Artificial Bee Colony Algorithm with Local Search for Numerical Optimization

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

Zhenyue Ma and Haojin Li
Dalian University of Technology / School of Hydraulic Engineering, Dalian, China
Email: dmzy@dlut.edu.cn, lihaojin1983@163.com

***Abstract*—Artificial bee colony (ABC) algorithm is one of the most recently proposed swarm intelligence algorithms for global numerical 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 a comprehensive set of 36 complex benchmark functions and a slope stability analysis problem including a wide range of dimensions. Comparisons are made with the basic ABC and some recent algorithms. Numerical results show that the new algorithm is promising in terms of convergence speed, success rate and solution accuracy**

***Index Terms*—swarm intelligence; artificial bee colony algorithm; pattern search; numerical optimization; critical circular slip surface**

I. INTRODUCTION

Optimization is a field with wide applications in many areas of science and engineering, where mathematical modeling is used. 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]. Unconstrained global optimization problems can be formulated as following model:

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

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

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 [2]. Artificial bee colony (ABC) algorithm was proposed by Karaboga in 2005 [3]. It is an optimization algorithm based on particular intelligent behavior of honey bee swarms. ABC has been compared with genetic algorithm, particle swarm optimization (PSO), differential evolution (DE), and evolutionary algorithms [4], [5] on a limited number of test functions. It also has been used for designing IIR filters [6], for the leaf-constrained minimum spanning tree problem [7] and for structural parameter inverse analysis problems [8].

According to the recent studies [9], 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 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 numerical optimization. The algorithm maintains the main steps of ABC and incorporates a local search technique which is based on Hooke Jeeves method (HJ) [10]. The efficiency of the new algorithm is proved by comparison with the basic ABC and several other well known population based algorithms on extensive numerical test problems.

The rest of the paper is organized as follows. Section II reviews the fundamentals of the original ABC. Section III describes the proposed algorithm. Section IV presents comparative studies on benchmark functions and a slope stability analysis problem. Conclusions are given in Section V. Appendix A lists all test functions.

II. ARTIFICIAL BEE COLOBY ALGORITHM

ABC is a swarm intelligent optimization algorithm inspired by honey bee foraging [3-5]. 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

This work was supported by National Science Foundation for Post-doctoral Scientists of China and the State Key Program of National Natural Science of China (No. 90815024).

Corresponding author : Fei Kang (kangfei2009@163.com).

doi:10.4304/jsw.6.3.490-497

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 $\boldsymbol{x}_i$ (i=1,2,…,*NS*) is a n-dimensional vector.

In ABC, the fitness function is defined as follows:

$$fit_i = \begin{cases} \dfrac{1}{1+f_i} & f_i \geq 0 \\ 1+abs(f_i) & f_i < 0 \end{cases} \tag{2}$$

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,

$$p_i = fit_i \Big/ \sum_{j=1}^{NS} fit_j \tag{3}$$

A candidate solution $\boldsymbol{v}_i$ from the old solution $\boldsymbol{x}_i$ can be generated as

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{4}$$

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

$$x_{ij} = l_j + rand(0,1)\left(u_j - l_j\right) \tag{5}$$

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

III. THE PROPOSED ALGORITHM

A. Hooke-Jeeves Method

Hooke and Jeeves pattern search method is a simple yet very effective optimization technique proposed in 1961 [10]. Today, it is still a popular tool for various optimization problems, especially for deterministic local search.

The HJ method adopted here is modified according to the source code of Johnson [11]. 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. 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. Assume $\boldsymbol{x}_0$ is the current solution (the base point), $f_{\min}$ is the current minimum value of the objective function, $\boldsymbol{\delta}=(\delta_1, \delta_2,\ldots, \delta_n)$ is the step sizes of n directions. $\boldsymbol{x}_1$ is a temporary vector to store the obtained point after EM. The main steps of EM are described in Fig. 1.

1: Initialize i=1 and $\boldsymbol{x}_1=\boldsymbol{x}_0$.
2: $\boldsymbol{x}_{1i}=\boldsymbol{x}_{0i}+\delta_i$; if ($f(\boldsymbol{x}_1) < f_{\min}$), $f_{\min} = f(\boldsymbol{x}_1)$, go to step 4;else go to step 3.
3: $\boldsymbol{x}_{1i}=\boldsymbol{x}_{0i}-\delta_i$; 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 $\boldsymbol{x}_1$ is the result of EM and go to step 5.
5: If $f(\boldsymbol{x}_1)>=f_{\min}$, failure; else success.

