Kp9	5	10000	w=(15,20,17,8,31),C=80, p=(33,24,36,37,12)
Kp10	20	10000	w= (84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92), C = 879, p = (91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44)

Table 13. The parameters of the high-dimensional 0-1 knapsack problems

Instance	Dimension	Capacity	Maximum FCN
Kp11	100	1100	15000
Kp12	500	4000	20000
Kp13	1000	10000	30000
Kp14	1200	14000	40000
Kp15	1400	15000	40000
Kp16	1600	18000	50000
Kp17	1800	20000	50000
Kp18	2000	22000	50000
Kp19	2200	24000	60000
Kp20	2500	26000	60000

The optimization of the low-dimensional 0-1 KP instances is simple, thus all the algorithms find the global optima of each case. However, BHS suffers from low dimensionality and its performance is greatly impaired because of the improper parameters so that it even cannot find the maximum profit with 100% success rate on the 4-dimensional 0-1 KPs, of which the search space is only 16. With the increasing of the dimension, 0-1 KPs become more and more complicated. From the results of Table 15, we can observe that ABHS gains an overwhelming performance over the other three algorithms on the high-dimensional 0-1 KPs and it achieves the best results on all the instances. DBPSO maybe is not suitable for solving these high-dimensional 0-1 KPs as its performance is obviously poorer than that of ABHS, BHS and NGHS. Thus, only the average convergence curves of ABHS, BHS and NGHS are drawn in Fig.38-47 for depicting their performance clearly. Fig. 38-47 show that BHS and NGHS both have a quick convergence speed at the beginning of the iteration due to the serial updating mechanism. However, BHS and NGHS are likely to be harassed by the local optima so that their searching efficiency and effectiveness are spoiled. ABHS gradually exceeds BHS and NGHS and obtains the best

search accuracy and convergence speed finally. BHS surpasses NGHS on Kp11 and Kp12. But with the increasing of the dimension, the instances are more complex and the performance of BHS becomes worse than that of NGHS as the abandonment of the pitch adjustment operator weakens its optimization ability.

Table 14. The results of ABHS, BHS, NGHS and DBPSO on the low-dimensional 0-1 knapsack problems

		ABHS	BHS	NGHS	DBPSO			ABHS	BHS	NGHS	DBPSO
Kp1	SR	100%	78%	100%	100%	Kp6	SR	100%	82%	100%	100%
	Best	295	295	295	295		Best	52	52	52	52
	Mean	295	295	295	295		Mean	52	51.62	52	52
	Std.dev	0	1.20	0	0		Std.dev	0	0.94	0	0
	t-test	/	1	0	0		t-test	/	1	0	0
Kp2	SR	100%	92%	100%	100%	Kp7	SR	100%	62%	100%	100%
	Best	1024	1024	1024	1024		Best	107	107	107	107
	Mean	1024	1023.52	1024	1024		Mean	107	105.64	107	107
	Std.dev	0	1.63	0	0		Std.dev	0	2.86	0	0
	t-test	/	1	0	0		t-test	/	1	0	0
Kp3	SR	100%	98%	100%	100%	Kp8	SR	100%	94%	100%	100%
	Best	35	35	35	35		Best	9767	9767	9767	9767
	Mean	35	34.86	35	35		Mean	9767	9766.8	9767	9767
	Std.dev	0	0.98	0	0		Std.dev	0	0.85	0	0
	t-test	/	0	0	0		t-test	/	1	0	0
Kp4	SR	100%	98%	100%	100%	Kp9	SR	100%	98%	100%	100%
	Best	23	23	23	23		Best	130	130	130	130
	Mean	23	22.98	23	23		Mean	130	129.76	130	130
	Std.dev	0	0.14	0	0		Std.dev	0	1.68	0	0
	t-test	/	0	0	0		t-test	/	0	0	0
Kp5	SR	100%	88%	100%	100%	Kp10	SR	100%	94%	100%	100%
	Best	481.07	481.07	481.07	481.07		Best	1025	1025	1025	1025
	Mean	481.07	476.50	481.07	481.07		Mean	1025	1024.64	1025	1025
	Std.dev	0	13.28	0	0		Std.dev	0	1.42	0	0
	t-test	/	1	0	0		t-test	/	0	0	0

