Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 232704, 10 pages
http://dx.doi.org/10.1155/2014/232704

Research Article

An Improved Artificial Bee Colony Algorithm Based on Balance-Evolution Strategy for Unmanned Combat Aerial Vehicle Path Planning

Bai Li,[1] Li-gang Gong,[2] and Wen-lun Yang[1]

[1] *School of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China*
[2] *School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China*

Correspondence should be addressed to Bai Li; libai@zju.edu.cn

Received 4 December 2013; Accepted 19 February 2014; Published 20 March 2014

Academic Editors: Z. Chen and F. Yu

Unmanned combat aerial vehicles (UCAVs) have been of great interest to military organizations throughout the world due to their outstanding capabilities to operate in dangerous or hazardous environments. UCAV path planning aims to obtain an optimal flight route with the threats and constraints in the combat field well considered. In this work, a novel artificial bee colony (ABC) algorithm improved by a balance-evolution strategy (BES) is applied in this optimization scheme. In this new algorithm, convergence information during the iteration is fully utilized to manipulate the exploration/exploitation accuracy and to pursue a balance between local exploitation and global exploration capabilities. Simulation results confirm that BE-ABC algorithm is more competent for the UCAV path planning scheme than the conventional ABC algorithm and two other state-of-the-art modified ABC algorithms.

1. Introduction

Developments in automated and unmanned flight technologies have become an irresistible trend in many countries. As a matter of fact, unmanned combat aerial vehicles (UCAVs) have been of great importance to many military organizations throughout the world due to their capabilities to work in remote and hazardous environments [1]. Path planning is a critical aspect of the autonomous control module in UCAV, which aims to provide an optimal path from the starting point to the desired destination with the artificial threats and some natural constraints considered.

For the UCAV path planning scheme, an optimal solution corresponds to one path that minimizes the flight route, average altitude, fuel consumption, exposure to radar or artillery, and so forth [2]. With the development of the ground defending weapons, the difficulty of describing these artificial threats significantly becomes larger. Therefore, in order to deal with the increasing complexity when modeling a combat field, researchers have gradually shifted their interests away from deterministic algorithms [3–5].

Like most real-world optimization problems, finding the global optimum is enormously difficult. To avoid enumerating for the global optimums, evolutionary algorithms (EAs) have been well investigated and developed as a primary branch of the heuristic algorithms, such as genetic algorithm (GA) [6], differential evolution algorithm (DE) [7], ant colony optimization algorithm (ACO) [8], particle swarm optimization algorithm (PSO) [9], and artificial bee colony algorithm (ABC) [10]. Although these intelligent algorithms do not necessarily guarantee global convergence, some satisfying results can be acquired after all. That is why studies have been brought on developing new algorithms or modifying the existing ones in recent years [11]. For this UCAV path planning scheme, algorithms such as chaos theory based ABC (C-ABC) [12], immune GA (I-GA) [13], PSO [14], quantum-behaved PSO (Q-PSO) [15], master-slave parallel vector-evaluated GA (MPV-GA) [16], and intelligent water drops algorithm (IWD) [17] have been applied.

ABC algorithm is a swarm intelligence algorithm motivated by the foraging behaviors of bee swarms. In this algorithm, the bee swarm mainly consists of three components: the employed bees, the onlooker bees, and the scout bees [18]. Employed bees start the global exploration, then the qualified employed bees will be capable of attracting the onlooker bees to follow. At this point, following means exploiting locally around an employed bee. The qualification of each employed bee is determined by the roulette selection strategy. In the long run of the iterations, those unqualified employed bees eventually perish and scout bees will take their places.

Applications of ABC algorithm span the areas of image processing [19], structure identification [20], bioinformatics [21], neural network training [22], and so forth. At the same time, it is believed that ABC algorithm works well in the global exploration but poorly in the local exploitation [23]. Generally speaking, two prevailing ways have been taken to improve the conventional ABC algorithm. In the first way, strategies, for instance, Rosenbrock's rotational direction strategy [24], quantum theory [25], chaos theory [26], and Boltzmann selection strategy [27] from the outside world are introduced. The second way mainly focuses on combining ABC algorithm with some other intelligent algorithms. DE-ABC [28], PSO-ABC [29], and QEA-ABC [30] are typical examples. Apart from the two ways mentioned above, efforts have also been made on revising the crossover and mutation equations in the conventional ABC algorithm [21, 31, 32]. Especially, an improved ABC algorithm named I-ABC has shown fast convergence speed and accurate convergence precision in comparison with ABC algorithm and is regarded as a state-of-the-art version of ABC [32].

Viewing improvements ever made for ABC algorithm, attentions have seldom been paid to fully utilizing the convergence messages hiding in the iteration system [33, 34]. In other words, apart from the roulette selection strategy, the convergence status of a previous cycle may be regarded as feedback information to guide a subsequent cycle. In addition, it is noted that the generation of scout bees mainly intends for the escapement of premature convergence but scout bees are confirmed to be noneffective in some numerical cases [35]. Therefore, new rules may be needed to guide the scout bees so as to perform more efficiently. Moreover, it is believed that the exploration and the exploitation procedures need to match with each other so as to cooperate efficiently. That is, it is essential for an intelligent algorithm to capture a balance between global exploration and local exploitation.

In this paper, a novel ABC algorithm modified by a balance-evolution strategy is applied for this path planning scheme. In this new algorithm (which is named BE-ABC), convergence status in the iteration will be fully utilized so as to manipulate the exploration/exploitation accuracy and to make a tradeoff between local exploitation and global exploration. Besides, the rule guiding the scout bees is modified according to an overall degradation procedure. This work intends for some intensive research to evaluate the performance of BE-ABC algorithm in this UCAV path planning scheme, in comparison with two recent state-of-the-art modifications of ABC.

