Applied Soft Computing 13 (2013) 1781–1791

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Artificial bee colony algorithm and pattern search hybridized for global optimization

Fei Kang*, Junjie Li, Haojin Li

Faculty of Infrastructure Engineering, Dalian University of Technology, Dalian 116024, China

ARTICLE INFO

Article history:
Received 19 July 2012
Received in revised form
29 November 2012
Accepted 31 December 2012
Available online 10 January 2013

Keywords:
Artificial bee colony algorithm
Swarm intelligence
Memetic algorithm
Evolutionary computation
Global optimization

ABSTRACT

Artificial bee colony algorithm is one of the most recently proposed swarm intelligence based optimization algorithm. A memetic algorithm which combines Hooke–Jeeves pattern search with artificial bee colony algorithm is proposed for numerical global optimization. There are two alternative phases of the proposed algorithm: the exploration phase realized by artificial bee colony algorithm and the exploitation phase completed by pattern search. The proposed algorithm was tested on a comprehensive set of benchmark functions, encompassing a wide range of dimensionality. Results show that the new algorithm is promising in terms of convergence speed, solution accuracy and success rate. The performance of artificial bee colony algorithm is much improved by introducing a pattern search method, especially in handling functions having narrow curving valley, functions with high eccentric ellipse and some complex multimodal functions.

1. Introduction

Global optimization could be a very challenging task because many objective functions and real world problems are multimodal, highly non-linear, with steep and flat regions and irregularities [1], thus better optimization algorithms are always needed. Unconstrained global optimization problems can be formulated as

$$\min f(\boldsymbol{x}), \boldsymbol{x} = (x_1, x_2, \ldots, x_n) \quad (1)$$

where f: $\mathbf{R}^n \rightarrow \mathbf{R}$ is a real-valued objective function, $\boldsymbol{x} \in \mathbf{R}^n$, and n is the number of the parameters to be optimized.

With more complex systems arising in science and engineering fields often remained intractable to conventional mathematical and analytical methods, the research community has diverted their attention toward soft computing techniques to deal with various complex problems [2–4]. Evolutionary algorithms such as genetic algorithms [5], differential evolution [6] and particle swarm optimization (PSO) [7] are powerful tools for solving complex optimization problems. As extensions of evolutionary algorithms, memetic algorithms [8,9] have been developed to combine the advantage of evolutionary algorithms and some local search strategies. Such hybrids have been successfully applied to global optimization of numerical functions [5,10] and have been used to solve numerous real-world optimization problems [11,12].

Studies on swarm intelligence of honeybees for optimization problems are currently prevalent [13]. Artificial bee colony (ABC) algorithm is one of the newest global optimization techniques proposed by Karaboga and Basturk [14], inspired by the foraging behavior of honeybee swarms. It has received increasing interest from the optimization community due to its simplicity, wide applicability and outstanding performance. ABC methods that use chaotic maps as efficient alternatives to pseudorandom sequences were proposed by Alatas [15], and global-best-solution-guided ABC (GABC) was proposed by Zhu and Kwong [16]. Banharnsakun et al. [17] proposed a best-so-far selection ABC for numerical optimization and image registration. Gao and Liu [18] proposed an improved ABC algorithm inspired by differential evolution. Li et al. [19] proposed an ABC algorithm with the abilities of prediction and selection. The performance of ABC has already been compared with other optimization methods, such as GA, DE, and PSO [20,21]. The comparisons were made based on various numerical benchmark functions, which consist of unimodal and multimodal distributions. Results show that ABC can produce more optimal solutions and thus is more effective than the other methods in several sorts of engineering problems such as signal processing [22,23], parameter identification [24,25], clustering [26,27], image segmentation [28,29], leak detection [30], structure optimization [31] and geotechnical stability problems [32,33]. More extensive review of ABC can be seen in [34].

ABC has already been shown to be a promising global optimization algorithm. However, it still has some limitations in handling certain optimization problems [21]. Meanwhile, like

* Corresponding author. Tel.: +86 0411 84708516; fax: +86 0411 84708501.
E-mail addresses: kangfei2009@163.com, kangfei@dlut.edu.cn (F. Kang).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.12.025

most population-based algorithms, ABC takes a long time because of its stochastic nature. To improve the performance of ABC, a Hooke–Jeeves artificial bee colony (HABC) algorithm is proposed for global optimization of numerical functions. Compared to the algorithm proposed in [35], the HABC algorithm has only two more control parameters than ABC. The proposed algorithm retains the main steps of ABC and incorporates a pattern search based local search technique. ABC and pattern search have complementary advantages, and a hybrid of the two algorithms can result in a faster and more robust technique. The effect of control parameters for HABC was studied and efficiency of the new algorithm was tested on extensive functions.

The remainder of this paper is organized as follows. In Section 2, a concise presentation of ABC is provided. In Section 3, the proposed HABC algorithm is introduced. In Section 4, experimental results are presented. Conclusions are given in Section 5. Some test problems are listed in Appendix A.

2. Artificial bee colony algorithm

ABC is a population based optimization algorithm inspired by the honeybee foraging behavior. In ABC, there are three types of honeybees: employed bees, onlookers and scouts. The position of a food source represents a possible solution to the optimization problem and the profitability of a food source corresponds to the quality (fitness) of the associated solution. The number of employed bees is equal to the number of onlookers, and also equal to the number of food sources. Any food sources cannot be improved further in definite cycles will be replaced with a new food source by a scout bee.

At the beginning, an initial population contains NS solutions are generated randomly. Where $NS = NP/2$ is the number of food sources, and it is equal to the number of employed bees. NP is the population size. Each solution $\boldsymbol{x}_i$ ($i = 1, 2, \ldots, NS$) is an n-dimensional vector. Then, the honeybees perform cyclic search according to some specific rules.

To update feasible solutions, each employed bee selects a new candidate food source position. The choice is based on the neighborhood of the previously selected food source. A candidate solution $\boldsymbol{v}_i$ can be generated from the old solution $\boldsymbol{x}_i$ as

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{2}$$

where $k \in \{1, 2, \ldots, NS\}$ and $j \in \{1, 2, \ldots, n\}$ are indexes chosen randomly; k has to be different from i; ϕ_{ij} is a uniformly distributed random number in the range $[-1, 1]$.

The candidate solution will be compared with the old one. If the new food source has equal or better quality than the previous source, the old one is replaced by the new one. Otherwise, the old one is retained. Employed bees will return to their hive and share the information of the food sources they have found with the onlooker bees.

In the next step, each onlooker bee selects one of the food sources depending on the fitness value. The probability p_i of a food source will be selected by an onlooker bee can be calculated as

$$p_i = \frac{fit_i}{\sum_{j=1}^{NS} fit_j}, \tag{3}$$

where fit_i is the fitness value of food source i, which is related to the objective function value of the food source.

The probability of a food source being selected by the onlooker bees increases as the fitness value of the food source increases. After the food source is selected, each onlooker bee finds a new candidate food source in the neighborhood of the selected one. The candidate food source can be calculated by Eq. (2). By evaluating the fitness of the candidate food source, the new food source is determined by greedy selection.

If a position cannot be improved further through *limit* cycles, then that food source is assumed to be abandoned. The corresponding employed bee becomes a scout, and the food source will be replaced with a new one found by the scout. *limit* is a predetermined number by the users. If the abandoned source is $\boldsymbol{x}_i$, the scout discovers a new food source as follows:

$$x_{ij} = x_{j\min} + rand[0, 1](x_{j\max} - x_{j\min}), \tag{4}$$

where $x_{j\min}$ and $x_{j\max}$ are lower and upper bounds of variable x_{ij}, $rand[0,1]$ is a uniformly distributed random number in the range $[0,1]$.

The process will be repeated until the output of the objective function reaches a defined threshold value or the number of iteration equals a predefined maximum number of cycles. The main steps of ABC are shown in Fig. 1.

3. Hooke–Jeeves artificial bee colony algorithm

3.1. The modified Hooke–Jeeves method

Hooke–Jeeves pattern search method is a simple yet very effective optimization technique proposed by Hooke and Jeeves [36]. Today, it is still a popular tool for various optimization problems, especially for deterministic local search.

