An Improved Artificial Bee Colony Algorithm

Fei Kang, Junjie Li, Haojin Li
Faculty of Infrastructure Engineering
Dalian University of Technology
Dalian, China
kangfei2009@163.com, lijunjie@dlut.edu.cn

Zhenyue Ma, Qing Xu
School of Hydraulic Engineering
Dalian University of Technology
Dalian, China
dmzy@dlut.edu.cn, xuqing@dl.cn

***Abstract*—Artificial bee colony (ABC) algorithm is one of the most recently proposed swarm intelligence algorithms for global optimization. It performs well in most cases; however, there still exist some problems it cannot solve very well. This paper presents a novel hybrid Hooke Jeeves ABC (HJABC) algorithm with intensification search based on the Hooke Jeeves pattern search and the ABC. The main purpose is to demonstrate how the standard ABC can be improved by incorporating a hybridization strategy. The proposed algorithm is tested on 7 benchmark functions including a wide range of dimensions. Numerical results show that the new algorithm is promising in terms of convergence speed, success rate and solution accuracy.**

Keywords-swarm intelligence; artificial bee colony algorithm; pattern search; global optimization; benchmark functions

I. INTRODUCTION

Unconstrained global optimization problems can be formulated as following model:

$$\min f(x), x = (x_1, x_2, ..., x_n) \tag{1}$$

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

The foraging behavior, learning, memorizing and information sharing characteristics of bees have recently been one of the most interesting research areas in swarm intelligence. Studies on honey bees are in an increasing trend in the literature during the last few years. Artificial bee colony (ABC) algorithm was proposed by Karaboga in 2005 [1-3]. It is an optimization algorithm based on particular intelligent behavior of honey bee swarms.

According to the recent studies [4], ABC is better than or similar to other population-based algorithms with the advantage of employing fewer control parameters. However, it still has some deficiency in deal with functions having narrow curving valley, functions with high eccentric ellipse and some extremely complex multimodal functions. In order to improve the performance of ABC, Kang [5] has proposed a hybrid simplex ABC for structural inverse analysis and performs well on low dimensional problems. Compared to Nelder-Mead simplex search (NM), Hooke Jeeves (HJ) algorithm is suitable to solve more functions with high dimensions and it is better at adapting to the changes and at adjusting the step length than NM [6]. In order to offset the default of ABC mentioned above and improve its convergence speed, a Hooke Jeeves artificial bee colony algorithm (HJABC) with intensification search is proposed for unstrained global optimization. The algorithm matins the main steps of ABC and incorporates a local search technique which is based on HJ. The efficiency of the new algorithm is proved by comparison with the basic ABC on 7 selected functions, which having narrow curving valley, with high eccentric ellipse or are extremely complex multimodal functions.

The rest of the paper is organized as follows. Section □ reviews the fundamentals of the original ABC. Section □ introduces the Hooke Jeeves method and then describes the HJABC hybrid structure and algorithm. Section □ presents comparative studies on benchmark functions. Conclusions are given in section □.

II. ARTIFICIAL BEE COLOBY ALGORITHM

ABC is a swarm intelligent optimization algorithm inspired by honey bee foraging [1-3]. In ABC, the colony of the artificial bees contains three groups of bees: employed bees, onlookers and scouts. The first half of the colony consists of the employed bees and the second half includes the onlookers. For every food source, there is only one employed bee. The employed bee of an abandoned food source becomes a scout.

The position of a food source represents a possible solution of the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. At the first step, the algorithm generates a randomly distributed initial population contains *NS* solutions. Where *NS* is the number of food sources and it is equal to the number of employed bees. Each solution x_i (i=1,2,…,*NS*) is a n-dimensional vector.

In ABC, the fitness function is defined as follows:

$$fit_i = \begin{cases} \dfrac{1}{1+f_i} & f_i \geq 0 \\ 1+abs(f_i) & f_i < 0 \end{cases} \tag{1}$$

where f_i is the objective function value of solution i, fit_i is the fitness value of solution i after transformation.