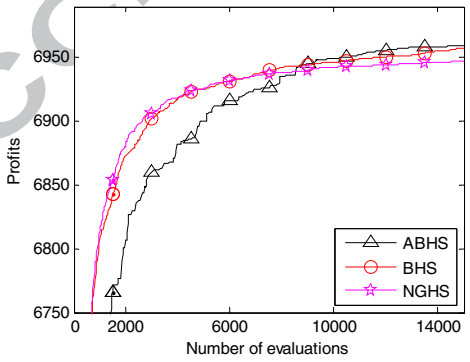


Fig.38 Convergence graph of Kp11

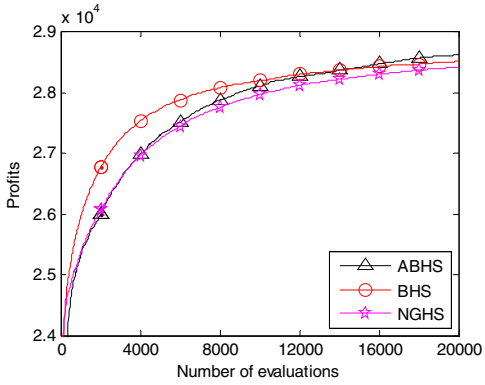


Fig.39 Convergence graph of Kp12

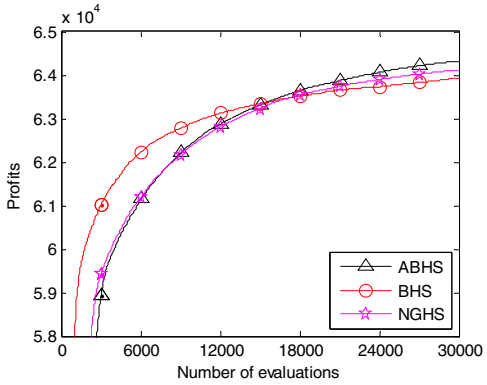


Fig.40 Convergence graph of Kp13

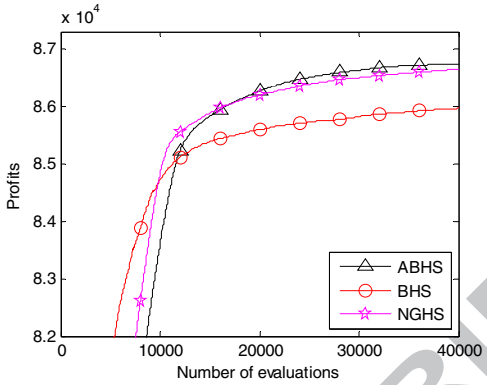


Fig.41 Convergence graph of Kp14

Table 15. The results of ABHS, BHS, NGHS and DBPSO on the high-dimensional 0-1 knapsack problems