In the pattern search method, a combination of exploratory move and pattern move is made iteratively to search out the optimum solution for the problem. It starts with an exploratory move to determine an appropriate direction of search by considering one variable at a time along the individual coordinate directions in the neighborhood of a base point solution. Following the exploratory search, a pattern move is made to accelerate the search in the direction determined in the exploratory search. Exploratory searches and pattern moves are repeated until a termination criterion is met, as illustrated in Fig. 2.

The basic pattern search method is modified to accommodate the hybrid strategy. The main characteristics of the modified method are: (1) to accelerate the procedure, direct search takes advantage of its knowledge of the sign of its previous move in each of the directions; (2) different step sizes for different dimensions are used to adaptive to the scaling problems of different variables.

Assume $\boldsymbol{x}_0$ is the current solution (the base point), $f_{\min}$ is the current minimum value of the objective function, $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_n)$ are the step sizes of n directions. $\boldsymbol{x}_1$ is a temporary vector to store the obtained point after exploratory move. The main steps of exploratory move are described in Fig. 3. Given two solutions $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ ($f(\boldsymbol{x}_1) < f(\boldsymbol{x}_0)$), the pattern move takes the step $\boldsymbol{x}_1 - \boldsymbol{x}_0$ from $\boldsymbol{x}_0$ as

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + (\boldsymbol{x}_1 - \boldsymbol{x}_0), \tag{5}$$

where $\boldsymbol{x}_2$ is the point obtained by pattern move. The pattern move is an aggressive attempt of the algorithm to exploit promising search directions because it exploits information gained from the search during previous successful iterations. The idea of pattern move is to investigate whether further progress is possible in the general direction $\boldsymbol{x}_1 - \boldsymbol{x}_0$ (since, if $f(\boldsymbol{x}_1) < f(\boldsymbol{x}_0)$, then $\boldsymbol{x}_1 - \boldsymbol{x}_0$ is clearly a promising direction) [37]. The main steps of the modified pattern search method are shown in Fig. 4. The variable s_a is adopted to judge when to stop the algorithm. The usually adopted value of step size reduction factor is $\rho = 0.5$.

3.2. The proposed hybrid artificial bee colony algorithm

The slightly modified pattern search is incorporated into ABC as a local exploitation tool. Meanwhile, to maintain the colony

1: Initialize the solutions $\boldsymbol{x}_i$, i=1, …, NS, and evaluate them. Set $cycle$ = 1.
2: **Repeat**
3: Generate candidate solutions $\boldsymbol{v}_i$ for employed bees by (2) and evaluate them.
4: Determine the new solutions of employed bees by greedy selection.
5: Calculate the selection probability values P_i for solutions $\boldsymbol{x}_i$ by (3).
6: Generate candidate solutions $\boldsymbol{v}_i$ for the onlookers from the solutions selected and evaluate them.
7: Determine the new solutions for onlookers by greedy selection.
8: If there exist an abandoned solution for the scout, replace it with a randomly produced solution $\boldsymbol{x}_i$ by (4).
9: Memorize the best solution achieved so far.
10: $cycle = cycle + 1$.
11: **Until** a termination condition is met.

Fig. 1. Main steps of ABC algorithm.

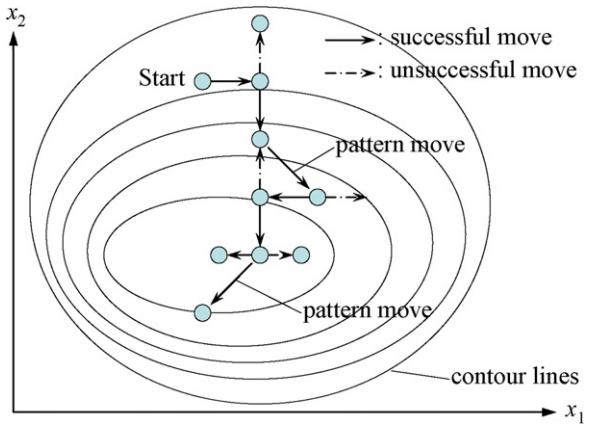


Fig. 2. A search guided by Hooke–Jeeves pattern search.

1: Initialize $\boldsymbol{x}_1 = \boldsymbol{x}_0$, $f_{\min} = f(\boldsymbol{x}_0)$, i = 1.
2: Set $\boldsymbol{x}_{1i} = \boldsymbol{x}_{0i} + \delta_i$, if ($f(\boldsymbol{x}_1) < f_{\min}$), $f_{\min} = f(\boldsymbol{x}_1)$, go to step 4; else go to step 3.
3: Set $\boldsymbol{x}_{1i} = \boldsymbol{x}_{0i} - \delta_i$, if ($f(\boldsymbol{x}_1) < f_{\min}$), $f_{\min} = f(\boldsymbol{x}_1)$, go to step 4; else $\boldsymbol{x}_{1i} = \boldsymbol{x}_{0i}$.
4: If $i < n$, set $i = i + 1$ and go to step 2; else go to step 5.
5: If $f_{\min} < f(\boldsymbol{x}_0)$, the exploratory move is successful; else it is failing.

Fig. 3. Main steps of the exploratory move.

diversity and avoid premature convergence on some particular problems, a rank-based fitness transformation [21] is adopted as

$$fit_i = 2 - SP + \frac{2(SP-1)(r_i-1)}{NS-1}, \tag{6}$$

where r_i is the position of solution i in the entire population after ranking, $SP \in [1.0, 2.0]$ is the selection pressure, and a medium value of SP = 1.5 is appropriate.

The main steps of the hybrid algorithm are summarized as follows. Every n_c cycles of ABC, the modified Hooke–Jeeves pattern search algorithm is activated to perform a local search using the current best solution as the base point. The step size $\boldsymbol{\delta}$ should be suitable to the current states of solutions, so an adaptive step size is adopted. It is set as a fraction of the average of distance between the selected solutions and the best solution achieved so far. The first 10% solutions after ranking are selected to calculate the step size as

$$\delta_j = 0.1\frac{\sum_{i=1}^{m}(x'_{ij} - x_{best,j})}{m}, \tag{7}$$

where δ_j is the step size of the jth dimension, $m = NS \times 10\%$ is the number of solutions selected to calculate the step size, x'_i is the ith solution after ranking, x_{best} is the current best solution. At early stages, the population will be diverse and this will result in larger δ_j. As the population converges, the distance between different solutions decreases and so does the step size of the pattern search method. The iteration times of pattern search are controlled by the parameter ε_s, when $s_a < \varepsilon_s$ the algorithm will return to the main framework of HABC.

The new algorithm conducts the optimization process in two phases alternately: during the exploration phase it employs the ABC algorithm to locate regions of attraction; and subsequently, during the exploitation phase, employs the adaptive pattern search technique to make a local exploitation search near the best solution. This process is repeated until the termination condition is met, e.g., the maximum number of function evaluations ($NFE_{\max}$) has reached. For the sake of clarity, the main steps of HABC are outlined in Fig. 5.

4. Experimental results

4.1. Test suite and experimental setup

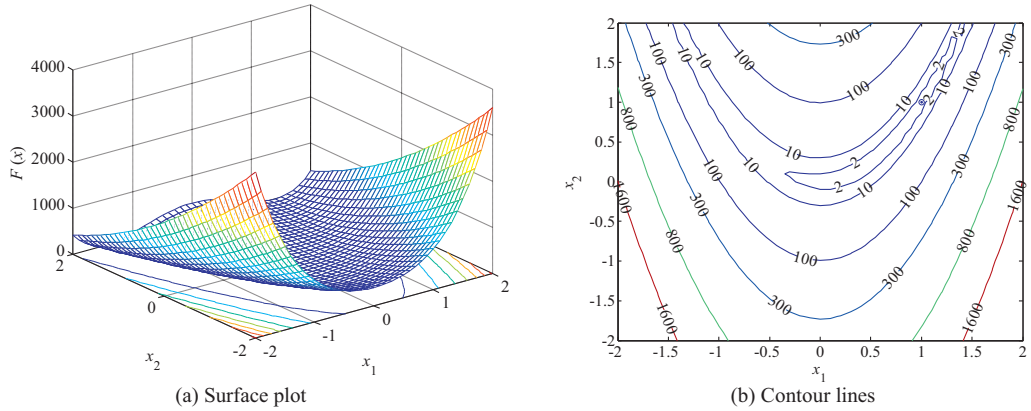
A comprehensive set of benchmark functions collected from several references [6,20,21,38], was adopted for performance verification of the proposed approach. This set is large enough to include many kinds of problems such as unimodal, multimodal, regular, irregular, separable, non-separable. Some of these functions are listed in Appendix A. The other functions can be seen in the work of Kang et al. [21]. Typical functions are illustrated in Figs. 6–8. The global optimum of Rosenbrock function lies inside a long, narrow, parabolic shaped flat valley. It is difficult to convergence to the global optimum for evolutionary algorithms. Zakharov and Schwefel Ridge are functions with high eccentric ellipse. The functions of Fletcher–Powell, Shekel Foxholes, and Modified Langerman are highly multimodal. Meanwhile, they are non-symmetrical and their local optima are randomly distributed.