An onlooker bee chooses a food source depending on the probability value p_i associated with that food source,

978-1-4244-5874-5/10/$26.00 ©2010 IEEE

$$p_i = fit_i \Big/ \sum_{j=1}^{NS} fit_j \tag{2}$$

A candidate solution v_i from the old solution x_i can be generated as

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{3}$$

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

After each candidate source position is produced and evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source has equal or better quality than the old source, the old one is replaced by the new one. Otherwise, the old one is retained.

If a position cannot be improved further through a predetermined number *limit* (limited cycles), then that food source is assumed to be abandoned. The corresponding employed bee becomes a scout. The abandoned position will be replaced with a new food source found by the scout. Assume that the abandoned source is x_i, and then the scout discovers a new food source as

$$x_{ij} = l_j + rand(0,1)\left(u_j - l_j\right) \tag{4}$$

where l_j and u_j are lower and upper bounds of the variable x_{ij}.

III. HOOKE JEEVES ABC WITH INTENSIFICATION SEARCH

A. Hooke Jeeve Algorithm

Hooke and Jeeves pattern search method is a simple yet very effective optimization technique proposed in 1961 [7]. Today, it is still a popular tool for various optimization problems, especially for deterministic local search.

The basic HJ method is modified to accommodate the hybrid strategy. The main characteristics of the modified HJ method are: (I) to accelerate the procedure, direct search takes advantage of its knowledge of the sign of its previous move in each of the directions; (Ⅱ) a different step size for each variable is used to adaptive to the scaling problems of different variables.

In the HJ method a combination of exploratory move (EM) and pattern move (PM) is made iteratively to search out the optimum solution for the problem. Exploratory searches and pattern moves are repeated until a termination criterion is met. Assume that the current solution (the base point) is x, the current minimum objective value is $f_{\min}$, z is a temporary vector to store the obtained point after EM. The main steps of EM are shown in Fig.1.

Given x_1 and x_0 ($x_1 < x_0$), the PM takes the step $x_1 - x_0$ from x_0 as

$$x_2 = x_1 + (x_1 - x_0) \tag{5}$$

where x_2 is the point obtained from PM. The HJ 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 PM is to investigate whether further progress is possible in the general direction $x_1 - x_0$ (since, if $x_1 < x_0$, then $x_1 - x_0$ is clearly a promising direction) [8]. The main steps of HJ are shown in Fig.2.

1: Initialize i=1 and z=x.
2: z_i=x_i+δ_i; if ($f(z) < f_{\min}$), $f_{\min} = f(z)$, go to step 4;else go to step 3.
3: z_i=x_i-δ_i; if ($f(z) < f_{\min}$), $f_{\min} = f(z)$, go to step 4; else z_i=x_i.
4: If i=n, set i=i+1 and go to step 1; else z is the result of EM and go to step 4.
5: If $f(z) >= f_{\min}$, failure; else success.

Fig. 1. The main steps of EM operator.

1: Choose the starting point x_0, the current step size δ_i (i=1, 2,…,n), the reduction factor for step size ρ<1, the termination parameter ε>0 and the max iterations $k_{\max}$. Initializing the iteration counter k=1, the auxiliary step size s_a=1.0.
2: Perform an exploratory move (referred as EM) with x_0 as the base point and the obtained point is x_1. If the exploratory move is successful go to step3; else go to step 6.
3: Perform a pattern move $x_2 = x_1 + (x_1 - x_0)$ and set $x_0 = x_1$.
4: Perform exploratory move with x_2 as the base point and the obtained point is x_1.
5: If $f(x_1) < f(x_0)$, go to step 3, else go to step 6.
6: If $s_a < \varepsilon$ or $k \geq k_{\max}$, 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. 2. The main steps of HJ algorithm.