		ABHS	BHS	NGHS	DBPSO			ABHS	BHS	NGHS	DBPSO
Kp11	Best	**6988**	6988	6968	6676	Kp16	Best	**113575**	112132	113512	74222
	Mean	**6959.5**	6957.2	6939.1	6495.7		Mean	**113353.2**	111832.4	113216.2	73851.4
	Std.dev	**11.6344**	11.3661	21.6904	72.5620		Std.dev	**117.7744**	134.1563	151.4718	321.6157
	t-test	/	0	1	1		t-test	/	1	1	1
Kp12	Best	**28728**	28638	28648	25232	Kp17	Best	**126495**	124636	126373	83187
	Mean	**28568.4**	28515.9	28421.3	24876.9		Mean	**126209.6**	124208.3	126136.5	82037.8
	Std.dev	**70.6192**	65.5928	82.3136	127.7782		Std.dev	**114.4706**	178.7048	157.8829	520.5435
	t-test	/	1	1	1		t-test	/	1	1	1
Kp13	Best	**64545**	64143	64433	48356	Kp18	Best	**139768**	137464	139726	91462
	Mean	**64341.8**	63943.6	64158.4	47440.9		Mean	**139415.3**	137046.5	139345.0	90543.6
	Std.dev	**117.2271**	94.3809	106.6808	416.3199		Std.dev	**159.7364**	173.2455	141.0533	432.6786
	t-test	/	1	1	1		t-test	/	1	1	1
Kp14	Best	**86909**	86150	86867	57161	Kp19	Best	**151338**	148355	151310	99715
	Mean	**86740.7**	85955.9	86677.8	56238.4		Mean	**150868.7**	147898.8	150845.2	98129.0
	Std.dev	**86.7676**	107.0587	93.9446	369.2952		Std.dev	**190.4204**	201.1472	171.9146	488.8666
	t-test	/	1	1	1		t-test	/	1	1	1
Kp15	Best	**97725**	96416	97546	65888	Kp20	Best	**166404**	162968	166240	112388
	Mean	**97364.2**	96175.5	97223.0	65368.5		Mean	**166045.4**	162455.4	165654.1	110525.9
	Std.dev	**148.4963**	130.1310	153.7824	267.2838		Std.dev	**193.9732**	222.6047	271.0752	577.0644
	t-test	/	1	1	1		t-test	/	1	1	1