1: Initialize the starting point $\boldsymbol{x}_0$, the step size δ_i (i = 1, 2, …, n), the step size reduction factor $\rho < 1$, the termination parameter $\varepsilon_s > 0$, the iteration counter k = 1, the variable s_a = 1.0.
2: Perform an exploratory move with $\boldsymbol{x}_0$ as the base point and the obtained point is $\boldsymbol{x}_1$. If the exploratory move is successful, go to step 3; else go to step 7.
3: If ($x_{1i} < x_{0i}$) $\delta_i = 0 - |\delta_i|$; else $\delta_i = |\delta_i|$, for i = 1, 2, …, n.
4: Perform a pattern move $\boldsymbol{x}_2 = \boldsymbol{x}_1 + (\boldsymbol{x}_1 - \boldsymbol{x}_0)$ and set $\boldsymbol{x}_0 = \boldsymbol{x}_1$.
5: Perform exploratory move with $\boldsymbol{x}_2$ as the base point and the obtained point is $\boldsymbol{x}_1$.
6: If $f(\boldsymbol{x}_1) < f(\boldsymbol{x}_0)$, go to step 3, else go to step 7.
7: If $s_a < \varepsilon_s$, terminate; else set $k = k + 1$, $s_a = s_a \times \rho$, $\delta_i = \delta_i \times \rho$, for i = 1, 2, …, n, and go to step 2.

Fig. 4. Main steps of the modified pattern search.

1: Initialize the population of solutions $\boldsymbol{x}_i$, $i = 1, \ldots, NS$, $cycle = 1$.
2: Evaluate the population and memorize the best solution $\boldsymbol{x}_{best}$.
3: **Repeat**
(*Exploration phase*)
4: Generate candidate solutions $\boldsymbol{v}_i$ for the employed bees by (2) and evaluate them.
5: Apply the greedy selection process for the employed bees.
6: Rank the population and calculate the fitness by (6).
7: Calculate the selection probability p_i for solutions $\boldsymbol{x}_i$ by (3).
8: Generate candidate solutions $\boldsymbol{v}_i$ for the onlookers from the solutions selected depending on p_i and evaluate them.
9: Apply the greedy selection process for the onlookers.
10: If there is an abandoned solution for the scout, replace it with a new randomly produced solution $\boldsymbol{x}_i$ by (4).
11: Memorize the best solution $\boldsymbol{x}_{best}$ achieved so far.
(*Exploitation phase*)
12: If (($cycle$ mod n_c) = 0), calculate step size δ_i for pattern search according to (7).
13: Call the modified pattern search with $\boldsymbol{x}_{best}$ as the base point until $s_a < \varepsilon_s$ and the obtained point is $\boldsymbol{x}_{best1}$.
14: If ($f(\boldsymbol{x}_{best1}) \le f(\boldsymbol{x}_{best})$) Replace the solution in the middle position after ranking by $\boldsymbol{x}_{best1}$ and set $\boldsymbol{x}_{best} = \boldsymbol{x}_{best1}$.
15: Set $cycle = cycle+1$.
16: **Until** a termination condition is met.

Fig. 5. Main steps of the proposed HABC algorithm.



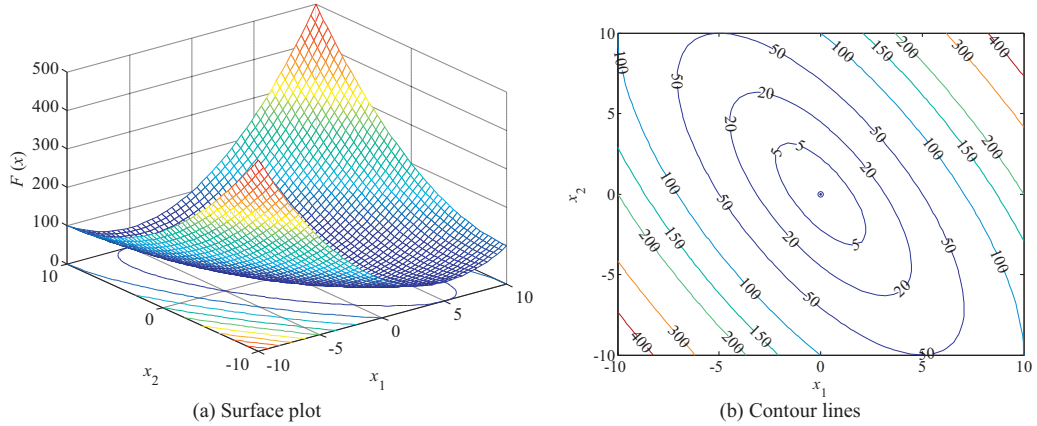
(a) Surface plot (b) Contour lines

Fig. 6. Rosenbrock function, $n = 2$.

The common parameters are population size, *limit* and NFE_{max}. The population size was set as 50, and the parameter *limit* is defined relate to the dimension of the problem and the colony size as $limit = NS \times n$ [20,39]. Two new control parameters n_c and ε_s were used in HABC. The parameter n_c was defined relate to the dimension of the problem as $n_c = k \times n$, $k = 1, 2, \ldots$, and each experiment was repeated 50 times with different random seeds.

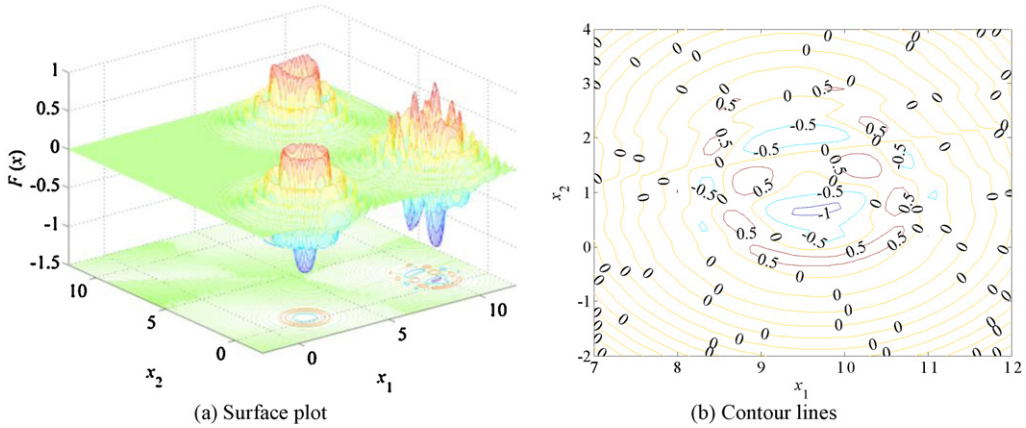
4.2. Effect of parameters on HABC

The effect of parameters on HABC was tested on 10 typical functions. The maximum number of function evaluations was set as 200,000 in the test. The mean best function values (mean) and the standard deviations (SD) of HABC for different values of ε_s and n_c are listed in Tables 1 and 2, respectively. The results reveal that a proper combination of parameter values can improve the performance of



(a) Surface plot (b) Contour lines

Fig. 7. Schwefel Ridge, $n = 2$.



(a) Surface plot (b) Contour lines

Fig. 8. Modified Langerman function, $n = 2$.

Table 1
Results of HABC with different values of parameter ε_s, $n_c = 5n$.