The remainder of this paper is organized as follows. In Section 2, the mathematical model of the combat field is given. In Section 3, the principles of four versions of ABC are introduced in detail. Simulations and the corresponding results are shown in Section 4, together with some remarks. Conclusions are drawn in the last section.

2. Combat Field Modeling for UCAV Path Planning

There are some threatening installations in the combat field, for instance, missiles, radars, and antiaircraft artilleries. The effects of such installations are presented by circles in the combat field of different radiuses and threat weights [36]. If part of its path falls in a circle, an UCAV will be vulnerable to the corresponding ground defense installation. To be more precise, the damage probability of an UCAV is proportional to its distance away from the threat center. Additionally, when the flight path is outside a circle, the probability of being attacked is 0. Let us define the starting point as S and the terminal point as T (see Figure 1). The UCAV flight mission is to calculate an optimal path from S to T, with all the given threat regions in the combat field and the fuel consumption considered.

To make this problem more concrete, let us draw a segment ST connecting the starting and terminal points first. Afterwards, ST is divided into $(D+1)$ equal portions by D vertical dash lines $\{L_k, k = 1, 2, \ldots, D\}$ as plotted in Figure 1. These lines are taken as new axes; then as many as D points (see the small rectangles in Figure 1) along such axes will be connected in sequence to form a feasible path from S to T. In this sense, the D corresponding coordinates Z_k ($k = 1, 2, \ldots, D$) are the variables to be optimized so as to acquire an optimal flight path.

To accelerate the processing speed, it is encouraged to take the ST direction as the x axis [37]. In this way, point movement on the L_k can be described more easily. Therefore, the first step before the computation of the optimal flight path is the transfer of axes. Any point (α_i, β_i) on the original combat field gets transferred to (α_i^*, β_i^*) in the new axes as defined in (1), where θ denotes the angle between the original x axis and the original ST direction, as follows:

$$\begin{bmatrix} \alpha_i^* \\ \beta_i^* \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}. \quad (1)$$

Regarding the evaluation of one candidate flight path, the threat cost J_{threat} and the fuel consumption J_{fuel} are taken into consideration [1], as shown in

$$\begin{aligned} J &= \lambda \cdot J_{\text{fuel}} + (1-\lambda) \cdot J_{\text{threat}} \\ &= \lambda \cdot \int_0^{\text{length}} w_{\text{threat}}\, dl + (1-\lambda) \cdot \int_0^{\text{length}} w_{\text{fuel}}\, dl, \end{aligned} \quad (2)$$

where J is the weighted sum of flight cost for this flight path, $\lambda \in [0,1]$ represents the weighting parameter, w_{threat} and w_{fuel} are variables related to every instantaneous position on the path, and length denotes the total length of this candidate flight path.

To simplify the integral operations, the flight cost from the point along L_i to the one along L_{i+1} is calculated at five sample points [38], as shown in Figure 2.

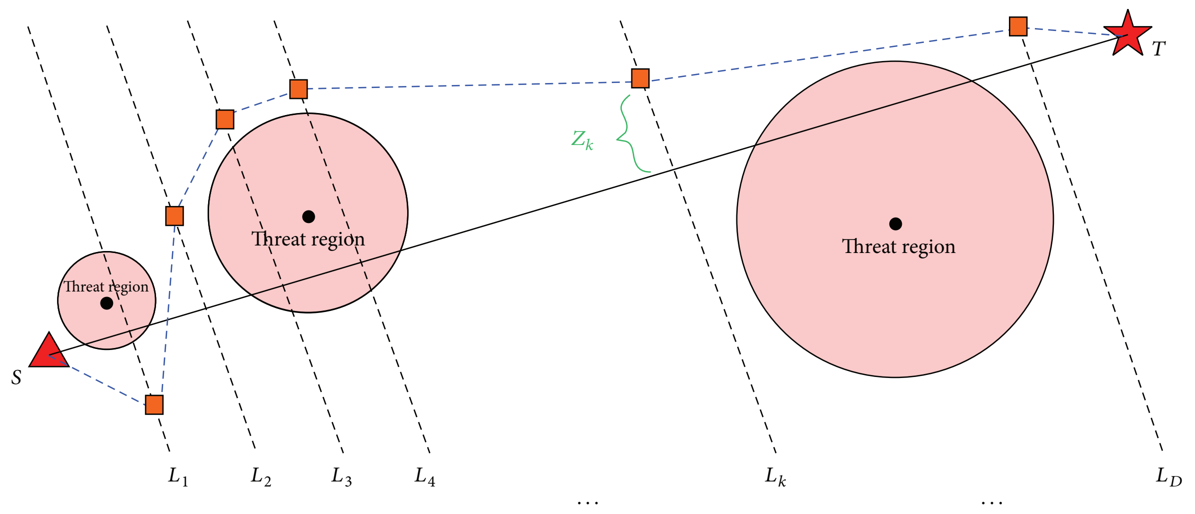

FIGURE 1: Schematic diagram of combat field modeling.

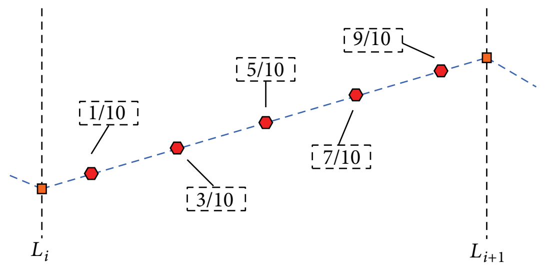

FIGURE 2: Schematic diagram of flight cost computation.