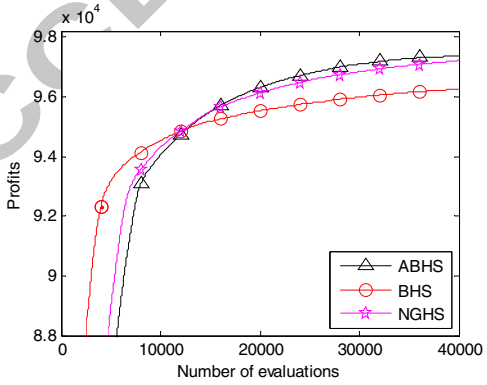


Fig.42 Convergence graph of Kp15

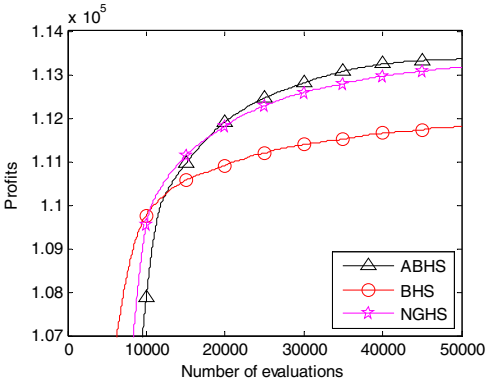


Fig.43 Convergence graph of Kp16

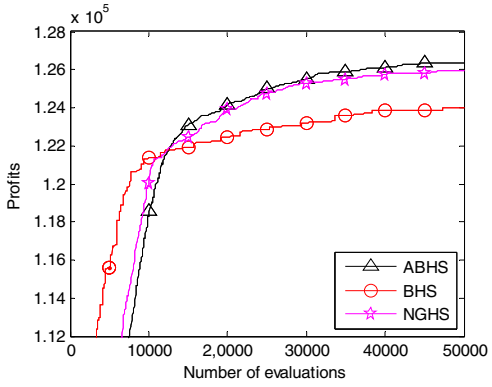


Fig.44 Convergence graph of Kp17

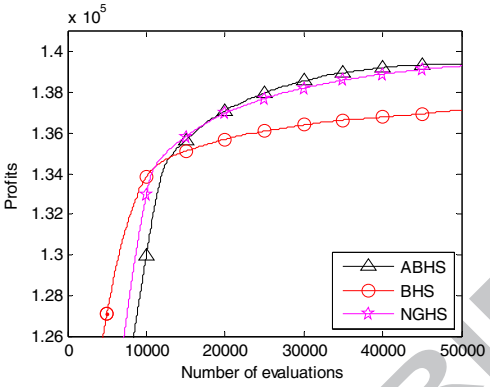


Fig.45 Convergence graph of Kp18

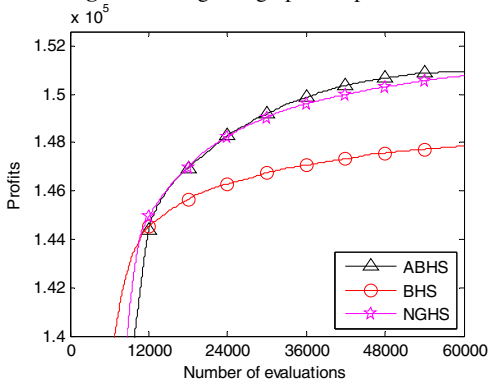


Fig.46 Convergence graph of Kp19

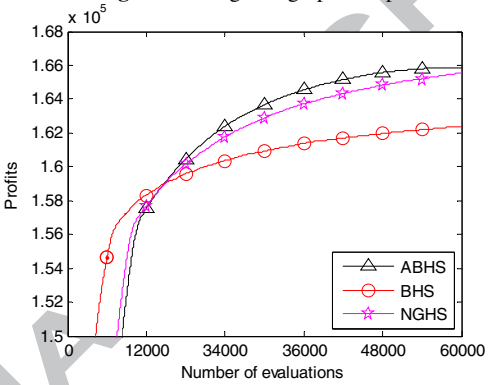


Fig.47 Convergence graph of Kp20

6 Conclusion

In this paper, an improved adaptive binary Harmony Search algorithm is proposed to extend HS to tackle binary-coded optimization problems efficiently and effectively. Based on the analysis on the drawback of the standard HS with binary-coding, a new pitch adjustment rule is used in ABHS to enhance the search ability. Two harmony memory consideration strategies (the bit selection strategy and the individual selection strategy) and two updating mechanisms (the serial updating and the parallel updating) are investigated, and a scalable adaptive strategy is developed for ABHS to improve its optimization ability and robustness based on the parameter study. Finally, ABHS was verified and compared with BHS, NGHS and DBPSO on the low-dimensional and high-dimensional functions as well as 0-1 KPs. The experimental results show that ABHS is superior to BHS, NGHS and DBPSO in

terms of the search accuracy and the convergence speed especially on the complicated optimization problems, which demonstrates that the proposed ABHS is an effective optimization tool and can be widely used in the scientific and engineering fields.

Acknowledgements

This work is supported by the Research Fund for the Doctoral Program of Higher Education of China (20103108120008), ChenGuang Plan (2008CG48), the Projects of Shanghai Science and Technology Community (10ZR1411800 & 10JC1405000), National Natural Science Foundation of China (Grant No. 60834002, 60804052 & 61074032), Shanghai University "11th Five-Year Plan" 211 Construction Project and the Mechatronics Engineering Innovation Group project from Shanghai Education Commission.

References

[1] M.T. Ayvaz, Identification of Groundwater Parameter Structure Using Harmony Search Algorithm, Studies in Computational Intelligence, 191 (2009) 129-140.
[2] M.T. Ayvaz, Application of Harmony Search algorithm to the solution of groundwater management models, Advances in Water Resources, 32 (2009) 916-924.
[3] J. Brest, S. Greiner, B. Bokovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation, 10 (2006) 646-657.
[4] Y.M. Cheng, L. Li, T. Lansivaara, S.C. Chi, Y.J. Sun, An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis, Engineering Optimization, 40 (2008) 95-115.
[5] L.d.S. Coelho, V.C. Mariani, An improved harmony search algorithm for power economic load dispatch, Energy Conversion and Management, 50 (2009) 2522-2526.
[6] 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 Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 41 (2011) 89-106.
[7] G. Fornarelli, A. Giaquinto, Adaptive particle swarm optimization for CNN associative memories design, Neurocomputing, 72 (2009) 3851-3862.
[8] R. Forsati, M. Mahdavi, M. Kangavari, B. Safarkhani, Web page clustering using Harmony Search optimization, in: Electrical and Computer Engineering, 2008, pp. 1601-1604.
[9] J. Fourie, S. Mills, R. Green, Visual Tracking Using Harmony Search, in: Z. Geem (Ed.) Recent Advances In Harmony Search Algorithm, Springer Berlin / Heidelberg, 2010, pp. 37-50.
[10] Z.W. Geem, Harmony Search Algorithm for Solving Sudoku, in: B. Apolloni, R. Howlett, L. Jain (Eds.) Knowledge-Based Intelligent Information and Engineering Systems, Springer Berlin / Heidelberg, 2007, pp. 371-378.
[11] Z.W. Geem, Harmony Search in Water Pump Switching Problem, in: L. Wang, K. Chen, Y. Ong (Eds.) Advances in Natural Computation, Springer Berlin / Heidelberg, 2005, pp. 445-445.