F	n	ε_s = 1e−1 mean(SD)	ε_s = 1e−3 mean(SD)	ε_s = 1e−5 mean(SD)	ε_s = 1e−7 mean(SD)	ε_s = 1e−9 mean(SD)
SP	30	1.77e−63 (2.30e−63)	6.36e−66 (1.90e−65)	8.25e−70 (4.51e−69)	3.93e−65 (2.15e−64)	1.11e−61 (4.79e−61)
CO	4	1.94e−05 (4.99e−05)	6.68e−08 (1.30e−07)	8.24e−12 (2.29e−11)	8.00e−16 (1.24e−15)	1.22e−02 (6.31e−02)
RO	30	2.86e−01 (9.53e−01)	2.50e−09 (1.35e−08)	3.73e−06 (1.75e−05)	4.71e+01(1.12e+02)	1.33e+01 (2.34e+01)
SR	30	5.66e−08 (1.52e−07)	4.40e−18 (1.58e−17)	2.09e−11 (7.36e−11)	6.19e−02 (2.59e−01)	3.86e+00 (1.74e+01)
ZA	30	4.76e−03 (4.84e−03)	1.05e−10 (1.72e−10)	2.74e−08 (1.50e−07)	6.21e−06 (3.39e−05)	9.12e−04 (4.44e−03)
AC	30	3.58e−14 (3.36e−15)	3.55e−14 (4.18e−15)	3.62e−14 (4.71e−15)	2.04e−01 (1.11e+0)	9.13e−01 (2.02e+00)
GR	30	9.84e−04 (4.22e−03)	2.46e−04 (1.35e−03)	2.47e−04 (1.35e−03)	4.09e−03 (1.09e−02)	8.67e−03 (1.92e−02)
RA	30	0 (0)	0 (0)	0 (0)	4.70 (8.51)	9.17 (9.97)
FP	10	7.62e+00 (1.53e+01)	3.73e+00 (6.79e+00)	4.02e+00 (7.12e+00)	4.49e+00 (7.24e+00)	1.98e+01 (5.07e+01)
MS	10	−8.08 (3.31)	−7.96 (3.51)	−7.73 (3.50)	−7.60 (3.50)	−7.02 (3.72)

HABC and the quality of solutions. In the ranges, ε_s = 1e−3 to 1e−5, and $n_c = 3n$–$7n$, satisfactory results can be obtained. The parameter values ε_s = 1e−3 and $n_c = 5n$ were adopted in the subsequent study.

4.3. Comparison between ABC and HABC

4.3.1. Fixed-iteration results

HABC was compared with ABC on 48 problems listed in Table 3. The total number of function evaluations was fixed to 300,000. The mean solutions, the standard deviations, and the results of nonparametric Wilcoxon signed-rank test [21] of 50 independent runs are also listed in Table 3. The best result for each function is highlighted in boldface.

HABC performs better than ABC for 33 functions, whereas ABC performs better than HABC only for two functions. The Wilcoxon signed-rank test results show that HABC performed better than ABC in the case of 31 problems, whereas from a statistical viewpoint, ABC performed better than HABC merely in the case of two problems. Some sample graphs for comparing the performances of ABC and HABC are shown in Fig. 9. These graphs show how HABC converged toward the optimal solution faster. It can be concluded that HABC is more efficient than ABC, and the ultimate solution of HABC is better than that of ABC in most cases.

4.3.2. Robustness analysis

The $NFE_{\max}$ was set as 300 000 for the test. A trial is considered to be successful if the following inequality holds:

$$|f_O - f_A| < \varepsilon_{rel}|f_A| + \varepsilon_{abs}, \tag{8}$$

where f_A is the analytical global minimum, f_O is the best value obtained by the algorithm, and the accuracy controlling parameters are set as $\varepsilon_{rel} = 10^{-4}$ and $\varepsilon_{abs} = 10^{-6}$ [5].

To compare the convergence speed, we use the acceleration rate (AR). It is defined based on the number of function evaluations (NFEs) for the two algorithms ABC and HABC.

$$AR = \frac{NFE_{ABC}}{NFE_{HABC}} \tag{9}$$

We compared the proposed algorithm with ABC in terms of convergence speed and robustness, and the results of 48 benchmark

Table 2
Results of HABC with different values of parameter n_c, ε_s = 1e−3.

F	n	$n_c = n$ mean(SD)	$n_c = 3n$ mean(SD)	$n_c = 5n$ mean(SD)	$n_c = 7n$ mean(SD)	$n_c = 9n$ mean(SD)
SP	30	4.10e−21 (2.24e−20)	1.11e−70 (6.06e−70)	6.36e−66 (1.90e−65)	6.65e−64 (2.25e−63)	1.05e−63 (3.05e−63)
CO	4	2.56e−08 (1.27e−07)	1.43e−08 (4.37e−07)	6.68e−08 (1.30e−07)	3.70e−07 (8.39e−07)	4.15e−07 (8.28e−07)
RO	30	2.39e−01 (8.72e−01)	6.99e−07 (3.64e−06)	2.50e−09 (1.35e−08)	9.97e−10 (4.70e−09)	1.52e−10 (8.12e−10)
SR	30	8.83e−10 (4.83e−10)	3.85e−16 (2.11e−15)	4.40e−18 (1.58e−17)	4.02e−08 (2.20e−07)	6.15e−09 (1.20e−08)
ZA	30	2.42e−20 (9.44e−20)	1.12e−14 (1.37e−14)	8.27e−09 (2.05e−08)	9.64e−09 (1.51e−08)	5.05e−07 (6.22e−07)
AC	30	7.12e−14 (9.68e−14)	3.68e−14 (7.37e−15)	3.55e−14 (4.18e−15)	3.42e−14 (4.04e−15)	3.42e−14 (3.58e−15)
GR	30	5.98e−03 (1.00e−02)	2.47e−04 (1.35e−03)	2.46e−04 (1.35e−03)	2.46e−04 (1.35e−03)	2.46e−04 (1.74e−03)
RA	30	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
FP	10	2.47e+01 (4.89e+01)	1.34e+01 (2.40e+01)	3.73e+00 (6.79e+00)	4.71e+00 (1.51e+01)	3.67e+00 (5.78e+00)
MS	10	−5.93 (3.83)	−7.50 (3.63)	−7.96 (3.51)	−8.02 (3.41)	−7.56 (3.54)