If the flight path shown above falls into a threat region, the threat cost is calculated as follows:

$$w_{\text{threat},L_i \to L_{i+1}} = \frac{\text{length}_i}{5} \cdot \sum_{k=1}^{N_t} \left[t_k \cdot \left(\frac{1}{d^4_{0.1,i,k}} + \frac{1}{d^4_{0.3,i,k}} + \frac{1}{d^4_{0.5,i,k}} + \frac{1}{d^4_{0.7,i,k}} + \frac{1}{d^4_{0.9,i,k}} \right) \right], \quad (3)$$

where N_t denotes the number of threatening circles that the path involves in, length_i refers to the ith subpath length, $d^4_{0.1,i,k}$ stands for the distance between the $1/10$ point on the path and the kth threat center, and t_k is regarded as the threat grade of the kth threat. In this work, it is assumed that the velocity of the UCAV is a constant. Then the fuel consumption J_{fuel} is considered in direct proportion to length (i.e., $w_{\text{fuel}} \equiv 1$).

3. Principles of ABC Relevant Algorithms

The preceding section holds a description of combat field model. The optimal vector $\mathbf{z} = [z_1, z_2, \ldots, z_D]$ is expected to derive using these intelligent algorithms that will be introduced in detail in this section. The conventional ABC algorithm and three state-of-the-art versions of ABC are applied for the UCAV flight path optimization scheme.

3.1. Review of Conventional ABC Algorithm.

In ABC, three kinds of bees cooperate to search for the optimal nectar sources in the space, namely, the employed bees, the onlooker bees, and the scout bees [18].

Let $\mathbf{X}_i = (X_i^1, X_i^2, \ldots, X_i^D)$ represent a candidate position searched by the ith employed bee in the space ($i = 1, \ldots, SN$). Here, a position refers to a feasible solution for the optimization problem. The number of employed bees is set to SN. In each iteration, onlooker bees search locally around those qualified employed bees. The number of onlooker bees is commonly set to SN. Here, the qualification standard concerns the roulette selection strategy and will be introduced later in this section. Those employed bees who cannot make any progress for some time will die out and the randomly initialized scout bees will take their places.

At first, all the SN employed bees need to be randomly initialized in the feasible solution space. In detail, the jth element of the ith solution $\mathbf{X}_i$ is initialized using

$$X_i^j \longleftarrow X_{\min}^j + \text{rand}(0,1) \cdot \left(X_{\max}^j - X_{\min}^j \right), \quad j = 1, 2, \ldots, D, \quad (4)$$

where $X_{\min}^j$ and $X_{\max}^j$ denote the lower and upper boundaries of this jth element and D denotes the dimension of any feasible solution.

In each cycle of the iterations, an employed bee executes a crossover and mutation process to share information with one randomly chosen companion and search in the new position X^{*j}_i utilizing

$$X^{*j}_i \longleftarrow X_i^j + \text{rand}(-1,1) \cdot \left(X_k^j - X_i^j \right). \quad (5)$$

In this equation, the ith employed bee exchanges information with the kth employed bee in the jth element. Here, j is a randomly selected integer from 1 to D. Similarly, k is a randomly selected integer from 1 to SN while it satisfies the condition that $k \neq i$.

After such crossover and mutation process for the employed bees, the greedy selection strategy will be implemented. If $\mathbf{X}_i^*$ is better (i.e., its corresponding objective function value is lower), the previous position $\mathbf{X}_i$ is discarded; otherwise, the employed bee remains at $\mathbf{X}_i$.

Afterwards, an index named P is calculated as the qualification reflection for each of the employed bees according to

$$P_i = \frac{\text{fitness}(i)}{\sum_{j=1}^{SN} \text{fitness}(j)},$$
$$\text{fitness}(i) = \begin{cases} \dfrac{1}{1 + \text{obj}(i)} & \text{if obj}(i) \geq 0 \\ 1 + \text{abs}(\text{obj}(i)) & \text{if obj}(i) < 0. \end{cases} \tag{6}$$

Here, obj(i) denotes the objective function value of the position that the ith employed bee currently stays in.

Each of the onlooker bees needs an employed bee to follow. In this case, if $P_1 \geq \text{rand}(0,1)$, the 1st employed bee is chosen for the specific onlooker bee; otherwise, a similar comparison between P_2 and rand(0, 1) will be carried on. If all the P_i are smaller than rand(0, 1), such process goes over again until one employed bee is selected. In this way, SN onlooker bees determine the corresponding employed bees to follow. In this sense, to follow means to search around locally using

$$Y^{*j}_0 \longleftarrow X_0^j + \text{rand}(-1,1) \cdot \left(X_k^j - X_0^j\right). \tag{7}$$

In this equation, the kth employed bee and the jth element are still randomly chosen. Again, the greedy selection strategy is implemented. If the position searched by the onlooker bee is more qualified than the position by the employed bee (i.e., if $\text{obj}(\mathbf{Y}_0^*) < \text{obj}(\mathbf{X}_0)$), the employed bee directly moves to the better place.

During the iteration, once an employed bee searches globally but finds no better position, or once an onlooker bee exploits around an employed bee without finding a better position, the invalid trail time *trial* adds one. On the other hand, when any better position can be found for the ith employed bee, the corresponding *trial*(i) is set to zero instantly. At the end of each iteration, it is necessary to check whether any *trial*(i) exceeds a certain threshold named Limit. If *trail*(i) > Limit, the ith employed bee will be directly replaced by a scout bee. A scout bee simply refers to a randomly initialized position in the food source using (4).