Figure 1. Main steps of EM operator.

Given two solutions $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ ($f(\boldsymbol{x}_1) < f(\boldsymbol{x}_0)$), the PM takes the step $\boldsymbol{x}_1 - \boldsymbol{x}_0$ from $\boldsymbol{x}_0$ as

$$\boldsymbol{x}_2=\boldsymbol{x}_1+(\boldsymbol{x}_1-\boldsymbol{x}_0) \tag{6}$$

where $\boldsymbol{x}_2$ is the point obtained by 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 $\boldsymbol{x}_1-\boldsymbol{x}_0$ (since, if $f(\boldsymbol{x}_1) < f(\boldsymbol{x}_0)$, then $\boldsymbol{x}_1-\boldsymbol{x}_0$ is clearly a promising direction) [12]. The main steps of modified HJ method are shown in Fig. 2. ρ=0.5 is the step size reduction factor. An auxiliary step size s_a is adopted to judge when to stop the algorithm because a different step size for each variable is used.

1: Choose the starting point $\boldsymbol{x}_0$, the current step size δ_i (i=1, 2, …, n), the step size reduction factor $\rho<1$, the termination parameter $\varepsilon>0$. Initializing the iteration counter k=1, the auxiliary step size 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 6.
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 6.
7: If $s_a<\varepsilon$, 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.

Figure 2. Main steps of modified HJ algorithm.

B. ABC with Local Search

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 (2) to select a source for an onlooker bee. To improve the robustness of the selection strategy, rank-based fitness transformation is adopted as

$$fit_i = 2 - SP + \frac{2(SP-1)(p_i-1)}{NS-1} \tag{7}$$

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 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:

$$\delta_j = 0.1\frac{\sum_{i=1}^{m}\left(x'_{ij} - x_{best,j}\right)}{m} \tag{8}$$

where δ_j is the step size of the *jth* dimension, m is the number of solutions selected to calculate the step size , $\boldsymbol{x}'_i$ 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.

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 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 affect the function of the scout operator.

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 $\boldsymbol{v}_i$ for the employed bees by using (4) and evaluate them.
6: Apply the greedy selection process for the employed bees.
7: Rank the population and calculate the fitness by (7)
8: Calculate the probability p_i for the solutions $\boldsymbol{x}_i$ by (3).
9: Produce the new solutions $\boldsymbol{v}_i$ 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 $\boldsymbol{x}_i$.
12: Memorize the best solution $\boldsymbol{x}_{best}$ achieved so far.
(*Exploitation phase*)
13: If ((*cycle* mod *interval*)=0), calculate step size δ_j of HJ according to (8).
14: Call modified HJ with $\boldsymbol{x}_{best}$ as the base point until $s_a<\varepsilon$ and the obtained point is $\boldsymbol{x}_{best2}$.
15: If ($f(\boldsymbol{x}_{best2})\leq f(\boldsymbol{x}_{best})$) Replace the solution in the middle position after ranking by $\boldsymbol{x}_{best2}$ and set $\boldsymbol{x}_{best}$=$\boldsymbol{x}_{best2}$
16: If ($f(\boldsymbol{x}_{best})<f(\boldsymbol{x}_{best1})$) set $\boldsymbol{x}_{best1}$= $\boldsymbol{x}_{best}$ and k=0, else set k=k+1;
17: Set *cycle*=*cycle*+1.
(*Intensification search*)
18: If (k>*counter*) perform intensification search by modified HJ.
19: **Until** a termination condition is met.

Figure 3. Main steps of HJABC with intensification search.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A. Comparison with ABC, DE and ODE

A comprehensive set of benchmark functions including 32 different global optimization problems in [13] (Listed in appendix A) is adopted for comparison. The four algorithms are compared by measuring the number of function evaluations (*NFE*) to reach a given accuracy. Each of the experiments in this section is repeated 50 trials with different random seeds. The $NFE_{\max}$ is set as 300 000 in the comparison. The termination criterion is the *NFE* has reached the maximum value or the following condition is satisfied,

$$\left|f^* - \tilde{f}\right| < \varepsilon_1 \tag{9}$$

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^{-8} [13].