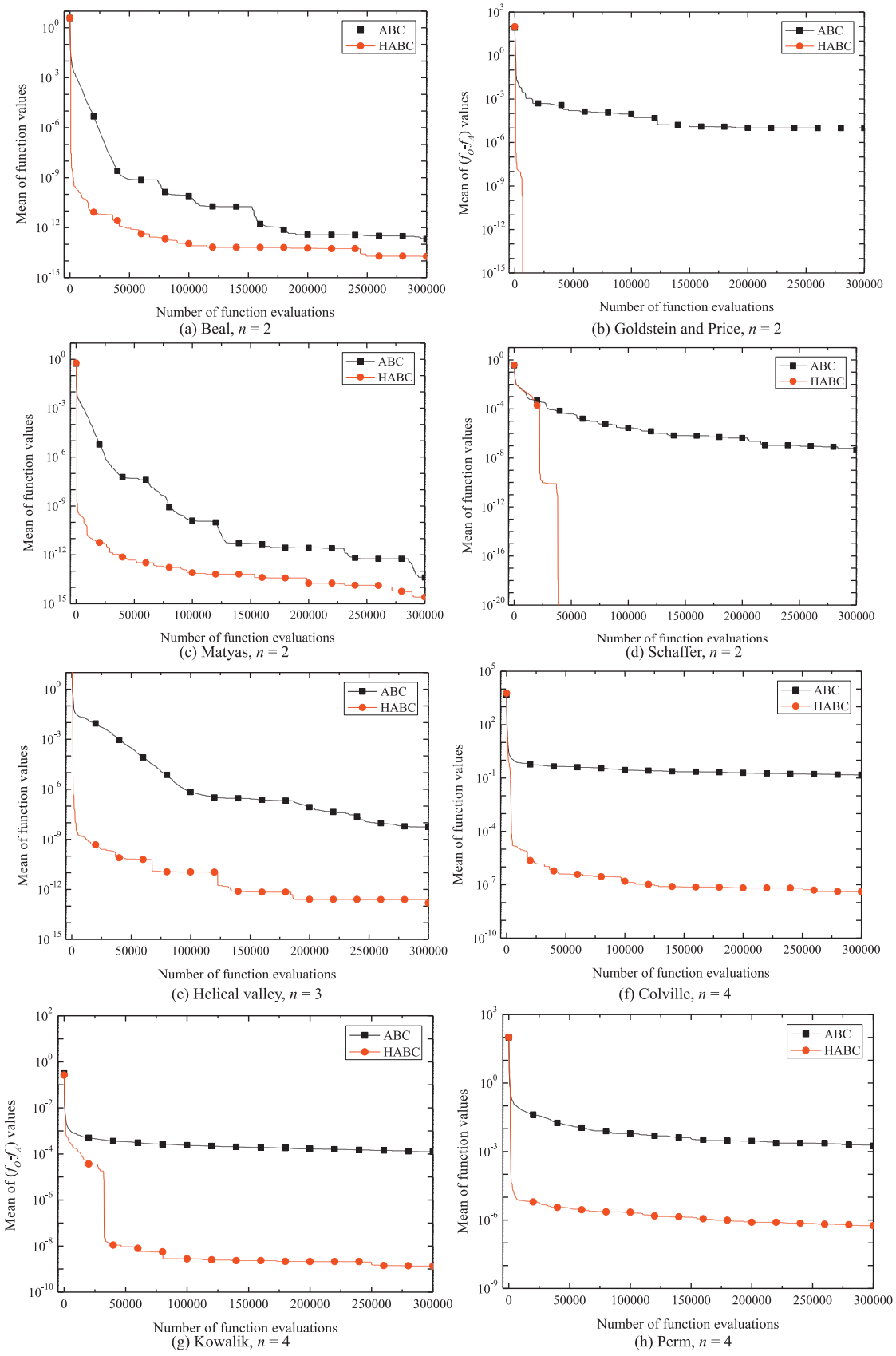


Fig. 9. Sample graphs for convergence process comparison between ABC and HABC.

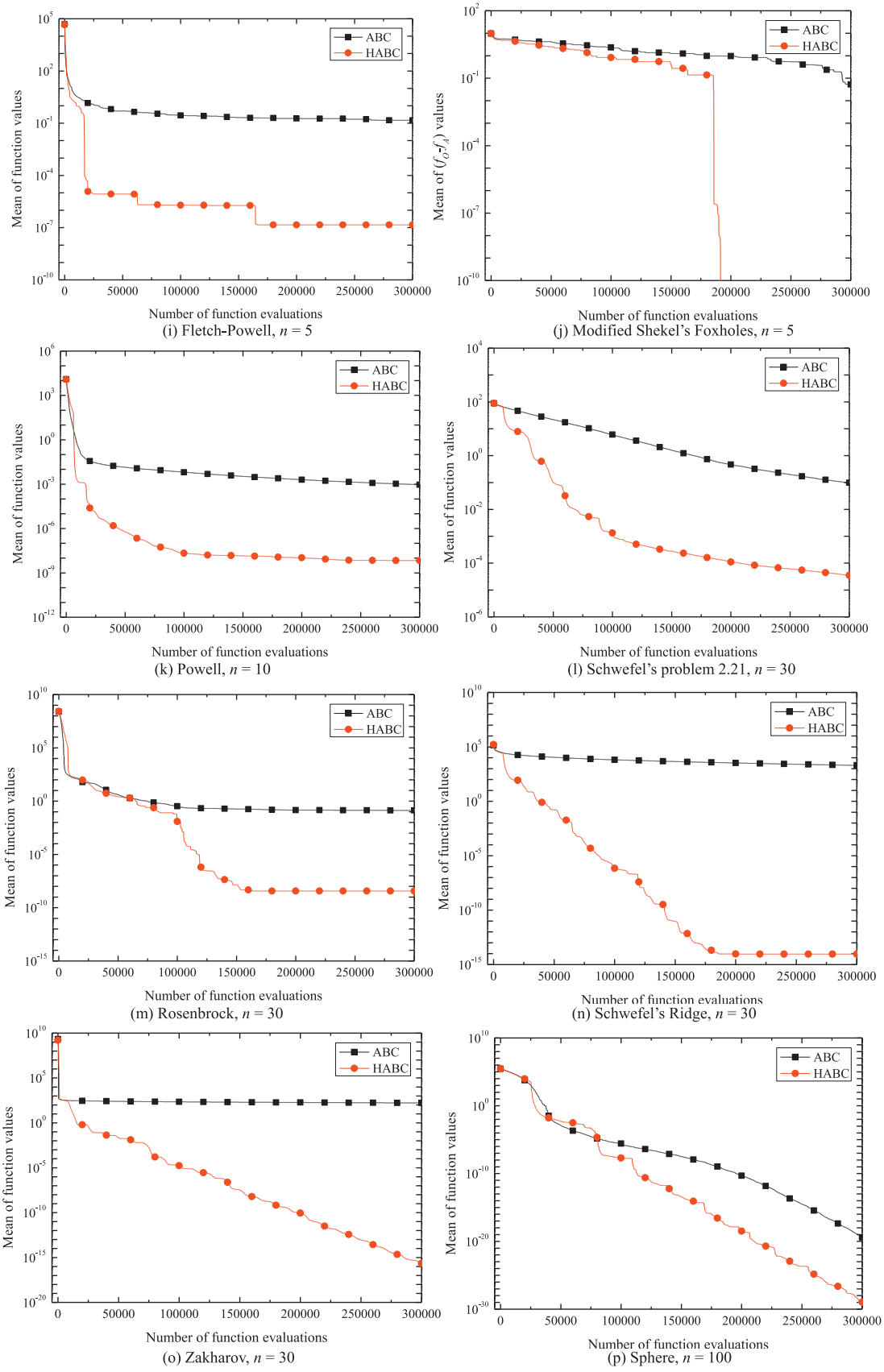


Fig. 9. (*Continued*).

Table 3
Fixed-iteration comparison between ABC and HABC. GM: global minimum.