3.2. Principle of I-ABC Algorithm. I-ABC algorithm was proposed by Li et al. in 2012 [32]. It differs from the conventional ABC merely in the crossover and mutation equations. Consider

$$X^{*j}_i \longleftarrow X_i^j \cdot \phi(i) + \left(X_k^j - X_i^j\right) \cdot \text{rand}(-1,1) \cdot \phi(i) + \left(\text{global}^j - X_i^j\right) \cdot \text{rand}(0,1), \tag{8}$$

$$Y^{*j}_0 \longleftarrow X_0^j \cdot \phi(i) + \left(X_k^j - X_0^j\right) \cdot \text{rand}(-1,1) \cdot \phi(i) + \left(\text{global}^j - X_0^j\right) \cdot \text{rand}(0,1) \cdot \phi(i), \tag{9}$$

$$\phi(i) = \frac{1}{1 + \exp\left(\left(-\text{fitness}(i)/ap\right)^{\text{iter}}\right)}, \tag{10}$$

$$ap \equiv \text{fitness}(1)|_{\text{iter}\equiv 1}. \tag{11}$$

Equation (8) is designed for global exploration and replaces (5) as in the conventional ABC. Similarly, (9) replaces (7) for the exploitation process. In (10) and (11), global^j denotes the jth element of the best position ever emerged in the cycles of iteration. It is notable that, as in (11), ap is randomly determined by the initialization conditions in the first iteration and is usually a relatively small positive number.

3.3. Principle of IF-ABC Algorithm. IF-ABC algorithm mainly differs from the conventional ABC algorithm in the utilization of *trial* as the internal feedback information and in the abandonment of the roulette selection strategy [21].

Before the iteration process, as many as SN employed bees are randomly sent out to explore in the nectar source space. Particularly, the process to initialize the jth element of the ith solution $\mathbf{X}_i$ is described in (4). Afterwards, the iteration process starts.

In each cycle of iteration, an employed bee executes a crossover and mutation procedure to share information with its one (randomly selected) companion. This procedure is expressed in (5). Then, the greedy selection strategy is implemented so as to select the better position between X^{*j}_i and X_i^j (i.e., to select the one with a relatively higher objective function value). Different from that in the conventional ABC algorithm, the number of elements involved in such crossover and mutation procedure is considered flexible. In other words, (5) should implement on each of the employed bees for *trial*(i) times, where *trial*(i) $\in [1, D]$ and will be discussed later. Then the searching procedure by the employed bees in this current cycle is completed, which is usually regarded as the global exploration procedure.

Afterwards, onlooker bees take over the searching process. In the IF-ABC algorithm, each of the employed bees is given a chance to be followed by an onlooker regardless they are "qualified" or not, pursuing to bring about more chances (i.e., more dynamics and diversity) for evolution and to fight against premature convergence. However, qualifications of the employed bees should be taken into account after all. IF-ABC algorithm seeks a new way to evaluate the searching performance.

Now that the roulette selection strategy is discarded in IF-ABC, the onlookers will directly choose their corresponding

```
(1) initialize solution population using (4)
(2) set $trial(i) = 1$ $(i = 1, 2, \ldots, SN)$
(3) for $iter = 1 : \mathrm{MCN}$, do
(4)     for $item = 1 : SN$, do
(5)         crossover and mutate using (14) in as many as $trial(item)$ randomly selected
            elements for the $item$-th employed bee
(6)         adopt greedy selection
(7)         if better position is found for the $item$-th employed bee, then
(8)             $trial(item) \leftarrow 1$
(9)         else
(10)            $trial(item) \leftarrow trial(item) + 1$
(11)        end if
(12)    end for
(13)    calculate each $P_i$ using (6)
(14)    set $item = 1$
(15)    while $item \le SN$, do
(16)        crossover and mutate using (16) in one randomly selected element for the $item$-th onlooker bee
(17)        adopt greedy selection
(18)        if better position is found, then
(19)            $trial(item) \leftarrow 1$
(20)        else
(21)            $trial(item) \leftarrow trial(item) + 1$
(22)        end if
(23)        $item \leftarrow item + 1$
(24)    end while
(25)    if $trial(i) > D$, $i \in \Omega$, then
(26)        set $trial(i) = D$, $\forall i \in \Omega$
(27)    end if
(28)    if $\mathrm{mean}(trial) \ge 0.9 \cdot D$, then
(29)        re-initialize randomly selected 90% employed bees using (4)
(30)    end if
(31)    memorize current best solution
(32) end for
(33) output global optimum
```

Pseudocode 1

employed bees to search locally using (12). Here, the companion $\mathbf{X}_k$ and the element item j (involved in the crossover and mutation procedure) are still randomly selected. Afterwards, the greedy selection strategy is applied on the onlookers to choose between X^{*j}_i and X^j_i. Consider

$$X^{*j}_i \longleftarrow X^j_i + \gamma(i) \cdot \mathrm{rand}(-1, 1) \cdot \left(X^j_k - X^j_i\right), \qquad (12)$$

where

$$\gamma(i) = \exp\left\{-[trial(i) - 1] \cdot \left(\frac{\ln 10}{D - 1}\right)\right\}. \qquad (13)$$

For each of the employed bees together with the corresponding onlookers, the parameter *trial* represents the number of inefficient searching times before even one better position is derived. If the ith employed bee or the ith onlooker bee finds a better position, $trial(i)$ is directly reset to 1 (but not 0 here); otherwise, it is added by 1. If $trial(i)$ is larger than D, the current ith position $\mathbf{X}_i$ should be replaced by a reinitialized position using (4).