In order to compare the convergence speeds of ABC and HJABC, we use the acceleration rate (*AR*) which is defined as follows,

$$AR = \frac{NFE_{ABC}}{NFE_{HJABC}} \tag{10}$$

A trial is successful, if (9) can be satisfied before *NFE* reaches the maximum value.

The proposed new algorithm is compared with the basic ABC, DE and ODE [13] in terms of number of function evaluations, success rate (*SR*). The common parameters of ABC and HJABC are set as NS=25 (population size is 50), $Limit = NS \times n$ and NFE_{max}=200 000. The other parameters of HJABC are set as $interval = 3\times n$, $\varepsilon=10^{-3}$ and $counter=50n$. Each of the experiments in this section is repeated 50 trials.

The results of solving 32 benchmark functions are given in Table I and Table II. The best results of the *NFE* and *SR* for each function are highlighted in boldface. The AR_{ave} and the SR_{ave} on 32 test functions are shown in the last row of Table I and Table II.

TABLE I.
COMPARISON OF DIFFERENT ALGORITHMS IN TERMS OF NUMBER OF FUNCTION EVALUATIONS.

F	*n*	Number of function evaluations				*AR*
		DE	ODE	ABC	HJABC	
f_1	2	4324	4776	35753	**699**	51.15
f_2	2	4884	5748	3956	**1679**	2.36
f_3	2	1016	996	1081	**335**	3.23
f_4	2	8016	6608	5540	**1702**	3.25
f_5	2	3608	3288	35571	**476**	74.73
f_6	2	7976	5092	7171	**2940**	2.44
f_7	2	5352	4468	2406	**643**	3.74
f_8	2	10788	11148	18278	**2972**	6.15
f_9	3	3376	3580	3382	**630**	5.37
f_{10}	6	-	-	8377	**1851**	4.53
f_{11}	4	**16600**	19144	-	36230	-
f_{12}	4	**13925**	15280	-	17889	-
f_{13}	4	549850	311800	-	**114172**	-
f_{14}	4	-	-	12293	**2264**	5.43
f_{15}	4	-	-	17086	**2507**	6.82
f_{16}	4	4576	4500	25203	**2792**	9.03
f_{17}	10	191340	213330	32297	**31678**	1.02
f_{18}	10	328844	70389	16520	**15376**	1.07
f_{19}	20	177880	168680	-	**43939**	-
f_{20}	30	169152	98296	70463	**54497**	1.29
f_{21}	30	411164	337532	134313	**99686**	1.35
f_{22}	30	96488	53304	37623	**14351**	2.62
f_{23}	30	113428	69342	66269	**56855**	1.17
f_{24}	30	101460	70408	32825	**16685**	1.97
f_{25}	30	818425	199810	-	-	-
f_{26}	30	403112	-	-	**102718**	-
f_{27}	30	570290	72250	-	**120315**	-
f_{28}	30	187300	155636	38125	**12509**	3.05
f_{29}	30	87748	47716	44811	**18322**	2.45
f_{30}	30	41588	23124	**7876**	17755	0.44
f_{31}	30	25140	8328	13175	**5365**	2.46
f_{32}	30	385192	369104	-	**112118**	-
Ave.						6.36

As can be seen from Table I, HJABC converges much faster than ABC, DE and ODE in most cases. HJABC only converges slower than DE on Colville function (f_{11}), Kowalik function (f_{12})and slower than ABC on Step function (f_{30}). The overall average acceleration rate of HJABC to ABC is 6.36, which means HJABC is on average 536% faster than ABC. Except to two much large values of Beal function (f_1) and Matyas function (f_5), the average acceleration rate of HJABC to ABC is 2.30, which means HJABC is on average 130% faster than ABC except to two very large values. Generally speaking, ABC converges slower than DE and ODE for low dimensional problems, but the efficiency of ABC outperforms DE and ODE as the number of variables increases to 10.

The results reported in Table II show that there are 8~10 functions can not be solved 100% successfully by DE, ODE and ABC, whereas only 5 functions can not be solved 100% successfully by HJABC. On the other hand, HJABC performs better than other algorithms on 10 functions, whereas DE, ODE and ABC only perform better than other algorithms on 5~6 functions. The average success rate of HJABC is 0.93, which is much higher than the other three algorithms.

TABLE II.
COMPARISON OF DIFFERENT ALGORITHMS IN TERMS OF SUCCESS RATE.