[12] Z.W. Geem, Optimal scheduling of multiple dam system using harmony search algorithm, in: Proceedings of the 9th international work conference on Artificial neural networks, Springer-Verlag, San Sebasti, Spain, 2007, pp. 316-323.
[13] Z.W. Geem, J. Kim, G. Loganathan, Music-Inspired Optimization Algorithm Harmony Search, Simulation, 76 (2001) 60-68.
[14] Z.W. Geem, C.L. Tseng, New methodology, harmony search and its robustness, in: 2002 Genetic and Evolutionary Computation Conference, 2002, pp. 174-178.
[15] Z.W. Geem, J.C. Williams, Ecological optimization using harmony search, in: Proceedings of the American Conference on Applied Mathematics, World Scientific and Engineering Academy and Society (WSEAS), Cambridge, Massachusetts, 2008, pp. 148-152.
[16] M. Gong, L. Jiao, X. Zhang, A population-based artificial immune system for numerical optimization, Neurocomputing, 72 (2008) 149-161.
[17] J. Greblicki, J. Kotowski, Analysis of the Properties of the Harmony Search Algorithm Carried Out on the One Dimensional Binary Knapsack Problem, in: 12th International Conference on Computer Aided Systems Theory, EUROCAST 2009, Springer Verlag, 2009, p. 697-704.
[18] O. Hasanebi, F. Erdal, M.P. Saka, Adaptive harmony search method for structural optimization, Journal of Structural Engineering, 136 (2010) 419-431.
[19] J. Kennedy, R. Eberhurt, A discrete binary version of the particle swarm algorithm, in: Proc. 1997 Conf. Systems, Man, Cybernetics,Piscataway, NJ, 1997, p.4107-4108.
[20] K. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Computer Methods in Applied Mechanics and Engineering, 194 (2005) 3902-3933.
[21] H.Q. Li, L. Li, A Novel Hybrid Particle Swarm Optimization Algorithm Combined with Harmony Search for High Dimensional Optimization Problems, in: 2007 International Conference on Intelligent Pervasive Computing, 2007, pp. 94-97.
[22] H.Q, Li., L. Li, A Novel Hybrid Real-Valued Genetic Algorithm for Optimization Problems, in: Computational Intelligence and Security, 2007, pp. 91-95.
[23] L.P. Li, L. Wang, Hybrid algorithms based on harmony search and differential evolution for global optimization, in: 2009 World Summit on Genetic and Evolutionary Computation, Association for Computing Machinery, Shanghai, China, 2009, pp. 271-278
[24] L. Li, Z. Huang, F. Liu, Q. Wu, A heuristic particle swarm optimizer for optimization of pin connected structures, Computers & Structures, 85 (2007) 340-349.
[25] L. Li, G. Yu, S. Lu, G. Wang, X. Chu, An improved harmony search algorithm for the location of critical slip surfaces in slope stability analysis, in: 5th International Conference on Intelligent Computing, Springer-Verlag, 2009, pp. 215-222.
[26] M. Mahdavi, M. Chehreghani, H. Abolhassani, R. Forsati, Novel meta-heuristic algorithms for clustering web documents, Applied Mathematics and Computation, 201 (2008) 441-451.
[27] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation, 188 (2007) 1567-1579.
[28] I. Montalvo, J. Izquierdo, R. Pérez-García, M. Herrera, Improved performance of PSO with self-adaptive parameters for computing the optimal design of Water Supply Systems, Engineering Applications of Artificial Intelligence, 23 (2010) 727-735.
[29] S. Mun, Z.W. Geem, Determination of individual sound power levels of noise sources using a harmony search algorithm, International Journal of Industrial Ergonomics, 39 (2009) 366-370.
[30] S. Ngonkham, P. Buasri, Harmony search algorithm to improve cost reduction in power generation system integrating large scale wind energy conversion system, in: World Non-Grid-Connected Wind Power and Energy Conference, 2009, pp. 1-5.
[31] M. Omran, M. Mahdavi, Global-best harmony search, Applied Mathematics and Computation, 198 (2008) 643-656.
[32] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGAL Report, 2005005 (2005).
[33] Q.K. Pan, P.N. Suganthan, M.F. Tasgetiren, A Harmony Search Algorithm with Ensemble of Parameter Sets, in: Evolutionary Computation, 2009. CEC '09, 2009, pp. 1815-1820
[34] Q.K. Pan, P.N. Suganthan, J.J. Liang, M.F. Tasgetiren, A local-best harmony search algorithm with dynamic subpopulations. Engineering Optimization, 42(2)(2009): p. 101-117.