Since $1 \le trial(i) \le D$, it is expected that as many as $trial(i)$ elements in a candidate feasible solution will be affected by the exploration process. But when it comes to onlooker bees, only one element is involved because here we believe that multicrossover process contributes little to local search ability [33]. Note that a convergence factor γ appears in (12), which is designed to manipulate the exploitation accuracy according to the current convergence status of the ith pair of employed bee and onlooker bee. As shown in (13), $\gamma(i)$ decreases exponentially to 0.1 as $trial(i)$ gradually approaches D. Here, 0.1 is a user-specified lower boundary of convergence scale, but the selection of such constant can be flexible according to the users. In this sense, the exploitation around one certain employed bee should be gradually intensified before this pair of bees is eventually discarded and replaced by means of reinitialization (when *trial* equals D).

To briefly conclude, the variable *trial* in IF-ABC is utilized to manipulate the local exploitation accuracy and to guide the crossover and mutation process in global exploration. Here, convergence performances of the bees are measured not by the corresponding objective function values but by the facts whether they are better than the previous one. In this sense, such change intends for the exploitation of positions where unqualified employed bees stay in.

3.4. Principle of BE-ABC Algorithm. BE-ABC algorithm mainly differs from IF-ABC algorithm in two aspects. One is the utilization of the parameter $trial(i)$ to manipulate the exploration/exploitation accuracy and the other is a new strategy for the generation of scout bees [39].

Regarding the exploration procedure, a convergence factor is added in the crossover and mutation equation to manipulate the exploration accuracy. Besides, the number of elements involved in the crossover and mutation process is designed to be adaptable.

In detail, the jth element of the ith employed bee changes to be X^{*j}_{i} as defined in the following:

$$X^{*j}_{i} \longleftarrow X^{j}_{m} + \operatorname{rand}(0,1) \cdot \left(X^{j}_{k} - X^{j}_{i}\right) \cdot \mu(i), \quad (14)$$

$$\mu(i) = \frac{trial(i)}{trial(i) + trial(k)}, \quad (15)$$

where $j \in [1, D]$, for all $m, k \in [1, SN]$, $trial(i) \in [1, D]$, and $i \neq k$.

i is not necessarily equal to k as shown in (14). Here, such modification intends to provide more dynamics for the global exploration. In addition, it is also notable that the lower boundary of $trial(i)$ is 1 (like that in IF-ABC). Then $\mu(i) \in [1/(D+1), D/(D+1)] \subseteq (0,1)$ is regarded as a manipulator for the exploration, which reflects the convergence efficiency of the ith employed bee. Besides, the upper boundary of each $trial(i)$ is set to D; then it is required that as many as $trial(i)$ out of the D elements will be involved in such crossover and mutation procedure for the ith employed bee. Such idea may be intuitively interpreted as follows: a relatively large $trial(i)$ corresponds to a relatively inefficient ith employed bee. Therefore, by changing more elements at one time, the ith employed bee gradually becomes distrustful of its current position. It is similar regarding the definition of $\mu(i)$. If $trial(i)$ is small, $\mu(i)$ will be relatively small (when $trial(k)$ is temporarily fixed). In this sense, the exploration process is more likely to be an exploitation process. In this work, it is believed that there should not be an explicit difference between the exploration and exploitation procedures. In other words, when the exploration/exploitation ability needs to be enhanced, the searching system should be capable of adaptively catering for such demands. In this sense, the proposed BE-ABC algorithm aims to effectively capture a balance between the exploration and the exploitation, so as to make the evolution efficient.

After such crossover and mutation process for the employed bees, the greedy selection strategy will be implemented. If $\mathbf{X}^{*}_{i}$ (defined in (14)) is more qualified, the previous position $\mathbf{X}_i$ is discarded and $trial(i)$ is set to 1; otherwise, the employed bee remains at $\mathbf{X}_i$.

Regarding the onlooker bees, only one element in the solutions will be changed at one time using (16) so as to guarantee that such procedure is sufficiently "local" as follows:

$$Y^{*j}_{0} \longleftarrow X^{j}_{0} + \operatorname{rand}(-1,1) \cdot \left(X^{j}_{k} - X^{j}_{0}\right) \cdot \mu(i). \quad (16)$$

Similarly, it is assumed that the local exploitation needs to be intensified (by using $\mu(i)$) when $trial(i)$ is large. Afterwards, the greedy selection is implemented. If $\mathbf{Y}^{*}_{i}$ is more qualified, the previous position $\mathbf{X}_i$ is discarded and $trial(i)$ is set to 1; otherwise, $trial(i)$ adds one.

In each cycle of iteration, after the exploration and the exploitation procedures, any $trial(i)$ that exceeds D will be reset to D (rather than generating scout bees in the conventional ABC). Before it proceeds to the next cycle, the average value of $trial$ (i.e., $(1/SN)\sum_{i=1}^{SN} trial(i)$) is compared with $90\% \cdot D$. If $90\% \cdot D$ is smaller, which indicates that the overall iteration system is not functioning efficiently, the overall degradation procedure will be carried out. If not, it directly proceeds to the next cycle of iteration. Here, 90% is a user-specified parameter to determine the degree of the inefficient evolution.