B. The proposed Algorithm

The HJABC is proposed considering the local search property and pattern move operator of HJ is complementary for ABC.

In the original ABC, the fitness value is calculated by (1) to select a source for an onlooker bee. However, this selection strategy is difficult to maintain diversity and avoid premature convergence [9]. To overcome this problem, rank-based fitness transformation is adopted as

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

where p_i is the position of the solution in the whole population after ranking, $SP \in [1.0,2.0]$ is the selection pressure and a medium value of SP=1.5 can be a good choice.

The HJ algorithm is incorporated into ABC as a local exploration tool. The main steps of the hybrid algorithm are summarized as below. Every *interval* cycles of ABC, HJ is activated to perform a local search using the current best solution as the base point. The step size δ should 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 follows:

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

where δ_j is the step size of the *jth* dimension, m 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 HJ search. Sometime the step can become large again because of the scout operator to avoid premature convergence. The iteration times of HJ is controlled by the parameter ε, when $s_a<\varepsilon$ the algorithm will return to the main framework of HJABC.

1: Initialize the population of solutions x_i, i=1,…,*NS*.
2: Evaluate the population, *cycle*=1, k=0.
3: Memorize the best solution x_{best} and set x_{best1}= x_{best}
4: **Repeat**
(*Exploration phase*)
5: Produce new solutions v_i for the employed bees by using (3) and evaluate them.
6: Apply the greedy selection process for the employed bees.
7: Rank the population and calculate the fitness by (6)
8: Calculate the probability P_i for the solutions x_i by (2).
9: Produce the new solutions v_i for the onlookers from the solutions selected depending on P_i and evaluate them.
10: Apply the greedy selection process for the onlookers.
11: Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution x_i.
12: Memorize the best solution x_{best} achieved so far.
(*Exploitation phase*)
13: If ((*cycle* mod *interval*)=0), calculate step size δ_j of HJ according to (7).
14: Call HJ with x_{best} as the base point until $s_a<\varepsilon$ and the obtained point is x_{best2}.
15: If (f(x_{best2})≤f(x_{best})) Replace the solution in the middle position after ranking by x_{best2} and set x_{best}=x_{best2}
16: If (f(x_{best})<f(x_{best1})) set x_{best1}= x_{best} and k=0, else set k=k+1;
17: Set *cycle*=*cycle*+1.
(*Intensification search*)
18: If (*k*>*counter*) perform intensification search by HJ.
19: **Until** a termination condition is met.

Fig. 3. The main steps of HJABC.

For the sake of clarity, the main steps of HJABC are described in Fig. 3. The new algorithm first 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 HJ technique to make a local exploitation search near the best solution. If the alternative process cannot improve the best solution any more, HJ is activated again to refine the obtained solution. This process is repeated until the termination condition is met, e.g., the maximum number of function evaluations (NFE_{max}) is reached. If calling the HJ algorithm for *counter* times can not improve the best solution, the algorithm will exit from the main loop and perform an intensification search by HJ algorithm until the termination condition is met.

The solution in the middle position after ranking is replaced by the obtained better point after HJ search. The worst solution in the population can not be replaced because that will make the scout operator of ABC algorithm useless and premature convergence will take place.

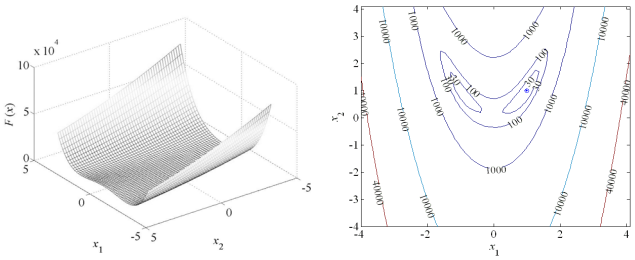
IV. EXPERIMENTAL RESULTS

The proposed new algorithm is compared with the basic ABC in terms of number of function evaluations (*NFE*), success rate (*SR*) and accuracy. The common parameters of ABC and HJABC are set as *NS*=25 (population size is 50), *Limit* = *NS*×*n* and NFE_{max}=200 000. The other parameters of HJABC are set as *interval* = 3×*n* and $\varepsilon=10^{-3}$. The parameter *counter* is set as different values to study its influence. Each of the experiments in this section is repeated 50 trials with different random seeds. Seven benchmark functions are selected to test the proposed algorithm. These functions are listed below:

1. (CO) Colville function (Fig. 4)

$$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$$

with $-10 \le x_i \le 10$, min (*f*) = *f*(1,1,1,1) =0.



(a) Surface plot (b) Contour lines

Fig. 4. Colville function, x_1=x_3, x_2=x_4.

2. (R_n) Rosenbrock function (*n* variables):

$$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

with $-5 \le x_i \le 10$, min (*f*) = *f*(0,0,…,0) =0.

3. (Z_n) Zakharov function

$$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$$

with $-5 \le x_i \le 10$, min (*f*) = *f*(0,0,…,0) =0.

4. (SR) Schwefel Ridge

$$f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$$

with $-100 \le x_i \le 100$, min (*f*) = *f*(0,0,…,0) =0.

5. (PE) Perm function

$$f(x) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i^k + 0.5)((x_i / i)^k - 1)\right]^2$$

with $-n \le x_i \le n$, min (*f*) = *f*(1,2,…,*n*) =0.

6. Kowalik function (f_{12})

$$f(x)=\sum_{i=1}^{11}\left(a_i-\frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right)^2$$

with $-5\le x_i\le 5$, min(f)=f(0.192,0.190,0.123,0.135) =3.075e-4

a=[0.1957,0.1947,0.1735,0.1600,0.0844,0.0627,0.0456,0.0342, 0.0323,0.0235,0.0246];
b=[4.0,2.0,1.0,0.5,0.25,0.167,0.125,0.1,0.0833,0.0714,0.0625].

7. (F_n) Fletch-Powell Problem

$$f(x)=\sum_{i=1}^{n}(A_i-B_i)^2 \qquad A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j)$$

$$B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$$

with $-\pi\le x_i\le\pi$, min (f) =0, all details can be found in [4].

A. Comparison in Terms of NFE and SR

In this comparison, the two algorithms are compared by measuring the number of function evaluations to reach a given accuracy. The termination criterion is the *NFE* has reached the maximum value or the following condition is satisfied,

$$\left|f^*-\tilde{f}\right|<\varepsilon_1 \tag{8}$$

where f^* is the exact global minimum, $\tilde{f}$ is the best function value obtained by the algorithm and the accuracy ε_1 is set equal to 10^{-6}.

A trial is successful, if (8) can be satisfied before *NFE* reaches the maximum value. The average *NFE* of successful runs and the *SR* are listed in Table□. It can be seen that all the function selected are hard for ABC and HJABC can solve all the functions with high *SR*, especially when *counter*=50*n*.

TABLE I. COMPARISON IN TERMS OF *NFE* AND *SR*

	ABC	HJABC *counter*=2*n*	HJABC *counter*=50*n*	HJABC *counter*=200*n*
F	*NFE/SR*	*NFE/SR*	*NFE/SR*	*NFE/SR*
CO	-/0.00	7273/1.00	13391/1.00	13114/1.00
R_{30}	-/0.00	64422/0.98	62634/0.98	65635/1.00
Z_{30}	-/0.00	90796/1.00	89880/1.00	89040/1.00
S_{30}	-/0.00	61941/1.00	63354/1.00	61262/1.00
PE	-/0.00	11104/1.00	46631/1.00	35051/0.92
KO	-/0.00	3503/0.88	6196/1.00	6093/1.00
F_5	-/0.00	10682/0.92	15700/1.00	13461/0.98

B. Comparison in Terms of accuracy

In this comparison, the termination criterion is the *NFE* has reached the maximum value. The average best values (mean) and the standard deviation (SD) of 50 trials are listed in table□. It can be seen that the accuracy of HJABC is much higher than ABC.