No.	Function	n	GM	ABC		HABC		Signed-rank test	
				Mean	(SD)	Mean	(SD)	p-value	h
1	BE	2	0	2.0718e−13	(9.1463e−13)	**1.8769e−14**	**(3.3566e−14)**	0.0940	0
2	GP	2	3.0	3.0000100	(3.7232e−05)	**3.0000000**	**(1.9720e−16)**	0.0050	1
3	MA	2	0	4.0492e−14	(1.9543e−13)	**2.5486e−15**	**(8.7608e−15)**	0.8205	0
4	SC	2	0	4.4402e−08	(1.7112e−07)	**0**	**(0)**	7.95e−07	1
5	SF	2	0.998004	0.9980040	(0)	0.9980040	(0)	1.0	0
6	SH	2	−186.7309	−186.7309	(9.0061e−14)	−186.7309	(8.8118e−14)	1.0	0
7	$H_{3,4}$	3	−3.86278	−3.862782	(2.6916e−15)	−3.862782	(2.6916e−15)	1.0	0
8	HV	3	0	5.5088e−09	(2.3903e−08)	**1.5467e−13**	**(7.7326e−13)**	7.56e−10	1
9	CO	4	0	1.5009e−01	(8.8776e−02)	**4.0874e−08**	**(7.5055e−08)**	7.56e−10	1
10	KO	4	3.075e−4	4.3215e−04	(6.3152e−05)	**3.0751e−04**	**(5.6600e−09)**	7.56e−10	1
11	PE	4	0	1.7810e−03	(3.0784e−03)	**5.6395e−07**	**(8.7011e−07)**	8.03e−10	1
12	PS	4	0	6.6536e−03	(5.7332e−03)	**2.8877e−04**	**(5.3071e−04)**	7.56e−10	1
13	$S_{4,10}$	4	−10.53641	−10.536410	(1.1090e−14)	−10.536410	(1.0052e−14)	1.0	0
14	FP	5	0	1.4593e−01	(1.4422e−01)	**1.4578e−07**	**(6.8194e−07)**	7.56e−10	1
15	ML	5	−0.965	−0.9649590	(7.8491e−05)	**−0.9650000**	**(1.1266e−15)**	1.42e−08	1
16	MS	5	−10.4056	−10.352770	(8.1346e−02)	**−10.405620**	**(9.3962e−15)**	7.56e−10	1
17	$H_{6,4}$	6	−3.32237	−3.3223680	(8.9720e−16)	−3.3223680	(8.9720e−16)	1.0	0
18	FP	10	0	1.2616e+01	(1.2322e+01)	**1.5061e+00**	**(3.9671e+00)**	2.48e−08	1
19	ML	10	−0.965	−0.5474770	(1.1611e−01)	**−0.5529050**	**(1.2180e−01)**	1.26e−02	1
20	MS	10	−10.2088	−8.5263420	(2.9684e+00)	**−9.3581360**	**(2.3280e+00)**	1.02e−04	1
21	MI	10	−9.66015	−9.6601520	(3.5888e−16)	−9.6601520	(4.3953e−16)	0.8125	0
22	WI	10	0	4.5777e−01	(8.0434e−01)	**2.3675e−01**	**(1.2693e+00)**	7.33e−06	1
23	PO	24	0	9.2588e−04	(1.1348e−04)	**6.8295e−09**	**(1.5157e−08)**	7.56e−10	1
24	QU	30	0	3.4251e−02	(7.3449e−03)	**1.8927e−02**	**(6.3893e−03)**	1.47e−09	1
25	S21	30	0	9.7380e−02	(3.9357e−02)	**3.5291e−05**	**(2.7927e−05)**	7.56e−10	1
26	S26	30	0	0	(0)	0	(0)	1.0	0
27	ST	30	0	0	(0)	0	(0)	1.0	0
28	WE	30	0	0	(0)	0	(0)	1.0	0
29	AC	30	0	**3.2421e−14**	**(4.0577e−15)**	3.4553e−14	(4.4339e−15)	0.0129	2
30	GR	30	0	0	(0)	0	(0)	1.0	0
31	P1	30	0	1.5704e−32	(1.6588e−47)	1.5704e−32	(1.6588e−47)	1.0	0
32	P2	30	0	1.3497e−32	(8.2941e−48)	1.3497e−32	(8.2941e−48)	1.0	0
33	RA	30	0	0	(0)	0	(0)	1.0	0
34	RO	30	0	1.3431e−01	(1.9219e−01)	**3.6891e−09**	**(2.1114e−08)**	7.56e−10	1
35	SR	30	0	1.9868e+03	(8.3930e+02)	**9.3230e−15**	**(6.5923e−14)**	7.56e−10	1
36	SP	30	0	1.9285e−93	(2.5889e−93)	**2.1058e−97**	**(5.2736e−97)**	7.56e−10	1
37	S22	30	0	1.9832e−96	(5.9984e−96)	**5.7468e−99**	**(1.4200e−98)**	7.56e−10	1
38	ZA	30	0	1.6466e+02	(3.0345e+01)	**2.2329e−16**	**(5.5888e−16)**	7.56e−10	1
39	AC	100	0	2.5317e−11	(9.0195e−12)	**6.4732e−13**	**(1.0593e−12)**	7.56e−10	1
40	GR	100	0	8.3267e−16	(2.8695e−15)	**0**	**(0)**	4.88e−04	1
41	P1	100	0	2.1030e−22	(9.0628e−22)	**2.8638e−28**	**(9.8923e−28)**	7.56e−10	1
42	P2	100	0	4.5454e−20	(6.3950e−20)	**1.4409e−27**	**(4.2037e−27)**	7.56e−10	1
43	RA	100	0	**5.7189e−09**	**(4.0438e−08)**	6.8933e−01	(8.2312e−01)	8.03e−10	2
44	RO	100	0	1.0828030	(1.3888910)	**2.5656e−01**	**(9.5927e−01)**	3.85e−06	1
45	SR	100	0	9.6249e+04	(8.0981e+03)	**2.3822e+00**	**(1.9318e+00)**	7.56e−10	1
46	SP	100	0	3.8114e−20	(3.6304e−20)	**1.7712e−29**	**(4.1747e−29)**	7.56e−10	1
47	S22	100	0	1.1441e−22	(1.1098e−22)	**5.1357e−31**	**(1.5247e−30)**	7.56e−10	1
48	ZA	100	0	1.3027e+03	(1.0086e+02)	**4.5063e+01**	**(2.0540e+01)**	7.56e−10	1

Meaning of h values: 0 – there is no difference; 1 – HABC is better; 2 – ABC is better.

functions are summarized in Table 4. The average of the objective function evaluation numbers was evaluated in relation to only the successful minimizations. The best results of *NFE* and success rate (*SR*) for each function are highlighted in boldface. The average acceleration rate AR_{ave} and the average success rate SR_{ave} are listed in the last row of Table 4.

As can be inferred from Table 4, HABC converged much faster than ABC in most cases except for six functions. The overall average acceleration rate of HABC to ABC is 7.48. Except for four functions with *AR* larger than 20, the average acceleration rate of HABC to ABC is 1.79. To check whether HABC saves computational time, the execution time of 50 runs was also shown in Table 4. Less time is needed by HABC for most functions.

As for the success rate, there are 16 functions with success rate equals to zero by ABC, whereas the number of HABC is three. The *SR* of HABC is only less than that of ABC for one function. The average success rate of HABC was 0.84, which is much higher than the 0.61 obtained by ABC. Therefore, it can be concluded that HABC is more robust than ABC.

4.4. Comparison with other algorithms

The performance of HABC was also compared with the GABC [16], and the results are shown in Table 5. The parameters are set the same as [16]. It is clear that HABC performs better than GABC on these problems. The performance of HABC was further compared with several hybrid algorithms listed in Table 6. The parameters $NFE_{\max}$, ε_{rel} and ε_{abs} were set the same as Section 4.3. The number of function evaluations and success rates of each algorithm were given in Tables 7 and 8. There are only two algorithms GA-SPO and HABC can achieve 100% success rates on all the 15 test functions. HABC has a moderate convergence speed, but it is more efficient than GA-PSO in most cases.

4.5. Discussion

The experimental results reveal that ABC still has some deficiency in dealing with functions having narrow curving valley, functions with high eccentric ellipse and some extremely complex

Table 4
Comparison between ABC and HABC in terms of number of function evaluations, success rate and execution time.

No.	Function	n	ABC			HABC			*AR*
			NFE	SR	Time (s)	*NFE*	SR	Time (s)	
1	BE	2	22,340	1.0	0.922	**635**	1.0	**0.078**	35.18
2	GP	2	27,773	1.0	0.953	**683**	1.0	**0.063**	40.66
3	MA	2	21,867	1.0	0.328	**616**	1.0	**0.062**	35.50
4	SC	2	61,087	1.0	1.140	**8002**	1.0	**0.234**	7.63
5	SF	2	**890**	1.0	**0.359**	1058	1.0	**0.453**	0.84
6	SH	2	2114	1.0	0.156	**622**	1.0	**0.078**	3.40
7	$H_{3,4}$	3	1014	1.0	0.171	**782**	1.0	**0.140**	1.30
8	HV	3	84,229	1.0	3.484	**1081**	1.0	**0.093**	77.92
9	CO	4	–	0	16.28	**31,038**	**0.98**	**2.156**	–
10	KO	4	–	0	8.640	**6087**	**1.0**	**0.250**	–
11	PE	4	–	0	35.38	**82,689**	**0.76**	**16.05**	–
12	PS	4	–	0	41.58	**107,524**	**0.14**	**39.05**	–
13	$S_{4,10}$	4	7827	1.0	2.421	**1403**	1.0	**0.468**	5.58
14	FP	5	–	0	119.0	**33,141**	**0.98**	**15.23**	–
15	ML	5	99,815	0.96	25.19	**78,888**	**0.98**	**19.36**	1.27
16	MS	5	130,562	0.10	41.86	**40,808**	**1.0**	**6.343**	3.20
17	$H_{6,4}$	6	3240	1.0	0.718	**1636**	1.0	**0.375**	1.98
18	FP	10	–	0	407.2	**97,620**	**0.80**	**183.3**	–
19	ML	10	171,255	0.02	116.1	**151,092**	**0.06**	**112.3**	1.13
20	MS	10	152,711	0.02	79.53	**98,815**	**0.80**	**38.88**	1.55
21	MI	10	25,381	1.0	3.718	**24,839**	1.0	**3.578**	1.02
22	WI	10	–	0	482.6	**28,589**	**0.94**	**73.45**	–
23	PO	24	–	0	51.20	**42,859**	**1.0**	**7.516**	–
24	QU	30	–	0	**11.08**	–	0	12.03	–
25	S21	30	–	0	**22.56**	**148,730**	**0.06**	22.77	–
26	S26	30	**78,428**	1.0	**15.98**	96,080	1.0	19.52	0.82
27	ST	30	**6816**	1.0	**0.546**	15,253	1.0	1.141	0.45
28	WE	30	66,915	1.0	557.8	**61,332**	1.0	**517.8**	1.09
29	AC	30	58,103	1.0	7.781	**41,133**	1.0	**5.453**	1.41
30	GR	30	56,555	1.0	8.625	**44,508**	1.0	**7.140**	1.27
31	P1	30	25,300	1.0	13.44	**21,085**	1.0	**11.14**	1.20
32	P2	30	33,688	1.0	18.11	**25,323**	1.0	**13.53**	1.33
33	RA	30	**52,326**	1.0	**5.687**	61,400	1.0	6.515	0.85
34	RO	30	–	0	114.02	**92,759**	**0.98**	**37.30**	–
35	SR	30	–	0	57.70	**90,552**	**1.0**	**17.42**	–
36	SP	30	31,778	1.0	1.218	**17,770**	1.0	**0.718**	1.79
37	S22	30	27,680	1.0	2.265	**12,085**	1.0	**1.031**	2.29
38	ZA	30	–	0	15.48	**126,788**	**1.0**	**6.313**	–
39	AC	100	204,092	1.0	72.80	**167,343**	1.0	**58.38**	1.22
40	GR	100	134,512	1.0	61.92	**112,250**	1.0	**53.44**	1.20
41	P1	100	**73,294**	1.0	**123.6**	90,833	1.0	154.6	0.81
42	P2	100	112,612	1.0	189.2	**95,472**	1.0	**164.5**	1.18
43	RA	100	**227,535**	**1.0**	**69.75**	272,630	0.44	87.42	0.83
44	RO	100	–	0	372.9	**271,382**	**0.34**	**363.0**	–
45	SR	100	–	0	**495.6**	–	0	501.8	–
46	SP	100	105,202	1.0	8.625	**65,572**	1.0	**5.281**	1.60
47	S22	100	95,891	1.0	21.19	**54,233**	1.0	**11.95**	1.77
48	ZA	100	–	0	33.39	–	0	**28.13**	–
Ave.				0.61			0.84		7.48