In detail, $90\% \cdot SN$ randomly selected employed bees will be reinitialized using (4) in such overall degradation procedure. At the same time, the corresponding $trial(\cdot)$ will be reset to 1 as well. This idea is named overall degradation strategy in contrast with the conventional rule for scout bees in ABC.

Table 1: Information of threat installations in combat field.

Case number	Threat center location	Threat radius	Threat grade
1	[52, 52]	10	2
	[32, 40]	10	10
	[12, 48]	8	1
	[36, 26]	12	2
	[80, 60]	9	3
	[63, 56]	7	5
	[50, 42]	10	2
	[30, 70]	10	4
2	[0, 200]	90	7
	[200, 0]	90	7
	[50, 50]	20	5
	[95, 95]	20	5
	[150, 150]	20	5
	[95, 50]	20	6
	[50, 95]	20	5
	[140, 105]	20	6
	[105, 140]	20	5
3	[30, 20]	10	5
	[50, 15]	21	5
	[65, 55]	10	5
	[0, 24]	17	5
	[50, 80]	27	5
	[75, 90]	10	5
	[100, 70]	20	5
	[50, 36]	11	5

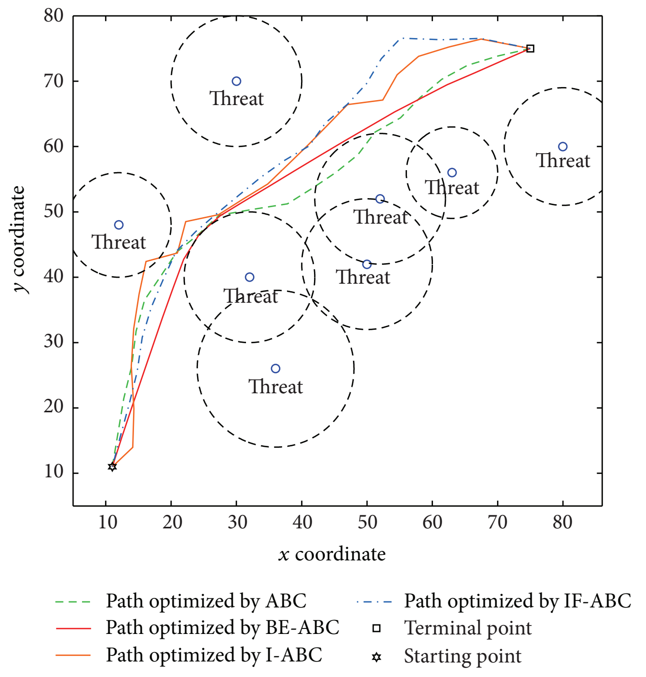

Figure 3: Comparative path planning results optimized by different ABC relevant algorithms in Case 1 ($D = 20$).

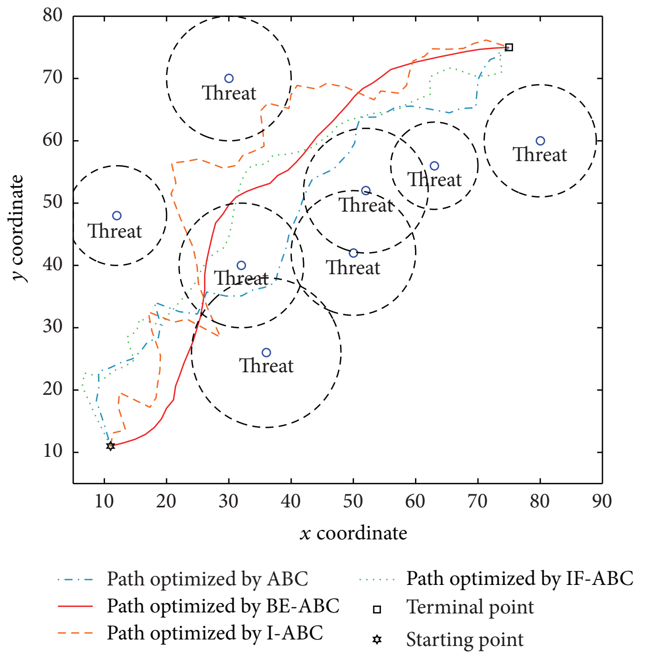

Figure 4: Comparative path planning results optimized by different ABC relevant algorithms in Case 1 ($D = 50$).

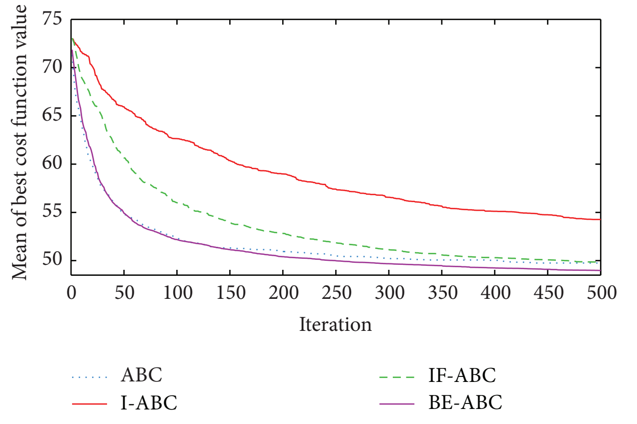

Figure 5: Comparative convergence curves of ABC, I-ABC, IF-ABC, and BE-ABC in Case 1 ($D = 20$ and MCN = 500).