For the functions Perm and Colville, a small *counter*=2*n* can get higher accuracy and save *NFE*, while for complex multimodal functions Kowalik and Fletch-Powell, larger value of *counter* is needed. Generally a moderate value *counter*=50*n* is suitable to various functions.

TABLE II. COMPARISON IN TERMS OF ACCURACY

	ABC	HJABC *counter*=2*n*	HJABC *counter*=50*n*	HJABC *counter*=200*n*
	Mean(SD)	Mean(SD)	Mean(SD)	Mean(SD)
CO	1.80e-1 (1.02e-1)	6.21e-14 (3.34e-13)	4.51e-9 (1.65e-8)	1.63e-8 (1.04e-7)
R_{30}	1.53e-2 (1.11e-2)	3.58e-9 (2.49e-8)	2.88e-8 (2.03e-7)	1.17e-8 (8.08e-8)
Z_{30}	1.89e+2 (3.05e+1)	2.14e-14 (5.60e-14)	2.20e-14 (3.46e-14)	1.92e-14 (3.80e-14)
S_{30}	3.01e+3 (1.19e+3)	4.72e-9 (3.33e-8)	4.13e-26 (2.79e-25)	4.51e-20 (3.19e-19)
PE	2.47e-3 (3.12e-3)	9.06e-11 (5.88e-10)	1.43e-7 (2.77e-7)	4.16e-7 (9.58e-7)
KO	4.844e-4 (7.473e-5)	3.441e-4 (1.813e-4)	3.075e-4 (5.563e-11)	3.075e-4 (6.568e-11)
F_5	2.38e-1 (3.86e-1)	5.67e-1 (2.2655)	2.67e-10 (1.78e-9)	1.77e-9 (8.79e-9)

V. CONCLUSION

In this paper, a hybrid global optimization algorithm called HJABC has been proposed, studied and discussed. The proposed HJABC algorithm has been tested on 7 benchmark mathematical functions and has shown very reliable performance in most cases. When compared with the basic ABC, 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 in areas of science, engineering, economics and others would also be worth further studying.

REFERENCES

[1] D.Karaboga, "An idea based on bee swarm for numerical optimization," Tech. Rep. TR-06, Erciyes University , Engineering Faculty , Computer Engineering Department, 2005.

[2] D. Karaboga and B. Basturk. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," J. Global Optim, Vol. 39, pp. 459–471, 2007.

[3] D. Karaboga and B. Basturk. "On the performance of artificial bee colony (ABC) algorithm," Appl. Soft Comput., Vol. 8, pp. 687–697, 2008.

[4] D.Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," Appl. Math. Comput., vol. 214, pp. 108–132, Aug. 2009.

[5] F. Kang, J. Li, and Q. Xu, "Structural inverse analysis by hybrid simplex artificial bee colony algorithms," Comput. Struct., Vol. 87, pp. 861-870, July 2009.

[6] L. M. Hvattum and F. Glover, "Finding local optima of high-dimensional functions using direct search methods," Eur. J. Oper. Res., vol. 195, pp. 31–45, May 2009.

[7] R. Hooke and T. A. Jeeves, "Direct search solution of numerical and statistical problems," Journal of the ACM, vol. 8, pp. 212-229, Mar. 1961.

[8] V. Torczon, "On the convergence of pattern search algorithms," SIAM J. Optim., vol. 7, pp. 1–25, 1997.

[9] L. Bao and J. C. Zeng, "Comparison and analysis of the selection mechanism in the artificial bee colony algorithm," Ninth International Conference on Hybrid Intelligent Systems, pp. 411-416, 2009.