F	*n*	Success rate			
		DE	ODE	ABC	HJABC
f_1	2	1	1	1	1
f_2	2	1	1	1	1
f_3	2	1	1	1	1
f_4	2	1	1	1	1
f_5	2	1	1	1	1
f_6	2	1	1	1	1
f_7	2	1	1	1	1
f_8	2	1	1	1	1
f_9	3	1	1	1	1
f_{10}	6	0	0	**1**	**1**
f_{11}	4	**1**	**1**	0	0.90
f_{12}	4	0.80	0.60	0	**1**
f_{13}	4	0.04	0.12	0	**0.48**
f_{14}	4	0	0	**1**	**1**
f_{15}	4	0	0	**1**	**1**
f_{16}	4	1	1	1	1
f_{17}	10	0.76	0.56	**1**	**1**
f_{18}	10	**1**	0.76	**1**	**1**
f_{19}	20	**1**	**1**	0	**1**
f_{20}	30	1	1	1	1
f_{21}	30	1	1	1	1
f_{22}	30	1	1	1	1
f_{23}	30	**1**	0.96	**1**	**1**
f_{24}	30	1	1	1	1
f_{25}	30	0.12	**1**	0	0
f_{26}	30	**1**	0	0	0.96
f_{27}	30	0.28	**0.88**	0	0.32
f_{28}	30	1	1	1	1
f_{29}	30	1	1	1	1
f_{30}	30	1	1	1	1
f_{31}	30	1	1	1	1
f_{32}	30	**1**	**1**	0	**1**
Ave.		0.81	0.81	0.75	0.93

B. The Effect of Parameter counter

Seven typical functions hard for ABC is selected to study the effect of parameter *counter*. The termination control parameters ε_1 is set equal to 10^{-6} and NFE_{max}=200 000. The other parameters are set the same as the previous section. The results are listed in Table III and Table IV.

TABLE III.
THE EFFECT OF PARAMETER *COUNTER* TO *NFE* AND *SR*

F	*n*	ABC	HJABC *counter*=2*n*	HJABC *counter*=50*n*	HJABC *counter*=200*n*
f_{11}	4	*NFE/SR*	*NFE/SR*	*NFE/SR*	*NFE/SR*
f_{26}	30	-/0.00	7273/1.00	13391/1.00	13114/1.00
f_{32}	30	-/0.00	64422/0.98	62634/0.98	65635/1.00
f_{19}	30	-/0.00	90796/1.00	89880/1.00	89040/1.00
f_{13}	4	-/0.00	61941/1.00	63354/1.00	61262/1.00
f_{12}	4	-/0.00	11104/1.00	46631/1.00	35051/0.92
f_{36}	5	-/0.00	3503/0.88	6196/1.00	6093/1.00
f_{11}	4	-/0.00	10682/0.92	15700/1.00	13461/0.98

The average *NFE* of successful runs and the *SR* are listed in Table III. 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 IV.
THE EFFECT OF PARAMETER *COUNTER* TO ACCURACY

F	*n*	ABC	HJABC *counter*=2*n*	HJABC *counter*=50*n*	HJABC *counter*=200*n*
		Mean(SD)	Mean(SD)	Mean(SD)	Mean(SD)
f_{11}	4	1.80e-1 (1.02e-1)	6.21e-14 (3.34e-13)	4.51e-9 (1.65e-8)	1.63e-8 (1.04e-7)
f_{26}	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)
f_{32}	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)
f_{19}	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)
f_{13}	4	2.47e-3 (3.12e-3)	9.06e-11 (5.88e-10)	1.43e-7 (2.77e-7)	4.16e-7 (9.58e-7)
f_{12}	4	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_{36}	5	2.38e-1 (3.86e-1)	5.67e-1 (2.2655)	2.67e-10 (1.78e-9)	1.77e-9 (8.79e-9)

The average best values (mean) and the standard deviation (SD) of 50 trials are listed in Table Ⅳ. It can be seen that the accuracy of HJABC is much higher than ABC on functions having narrow curving valley (Rosenbrock, Colville) and functions with high eccentric ellipse (Zakharov, Schwefel's problem 1.2).

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. In most cases, *counter* can be set as a very large value and the intensification search even can be omitted.