Table 5
Comparison between GABC and HABC. F: function, n: dimensionality.

F(n)	GABC		HABC	
	Mean	SD	Mean	SD
SC(2)	0	0	0	0
SC(3)	1.85e−18	1.01e−17	0	0
RO(2)	1.68e−4	1.45e−4	4.01e−8	1.82e−7
RO(3)	2.66e−3	2.22e−3	7.64e−8	5.67e−7
SP(30)	4.18e−16	7.37e−17	6.66e−85	1.13e−84
SP(60)	1.43e−15	1.38e−16	3.42e−43	1.18e−42
GR(30)	2.96e−17	4.99e−17	0	0
GR(60)	7.55e−16	4.13e−16	0	0
RA(30)	1.33e−14	2.45e−14	0	0
RA(60)	3.52e−13	1.24e−13	0	0
AC(30)	3.22e−14	3.25e−15	3.18e−14	3.75e−15
AC(60)	1.00e−13	6.09e−15	8.56e−14	1.05e−14

multimodal functions. The accuracy, success rate, and convergence speed of HABC shows considerable improvement in the case of some hard problems for ABC, such as Rosenbrock, Powell, Schwefel Ridge, Zakharov, Fletch-Powell and so on. This is because the pattern move operator of Hooke–Jeeves pattern search can increase the number of search directions of ABC. Meanwhile, the best solution in HABC has more chance to contribute to the production of trial solutions than ABC. On the other hand, the population in

Table 6
List of various hybrid methods for comparison.

Method	Reference
Hooke–Jeeves ABC (HABC)	This work
Continuous hybrid algorithm (CHA)	[5]
Directed tabu search using adaptive pattern search strategy (DTS_{APS})	[40]
Hybrid genetic algorithm and PSO (GA-PSO)	[41]
Hybrid method using low-discrepancy sequences and simplex search (LPτNM)	[42]

Table 7
The number of function evaluations needed to achieve the accuracy. F: function, n: dimensionality.

F(n)	CHA	DTS_{APS}	$LP_{\tau}NM$	GA-PSO	HABC
RC(2)	295	212	247	8254	620
BO1(2)	132	–	–	174	852
ES(2)	952	223	248	809	850
GP(2)	259	230	182	25,706	683
SH(2)	345	274	303	96,211	622
SP(3)	371	446	266	206	905
$H_{3,4}(3)$	492	438	292	2117	785
$S_{4,5}(4)$	698	819	839	529,344	1236
$S_{4,7}(4)$	620	812	837	56,825	1306
$S_{4,10}(4)$	635	828	1079	43,314	1403
RO(5)	3290	1684	2353	1,358,064	35,115
ZA(5)	950	1003	1163	398	1628
$H_{6,4}(6)$	930	1784	1552	12,568	1636
RO(10)	14,563	9037	9188	5,319,160	27,396
ZA(10)	4291	4032	6826	872	12,116

HABC can still keep large diversity to avoid premature convergence while approaching an optimum by incorporating a local search technique.

The performance of standard evolutionary algorithms such as genetic algorithms can also be improved by a local search method [5,43]. The advantage of horizontal information transfer in genetic algorithms has already been showed in Ref. [43]. Artificial bee colony (ABC) algorithm and evolutionary algorithms are both use observed performance function values, and in fact ABC is as concise as an evolutionary algorithm, since it has only one main operator shown as Eq. (2). Therefore, ABC is no more complicated than an evolutionary algorithm. So we merely studied hybrid artificial bee colony algorithm, and the proposed algorithm was compared with other hybrid evolutionary algorithms. From the literature [44] we can know that the classical gradient based algorithms or direct search algorithms provide only one solution are suitable for local search. On the other hand, population based algorithms, like genetic algorithm and ABC adopt multipoint search strategies, and they are suitable for global optimization. The classical search algorithms generally can perform well on unimodal problems and population based evolutionary algorithms tend to outperform the classical algorithms on multimodal functions since they are likely to cover the search space better and are less likely to get trapped in the local minima. So the classical mathematical optimum search algorithms and the proposed population based algorithm are not compared again in this paper.

Fuzzy systems and fuzzy logic have been paid extensive concerns in science and engineering. It has already been showed that meta-heuristic algorithms like genetic algorithms [45] and particle swarm optimization [46] are powerful tools for optimization of fuzzy membership functions which are very important to fuzzy systems. The proposed HABC algorithm is also a meta-heuristic algorithm for continuous numerical optimization and it also has the potential to be applied to determine the fuzzy membership functions of fuzzy systems.

Table 8
The success rates of different hybrid algorithms. F: function, n: dimensionality.

F(n)	CHA	DTS_{APS}	$LP_{\tau}NM$	GA-PSO	HABC
RC(2)	100	100	100	100	100
BO1(2)	100	–	–	100	100
ES(2)	100	82	100	100	100
GP(2)	100	100	100	100	100
SH(2)	100	100	85	100	100
SP(3)	100	100	100	100	100
$H_{3,4}(3)$	100	100	100	100	100
$S_{4,5}(4)$	85	75	100	100	100
$S_{4,7}(4)$	85	65	100	100	100
$S_{4,10}(4)$	85	52	96	100	100
RO(5)	100	85	91	100	100
ZA(5)	100	100	100	100	100
$H_{6,4}(6)$	100	83	100	100	100
RO(10)	83	100	88	100	100
ZA(10)	100	85	100	100	100

5. Conclusion

A hybrid global optimization algorithm called HABC has been proposed, studied and discussed. It combines two search techniques: the ABC algorithm and a slightly modified pattern search method. Both ABC and Hooke–Jeeves pattern search are derivative-free algorithms for global optimization, so the proposed HABC also does not require the derivatives of the optimized objective functions. The proposed HABC algorithm has been tested on extensive benchmark mathematical functions and has shown very reliable performance in most cases. When compared with ABC and other well-known or recently proposed algorithms, the proposed new method has demonstrated strongly competitive results in terms of number of function evaluations, success rate and accuracy. Practical application of the new approach would also be worth further studying.

Acknowledgements

This research was supported by National Natural Science Foundation of China under grant no. 51109028, 90815024 and the Fundamental Research Funds for the Central Universities under grant no. DUT11RC(3)38.

Appendix A. List of test functions

Several functions are listed here, and the other functions can be seen in [21]. The source code in C for these functions can be obtained by e-mail. To define the test problems, we have adopted the following format:

No. Name (Symbol) Description of the problem

1. Beale (BE)

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2,$$
$$\text{subject to } -4.5 \le x_1, \quad x_2 \le 4.5.$$

It is a unimodal function, and the global minimum is located at $\boldsymbol{x}^* = (3, 0.5)$ with $f(\boldsymbol{x}^*) = 0$.