[35] Q.K. Pan, P.N. Suganthan, M.F. Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous optimization problems. Applied Mathematics and Computation, 216(3) (2010) 830-848.

[36] P.M. Pardalos, M. Resende (Eds.), Handbook of Applied Optimization. Oxford University Press, Oxford, 2002.

[37] P. M. Pardalos, E. Romeijn (Eds.), Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1995.

[38] M. Saka, Optimum design of steel sway frames to BS5950 using harmony search algorithm, Journal of Constructional Steel Research, 65 (2009) 36-43.

[39] M. Saka, Optimum geometry design of geodesic domes using harmony search algorithm, Advances in Structural Engineering, 10 (2007) 595-606.

[40] H. Sarvari, K. Zamanifar, A Self-Adaptive Harmony Search Algorithm for Engineering and Reliability Problems, in: Computational Intelligence, Modelling and Simulation (CIMSiM), 2010, pp. 59-64.

[41] J.W. Seok, K.H. Il, L.B. Hee, Hybrid Simplex-Harmony search method for optimization problems, in: 2008 IEEE Congress on Evolutionary Computation, Hong Kong, China, 2008, pp. 4157-4164.

[42] N. Taherinejad, Highly reliable harmony search algorithm, in: European Conference on Circuit Theory and Design Conference Program, IEEE Computer Society, 2009, pp. 818-822.

[43] C. Wang, Y. Huang, Self-adaptive harmony search algorithm for optimization, Expert Systems with Applications, 37 (2010) 2826-2837.

[44] L. Wang, Y.F. Mao, Q. Niu, M.R. Fei. A Multi-Objective Binary Harmony Search Algorithm. in: 2nd International Conference on Swarm Intelligence, Chongqing, China, 2011, p. 74-81.

[45] L. Wang, X. Wang, J. Fu, L. Zhen, A novel probability binary particle swarm optimization algorithm and its application, Journal of software, 3 (2008) 28-35.

[46] L. Wang, X.T. Wang, J.S. Yu, Naphtha cracking furnace fault diagnosis based on adaptive quantum ant colony algorithm, CIESC, 60 (2009) 401-408.

[47] L. Wang, Y. Xu, Y. Mao, M. Fei, A Discrete Harmony Search Algorithm, in: Int. Conf. on Life System Modeling and Simulation, LSMS 2010 and Int. Conf. on Intelligent Computing for Sustainable Energy and Environment, 2010, pp. 37-43.

[48] X. Wang, X.Z. Gao, S.J. Ovaska, A Hybrid Optimization Method for Fuzzy Classification Systems, in: Hybrid Intelligent Systems, HIS '08, 2008, pp. 264-271.

[49] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Information Sciences, 181 (2011) 4515-4538.

[50] S.Z. Zhao, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with harmony search. Expert Systems with Applications, 38(4) (2011) 3735-3742.

[51] D. Zou, L. Gao, S. Li, J. Wu, Solving 0-1 knapsack problem by a novel global harmony search algorithm, Applied Soft Computing, 11 (2011) 1556-1564.