C. Comparison with Particle Swarm Optimization

In this section, the proposed algorithm is compared with particle swarm optimization (PSO) and comprehensive learning PSO (CLPSO) [14]. Two unimodal functions and six multimodal benchmark functions in [14] (Listed in appendix A) are adopted, considering CLPSO is mainly proposed for multimodal functions. The mean values and standard deviation (SD) of the results are presented to compare the four algorithms. All functions are tested on 10 and 30 dimensions. The maximum fitness evaluations is set at 30 000 when solving the 10-dimensional problems and 200 000 when solving the 30-dimensional problems [14]. All experiments were run 30 times.

The results for *n*=10 is shown in Table Ⅴ. The best results among the four algorithms are shown in bold. From the results, it can be seen that ABC performs much better than PSO on seven functions except to the Sphere function. CLPSO performs better than other algorithms on 5 functions and HJABC performs better than other algorithms on 4 functions. But the result of Rosenbrock function obtained by HJABC is much better than the other three algorithms.

TABLE V.
RESULTS FOR 10-D PROBLEMS

F	PSO-*w* Mean (SD)	CLPSO Mean (SD)	ABC Mean (SD)	HJABC Mean (SD)
f_{29}	**7.96e-51 (3.56e-50)**	5.15e-29 (5.15e-29)	1.34e-23 (2.74e-23)	1.25e-39 (2.62e-39)
f_{26}	3.08e+00 (7.69e-01)	2.46e+00 (1.70e+0)	2.69e-01 (4.17e-01)	**1.67e-03 (1.92e-03)**
f_{20}	1.58e-14 (1.60e-14)	4.32e-14 (2.55e-14)	3.59e-12 (5.92e-12)	**1.30e-14 (4.15e-15)**
f_{23}	9.69e-02 (5.01e-02)	**4.56e-03 (4.81e-03)**	5.26e-03 (6.57e-03)	6.85e-03 (6.66e-03)
f_{35}	2.28e-03 (7.04e-03)	**0 (0)**	1.89e-15 (4.73e-15)	8.29e-16 (1.53e-15)
f_{18}	5.82e+00 (2.96e+0)	**0 (0)**	**0 (0)**	**0 (0)**
f_{33}	4.05e+00 (2.58e+0)	**0 (0)**	**0 (0)**	**0 (0)**
f_{34}	3.20e+02 (1.85e+2)	**0 (0)**	7.42e-09 (2.51e-08)	1.24e-12 (1.18e-12)

The results for *n*=30 is shown in Table Ⅵ. The results show that ABC and HJABC perform better than PSO and CLPSO on the 30-D problems. They perform better than other algorithms on 4 and 5 functions respectively. The result of Rosenbrock function obtained by HJABC is still much better than the other three algorithms.

TABLE VI.
RESULTS FOR 30-D PROBLEMS

F	PSO-*w* Mean (SD)	CLPSO Mean (SD)	ABC Mean (SD)	HJABC Mean (SD)
f_{29}	9.78e-30 (2.50e-29)	4.46e-14 (1.73e-14)	3.37e-59 (7.31e-59)	**6.74e-72 (2.11e-71)**
f_{26}	2.93e+01 (2.51e+1)	2.10e+01 (2.98e+00)	1.56e-01 (2.18e-01)	**1.31e-08 (7.21e-08)**
f_{20}	3.94e-14 (1.12e+0)	**0 (0)**	3.43e-14 (3.20e-15)	3.78e-14 (4.04e-15)
f_{23}	8.13e-03 (7.16e-03)	**3.14e-10 (4.64e-10)**	3.02e-4 (2.14e-3)	3.94e-4 (2.01e-3)
f_{35}	1.30e-04 (3.30e-04)	3.45e-07 (1.94e-07)	**0 (0)**	9.47e-16 (2.46e-15)
f_{18}	2.90e+01 (7.70e+0)	4.85e-10 (3.63e-10)	**0 (0)**	**0 (0)**
f_{33}	2.97e+01 (1.39e+1)	4.36e-10 (2.44e-10)	**0 (0)**	**0 (0)**
f_{34}	1.10e+03 (2.56e+2)	1.27e-12 (8.79e-13)	**1.21e-12 (9.94e-13)**	**1.21e-12 (8.72e-13)**

By analyzing the results on 10-D and 30-D problems, we can conclude that PSO only perform well on the sphere function, CLPSO and ABC perform well on all the functions except to the Rosenbrock function. HJABC performs the best, because it can solve all the eight problems well and it performs better than other algorithms on 9 scenarios whereas ABC and CLPSO only perform better than other algorithms on 6~7 scenarios.