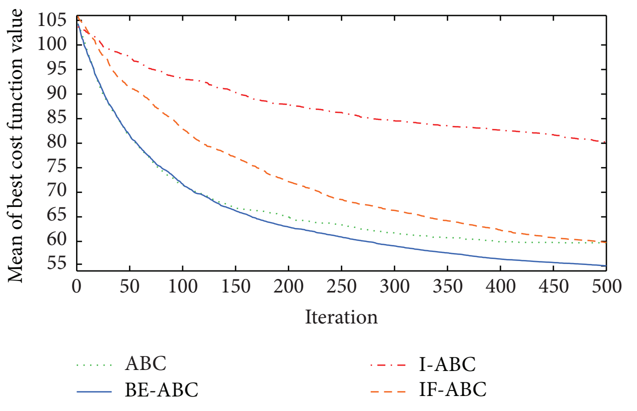

Figure 6: Comparative convergence curves of ABC, I-ABC, IF-ABC, and BE-ABC in Case 1 ($D = 50$ and MCN = 500).

The pseudocode of BE-ABC for numerical optimization is given in Pseudocode 1.

4. Experimental Results and Discussions

In order to investigate the efficiency and robustness of these ABC relevant algorithms, three simulation cases have been investigated. All the contrast experiments involved in this section were implemented with MATLAB R2010a and each kind of experiment was repeated 50 times with different random seeds. It is constantly set that $\lambda = 0.5$, $SN = 30$, and Limit $= 30$.

In the first case, the starting point is set to $(11, 11)$ and the terminal point is set to $(75, 75)$ [12]. In the second and third cases, the starting points are both $(0, 0)$ and the terminal points are $(200, 200)$ and $(100, 100)$, respectively. The threat locations and threat grades for the three cases are listed in Table 1. Some typical simulation results are demonstrated in Figures 3, 4, 5, 6, 7, 8, 9, and 10. In detail, Figures 3 and 4 show the comparative simulation results when $D = 20$ and $D = 50$, respectively, for Case 1. The corresponding convergence curves are shown in Figures 5 and 6. Similarly, paths optimized by different algorithms in Cases 2 and 3 are shown in Figures 7 and 9, respectively, when $D = 30$. Their corresponding convergence curves are plotted in Figures 8 and 10. Complete evaluations of convergence performances (i.e., the mean and the standard deviation of the threat cost function values) are listed in Table 2.

Regarding the paths plotted in Figures 3, 4, 7, and 9, the trajectories optimized by BE-ABC are usually more smooth

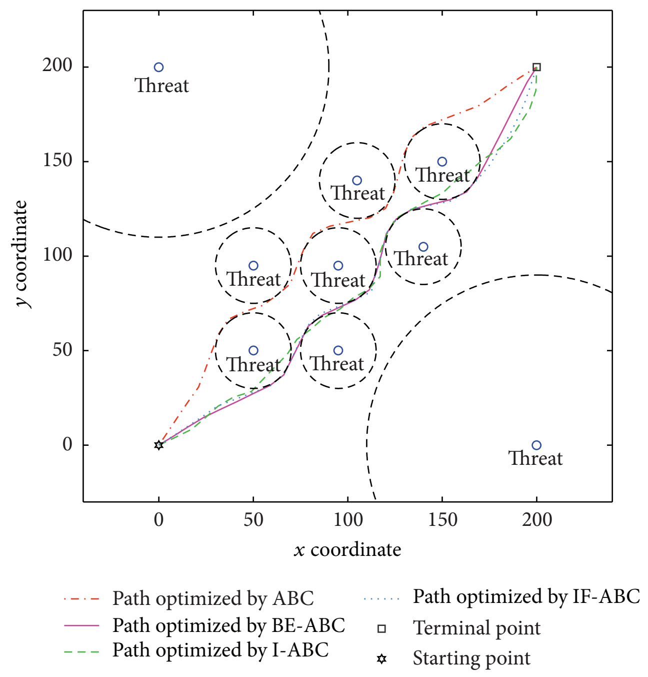

Figure 7: Comparative path planning results optimized by different ABC relevant algorithms in Case 2 ($D = 30$ and MCN = 1000).

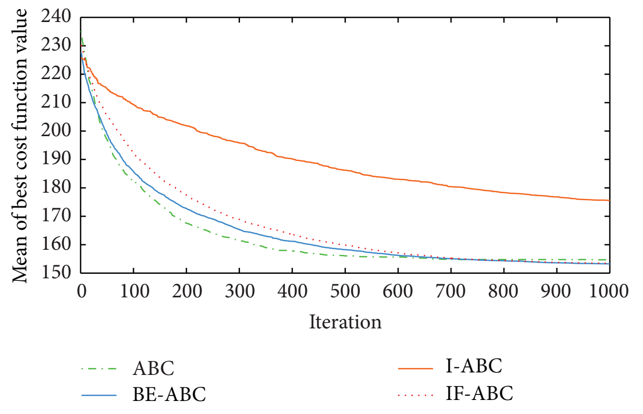

Figure 8: Comparative convergence curves of ABC, I-ABC, IF-ABC, and BE-ABC in Case 2 ($D = 30$ and MCN = 1000).