2. Goldstein and price (GP)

$$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)], \quad \text{subject to } -2 \le x_1, \quad x_2 \le 2.$$

It is a multimodal function, and the global minima is located at $\boldsymbol{x}^* = (0, -1)$ with $f(\boldsymbol{x}^*) = 3$

3. Matyas (MA)

$$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2, \quad \text{subject to } -10 \le x_1, x_2 \le 10.$$

It is a unimodal function, and the global minimum is located at $\boldsymbol{x}^* = (0, 0)$ with $f(\boldsymbol{x}^*) = 0$.

4. Schaffer (SC)

$$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^{n} x_i^2}\right) - 0.5}{\left(1 + 0.001\left(\sum_{i=1}^{n} x_i^2\right)\right)^2},$$

$$\text{subject to } -100 \le x_i \le 100, \quad i = 1, \ldots, n.$$

It is a multimodal function, and the global minima is located at $\boldsymbol{x}^* = (0, 0)$ with $f(\boldsymbol{x}^*) = 0$.

References

[1] A. Georgieva, I. Jordanov, Global optimization based on novel heuristics, low-discrepancy sequences and genetic algorithms, European Journal of Operational Research 196 (2) (2009) 413–422.
[2] H.Md. Azamathulla, F.C. Wu, Support vector machine approach for longitudinal dispersion coefficients in natural streams, Applied Soft Computing 11 (2) (2011) 2902–2905.
[3] H.Md. Azamathulla, A.Ab. Ghani, S.Y. Fei, ANFIS-based approach for predicting sediment transport in clean sewer, Applied Soft Computing 12 (3) (2012) 1227–1230.
[4] H.Md. Azamathulla, F.C. Wu, A.Ab. Ghani, S.M. Narulkar, N.A. Zakaria, C.K. Chang, Comparison between genetic algorithm and linear programming approach for real time operation, Journal of Hydro-environment Research 2 (3) (2008) 172–181.
[5] R. Chelouah, P. Siarry, Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions, European Journal of Operational Research 148 (2) (2003) 335–348.
[6] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer-Verlag, Berlin, 2005.
[7] F. Kang, J. Li, Q. Xu, Damage detection based on improved particle swarm optimization using vibration data, Applied Soft Computing 12 (8) (2012) 2329–2335.
[8] Y.S. Ong, A.J. Keane, Meta-Lamarckian learning in memetic algorithm, IEEE Transactions on Evolutionary Computation 8 (2) (2004) 99–110.
[9] J.E. Smith, Co-evolving memetic algorithms: a review and progress report, IEEE Transactions on Systems Man and Cybernetics Part B 37 (1) (2007) 6–17.
[10] Q.C. Nguyen, Y.S. Ong, M.H. Lim, A probabilistic memetic framework, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 604–623.
[11] F. Neri, E. Mininno, Memetic compact differential evolution for Cartesian robot control, IEEE Computational Intelligence Magazine 5 (2) (2010) 54–65.
[12] L. Wang, Y. Xu, L. Li, Parameter identification of chaotic systems by hybrid Nelder–Mead simplex search and differential evolution algorithm, Expert Systems with Applications 38 (4) (2011) 3238–3245.
[13] D. Karaboga, B. Akay, A survey: algorithms simulating bee swarm intelligence, Artificial Intelligence Review 31 (1–4) (2009) 61–85.
[14] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.
[15] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Systems with Applications 37 (8) (2010) 5682–5687.
[16] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied Mathematics and Computation 217 (7) (2010) 3166–3173.
[17] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in artificial bee colony algorithm, Applied Soft Computing 11 (2) (2011) 2888–2901.
[18] W. Gao, S. Liu, A modified artificial bee colony algorithm, Computers & Operations Research 39 (3) (2012) 687–697.
[19] G. Li, P. Niu, X. Xiao, Development and investigation of efficient artificial bee colony algorithm for numerical function optimization, Applied Soft Computing 12 (1) (2012) 320–332.
[20] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation 214 (1) (2009) 108–132.
[21] F. Kang, J. Li, Z. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Information Sciences 181 (16) (2011) 3508–3531.
[22] N. Karaboga, A new design method based on artificial bee colony algorithm for digital IIR filters, Journal of The Franklin Institute 346 (4) (2009) 328–348.
[23] S.L. Sabat, S.K. Udgata, A. Abraham, Artificial bee colony algorithm for small signal model parameter extraction of MESFET, Engineering Applications of Artificial Intelligence 23 (5) (2010) 689–694.
[24] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers & Structures 87 (13–14) (2009) 861–870.
[25] N. Karaboga, S. Kockanat, H. Dogan, The parameter extraction of the thermally annealed Schottky barrier diode using the modified artificial bee colony, Applied Intelligence (2012), http://dx.doi.org/10.1007/s10489-012-0372-x.
[26] D. Karaboga, C. Ozturk, A novel clustering approach: artificial bee colony (ABC) algorithm, Applied Soft Computing 11 (1) (2011) 652–657.
[27] H. Li, J. Li, F. Kang, Risk analysis of dam based on artificial bee colony algorithm with fuzzy c-means clustering, Canadian Journal of Civil Engineering 38 (5) (2011) 483–492.
[28] M. Horng, Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation, Expert Systems with Applications 38 (11) (2011) 13785–13791.
[29] E. Cuevas, F. Sención, D. Zaldivar, M. Pérez-Cisneros, H. Sossa, A multi-threshold segmentation approach based on artificial bee colony optimization, Applied Intelligence 37 (3) (2012) 321–336.
[30] S.K. Mandal, F.T.S. Chan, M.K. Tiwari, Leak detection of pipeline: an integrated approach of rough set theory and artificial bee colony trained SVM, Expert Systems with Applications 39 (3) (2012) 3071–3080.
[31] M. Sonmez, Artificial bee colony algorithm for optimization of truss structures, Applied Soft Computing 11 (2) (2011) 2406–2418.
[32] F. Kang, J. Li, Z. Ma, An artificial bee colony algorithm for locating the critical slip surface in slope stability analysis, Engineering Optimization (2012), http://dx.doi.org/10.1080/0305215X.2012.665451.
[33] H. Li, J. Li, F. Kang, Application of the artificial bee colony algorithm-based projection pursuit method in statistical rock mass stability estimation, Environmental Earth Sciences (2012), http://dx.doi.org/10.1007/s12665-012-1912-8.
[34] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review (2012), http://dx.doi.org/10.1007/s10462-012-9328-0.
[35] F. Kang, J. Li, Z. Ma, H. Li, Artificial bee colony algorithm with local search for numerical optimization, Journal of Software 6 (3) (2011) 490–497.
[36] R. Hooke, T.A. Jeeves, Direct search solution of numerical and statistical problems, Journal of the ACM 8 (2) (1961) 212–229.
[37] V. Torczon, On the convergence of pattern search algorithms, SIAM Journal on Optimization 7 (1) (1997) 1–25.
[38] M.M. Ali, C. Khompatraporn, Z.B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, Journal of Global Optimization 31 (4) (2005) 635–672.
[39] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (1) (2008) 687–697.
[40] A. Hedar, M. Fukushima, Tabu search directed by direct search methods for nonlinear global optimization, European Journal of Operational Research 170 (2) (2006) 329–349.
[41] Y.T. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Applied Soft Computing 8 (2) (2008) 849–857.
[42] I. Jordanov, A. Georgieva, A hybrid meta-heuristic for global optimization using low-discrepancy sequences of points, Computers & Operations Research 37 (3) (2010) 456–469.
[43] H.J.C. Barbosa, C.C. Lavor, F.M.P. Raupp, A GA-simplex hybrid algorithm for global minimization of molecular potential energy functions, Annals of Operations Research 138 (1) (2005) 189–202.
[44] R. Salomon, Evolutionary algorithms and gradient search: similarities and differences, IEEE Transactions on Evolutionary Computation 2 (2) (1998) 45–55.
[45] M. Mucientes, D.L. Moreno, A. Bugarín, S. Barro, Design of a fuzzy controller in mobile robotics using genetic algorithms, Applied Soft Computing 7 (2) (2007) 540–546.
[46] Y. Maldonado, O. Castillo, P. Melin, Particle swarm optimization of interval type-2 fuzzy systems for FPGA applications, Applied Soft Computing 13 (1) (2013) 496–508.