D. Application on Slope Stability Analysis

Determination of the critical slip surface and the corresponding factor of safety is the central issue to slope stability analysis. Several methods such as Monte Carlo technique [15] and genetic algorithm [16] has been

proposed for searching the critical circular slip surface. The objective function for this problem can be stated as

$$\min\ F(S),\ S=(x,y,d) \tag{11}$$

where F is the factor of safety, x, y are the coordinates of the center of the circular slip surface, $d = y - r$ is bottom of the slip surface and r is the radius.

A complex slope shown in Fig. 4 is taken as an example. The parameters of soil layers are shown in Table Ⅶ. The factor of safety in (11) is calculated by the ordinary method.

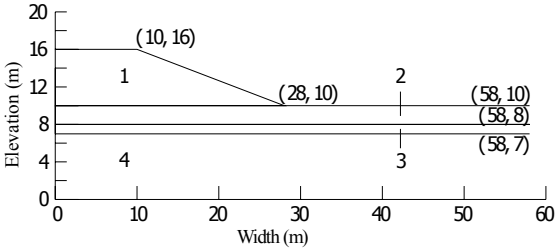

Figure 4. Cross section of the complex slope.

TABLE VII.
MECHANICAL PARAMETERS OF SOIL LAYERS

Layer	Internal friction φ (°)	Cohesion c (kN·m^{-2})	Unit weight γ (kN·m^{-3})
1	35.0	0.0	19.22
2	32.5	8.0	19.60
3	0.0	10.5	18.50
4	30.0	10.0	19.80

The search range is set as $x \in [10,100]$, $y \in [15,100]$, $d \in$ [-14,16] The *NFE* is set as 10000. The statistic results during 30 runs of ABC and HJABC are shown in Table Ⅷ. The mean best objective function values of ABC and HJABC are shown in Fig. 5.

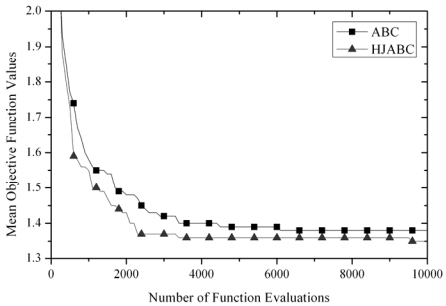

Figure 5. Variation of mean best objective function values with number of function evaluations.

TABLE VIII.
RESULTS FOR THE SLOPE STABILITY ANALYSIS PROBLEM

Method	Factor of safety				Position of critical circle (m)		
	Min.	Max.	Mean	SD	x	y	r
ABC	1.351	1.416	1.375	1.188e-2	21.060	19.443	12.438
HJABC	1.351	1.369	1.355	7.021e-3	21.057	19.455	12.451

From Table VIII and Fig.5 it can be seen that both ABC and HJABC can find the minimum factor of safety. However, HJABC performs better than ABC in the problem because it is more robust and has higher accuracy. A efficient global optimization algorithm is proposed for critical slip surface searching of slope stability analysis.

V. CONCLUSION

An artificial bee colony algorithm with local search is proposed for global numerical optimization problems. HJ can enrich the search directions of ABC and accelerate the search process. The performance of ABC on functions have narrow curving valley (for example Rosenbrock function, Colville function) and functions with high eccentric ellipse (for example Zakharov function, Schwefel's problem 1.2) can be much improved by incorporate the pattern move. The proposed method has been tested on 36 benchmark mathematical functions and a slope stability analysis problem. When compared with the basic ABC and several most recent algorithms, the proposed new method has demonstrated strongly competitive results.

APPENDIX A LIST OF TEST FUNCTIONS

1. Beale function (f_1)

$$f(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$$

with $-4.5 \le x_1, x_2 \le 4.5$, min ($f$) =$f$(3,0.5)=0.

2. Branin RCOS function (f_2)

$$f(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$$

with $-5 \le x_1, x_2 \le 15$,

min(f)= $f(-\pi,12.275)/(\pi,2.275)/(3\pi,2.475)=0.397887$

3.De Jong's function 4 (f_3)

$$f(x)=\sum_1^n ix_i^4$$

with $-1.28 \le x_i \le 1.28$, min (f) =f(0,…,0)=0.

4. Easom function (f_4)

$$f(x)=-\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$$

with $-10 \le x_1, x_2 \le 10$, min (f) = $f(\pi,\pi)$=-1.

5. Matyas function (f_5)

$$f(x)=0.26(x_1^2+x_2^2)-0.48x_1x_2$$

with $-10 \le x_1, x_2 \le 10$, min (f) = f(0,0)=0.

6. Schaffer's function 6 (f_6)

$$f(x)=0.5+\frac{\sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{(1+0.01(x_1^2+x_2^2))^2}$$

with $-10 \le x_1, x_2 \le 10$, min (f) = f(0,0)=0.

7. Six Hump Camel Back function (f_7)

$$f(x)=4x_1^2-2.1x_1^4+1/3x_1^6+x_1x_2-4x_2^2+4x_2^4$$

with $-5 \le x_1, x_2 \le 5$,

min (f) = f(0.089842,-0.712656)/ (-0.089842, 0.712656)=0.

8. Tripod function (f_8)

$$f(x)=p(x_2)(1+p(x_1))+\left|(x_1+50p(x_2))(1-2p(x_1))\right|$$

$$+\left|(x_2+50(1-2p(x_2)))\right|$$

with $-100 \le x_1, x_2 \le 100$, min $(f) = f(0,-50) = 0$

where $p(x)=1$ for $x \ge 0$, otherwise, $p(x)=0$.

9. Hartman 3 function (f_9)

$$f(x)=\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j-p_{ij})^2\right)$$

with $0 \le x_i \le 1$,

min $(f) = f(0.114614, 0.555649, 0.852547) = -3.862782$

The coefficients can be seen in [13].

10. Hartman 6 function (f_{10})

$$f(x)=\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2\right)$$

with $0 \le x_i \le 1$,

min $(f) = f(0.20169, 0.15001, 0.47687, 0.27533, 0.31165, 0.65730) = -3.322368$.

The coefficients can be seen in [13].

11. Colville function (f_{11})

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

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

12. 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].

13. Perm function (f_{13})

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

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

14,15,16. Shekel's Family (f_{14}, f_{15}, f_{16})

$$f(x)=-\sum_{j=1}^{m}\frac{1}{\sum_{i=1}^{4}(x_j-a_{ij})^2+c_i}$$

with $0 \le x_i \le 10$, m=5, min $(f_{14}) = f_{14}(4,4,4,4) = -10.1532997$

m=7, min $(f_{15}) = f_{15}(4,4,4,4) = -10.4029406$

m=10, min $(f_{16}) = f_{16}(4,4,4,4) = -10.5364098$.

17. Michalewicz function (f_{17})

$$f(x)=-\sum_{i=1}^{n}\sin(x_i)(\sin(ix_i^2/\pi))^{2m},\ m=10$$

with $0 \le x_i \le \pi$, min $(f_{n=10}) = -9.66015171$.

18. Rastrigin function (f_{18})

$$f(x)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$$

with $-5.12 \le x_i \le 5.12$, min $(f) = f(0,0,\ldots,0) = 0$.

19. Schwefel problem 1.2 (f_{19})

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

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

20. Ackley function (f_{20})

$$f(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$$

with $-32 \le x_i \le 32$, min $(f) = f(0,0,\ldots,0) = 0$.

21. Alpine function (f_{21})

$$f(x)=\sum_{i=1}^{n}\left|x_i\sin(x_i)+0.1x_i\right|$$

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

22. Axis parallel hyperellipsoid (f_{22})

$$f(x)=\sum_{i=1}^{n} i x_i^2$$

with $-5.12 \le x_i \le 5.12$, min $(f) = f(0,0,\ldots,0) = 0$.

23. Griewank function (f_{23})

$$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

with $-600 \le x_i \le 600$, min $(f) = f(0,0,\ldots,0) = 0$.

24. Levy and Montalvo problem 2 (f_{24})

$$f(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]\right.$$
$$\left.+(x_n-1)^2[1+\sin^2(2\pi x_n)]\right\}$$

with $-5 \le x_i \le 5$, min $(f) = f(1,1,\ldots,1) = 0$.

25. Quartic function (f_{25})

$$f(x)=\sum_{i=1}^{n} i x_i^4+random[0,1)$$

with $-1.28 \le x_i \le 1.28$, min $(f) = f(0,0,\ldots,0) = 0$.

26. Rosenbrock function (f_{26})

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

with $-30 \le x_i \le 30$, min $(f) = f(0,0,\ldots,0) = 0$.

27. Schwefel problem 2.21 (f_{27})

$$f(x)=\max\left\{|x_i|,\ 1 \le i \le n\right\}$$

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

28. Schwefel problem 2.22 (f_{28})

$$f(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|$$

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

29. Sphere function (f_{29})

$$f(x)=\sum_{i=1}^{n}x_i^2$$

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

30. Step function (f_{30})

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

with $-100 \le x_i \le 100$, min $(f) = f(-0.5 \le x_i < 0.5) = 0$.

31. Sum of different power function (f_{31})

$$f(x)=\sum_{i=1}^{n}|x_i|^{(i+1)}$$

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

32. Zakharov function (f_{32})

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

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

33. Noncontinuous Rastrigin function (f_{33})

$$f(x)=10n+\sum_{i=1}^{n}(y_i^2-10\cos(2\pi y_i))$$

with $-5.12 \le x_i \le 5.12$, min $(f) = f(0,0,\ldots,0) = 0$ where

$$y_i = \begin{cases} x_i & |x_i| < 1/2 \\ round(2x_i)/2 & |x_i| \geq 1/2 \end{cases}$$

34. Schwefel's problem 2.26 (f_{34})

$f(x) = 418.9829n + \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|})$

with $-500 \leq x_i \leq 500$, min (f) = f(420.9687,…,420.9687) =0

35. Weierstrass function (f_{35})

$$f(x) = \sum_{i=1}^{n} \left\{ \sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right) \right] \right\} - n \sum_{k=0}^{k\max} \left(a^k \cos(\pi b^k) \right)$$

, a=0.5, b=3, k_{max}=30

with $-0.5 \leq x_i \leq 0.5$, min $(f) = f(0,0,\ldots,0) = 0$

36. Fletch-Powell Problem (f_{36})

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

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

with $-\pi \leq x_i \leq \pi$, min (f) =0, all details can be found in [9].

ACKNOWLEDGMENT

This work was partly supported by the National Science Foundation for Post-doctoral Scientists of China and the State Key Program of National Natural Science of China (No. 90815024).

REFERENCES

[1] A. Georgieva and I. Jordanov, "Global optimization based on novel heuristics, low-discrepancy sequences and genetic algorithms," *Eur. J. Oper. Res.*, vol. 196, pp. 413–422, July 2009.

[2] A. Baykasoglu, L. Ozbakir, and P. Tapkan, "Artificial bee colony algorithm and its application to generalized assignment problem," In Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, ITech Education and Publishing, Vienna, Austria, pp. 113–144, 2007.

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

[4] 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.

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

[6] N. Karaboga, "A new design method based on artificial bee colony algorithm for digital IIR filters," *J. Franklin I.*, vol. 346, pp. 328–348, May 2009.

[7] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," *Appl. Soft Comput.*, vol. 9, pp. 625–631, March 2009.

[8] 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.

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

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

[11] M. G. Johnson, "Nonlinear Optimization using the algorithm of Hooke and Jeeves," Available: http://www.netlib.org/opt/hooke.c.

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

[13] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition- based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, pp. 64-79, February 2008.

[14] J. J. Liang, A. K. Qin, and P. N. Suganthan, "Comprehensive Learning particle swarm optimizer for global optimization of multimodal functions", *IEEE Trans. Evol. Comput.*, vol. 10, pp. 281-295, June 2006.

[15] A. L. Husein Malkawi, W. F. Hassan, and S. K. Sarma, "An efficient search method for finding the critical circular slip surface using the Monte Carlo technique" *Can. Geotech. J.*, vol. 38, pp. 1081–1089, 2001.

[16] P. McCombie, P. Wilkinson, "The use of the simple genetic algorithm in finding the critical factor of safety in slope stability analysis", *Comput. Geotech.*, vol. 29, pp. 699–714, December 2002.

Fei Kang received the B. Sc. and Ph. D. degrees both in infrastructure engineering from the Dalian University of Technology, Dalian, China, in 2003 and 2009 . He is currently working as a postdoctor at Dalian University of Technology. His current research interests are evolutionary algorithms, swarm intelligence and their application in geotechnical engineering. He has published more than 10 papers over the past five years.