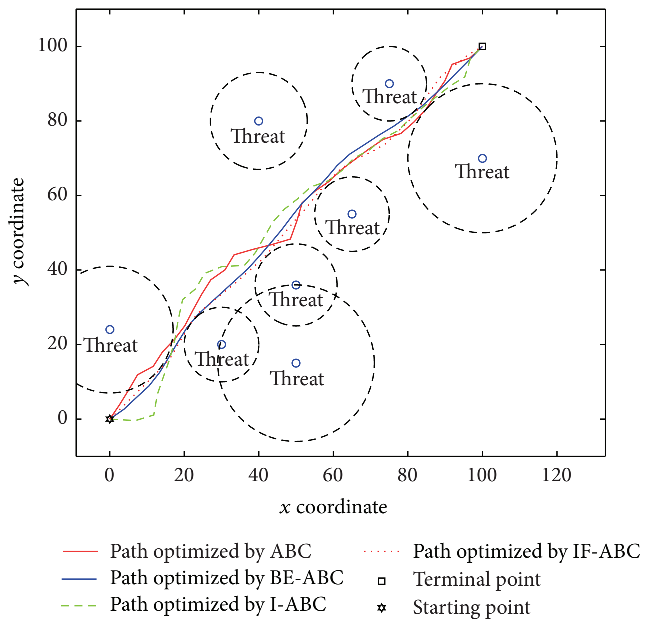

Figure 9: Comparative path planning results optimized by different ABC relevant algorithms in Case 3 ($D = 30$ and MCN = 1000).

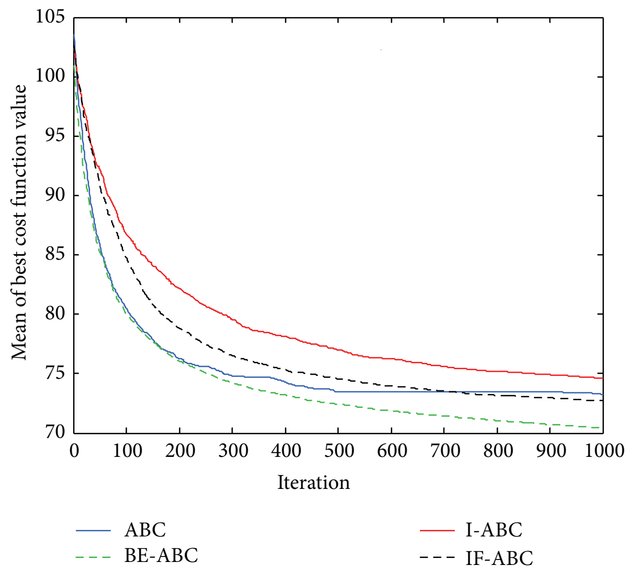

Figure 10: Comparative convergence curves of ABC, I-ABC, IF-ABC, and BE-ABC in Case 3 ($D = 30$ and MCN = 1000).

and advantageous. Specifically, as shown in Figure 7, the path optimized by the conventional ABC happened to be a local optimal solution.

Viewing the results in Table 2 and the comparative curves in Figures 5, 6, 8, and 10, we may notice that the superiority of BE-ABC gradually show up when the dimension D increases. The situation is similar but not so significant regarding IF-ABC. It is because the roulette selection strategy is discarded in IF-ABC, abandoning the feedback information hiding in the objective function values. In a way, IF-ABC sacrifices part of its ability to converge fast for the competence to converge globally [33]. In contrast, improvements made in BE-ABC are relatively moderate and mild, which may account for its good convergence performance.

In the meantime, in some early cycles of iteration, the convergence speed of BE-ABC tends to be roughly the same with that of the conventional ABC. Such phenomenon may be due to the fact that, during the early cycles of iteration, it is relatively easy to evolve for each of the employed bees; that is, $trial(i)$ are not large in general. Therefore, BE-ABC is similar with the conventional ABC in the convergence performance. However, as the iteration process continues, the efficiency of ABC will be impacted by the obstacles of local optimums. At the same time, the modifications in BE-ABC algorithm make sense.

Table 2: The mean and standard deviations of cost function values.

Case number	MCN	D	ABC		I-ABC		IF-ABC		BE-ABC	
			Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
1	500	20	49.8813	0.5976	54.3889	3.6656	49.7975	0.6393	**48.9558**	**0.5957**
	500	30	52.9887	1.4259	60.4591	3.2883	51.9396	1.3434	**50.4567**	**0.6726**
	500	50	59.9722	2.9133	80.1747	7.7575	59.9569	**2.2149**	**54.9826**	2.2310
2	1000	20	161.7459	2.5523	171.6983	2.1826	160.1839	**0.9241**	**160.0276**	1.5195
	1000	30	154.9276	2.5123	185.7927	3.7547	153.5837	**0.4925**	**153.4103**	0.5549
	1000	50	157.2044	3.6031	164.9908	2.8118	157.8147	1.9058	**153.5245**	**1.0479**
3	1000	20	73.5090	0.2486	73.7209	0.4729	73.0254	0.1079	**72.8889**	**0.1403**
	1000	30	73.9346	1.0134	74.8452	0.8904	73.6928	**0.4083**	**70.0789**	0.4634
	1000	50	78.8720	3.1422	81.8885	2.7718	77.4828	2.0989	**75.8906**	**1.3950**

Those bold values denote the best solutions (mean or S.D.) among four algorithms in every single case.

5. Conclusion

In this paper, BE-ABC algorithm is applied for the UCAV path planning optimization problem. Simulation results clearly indicate that BE-ABC shows more stability and efficiency in this two-dimensional flight path planning optimization scheme than ABC, I-ABC, and IF-ABC.

BE-ABC algorithm intends to fully utilize the convergence status within the iteration system so as to manipulate the searching accuracy and also to strike a balance between the local exploitation with global exploration. Previous studies concerning the improvements of ABC always focused on the remedies from the outside world, neglecting the true convergence status hiding in the internal iteration process. In this sense, this work can be regarded as publicity for such idea. Our future work will cover some further comparisons with more state-of-the-art intelligent algorithms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by the School of Advanced Engineering (SAE) in Beihang University and sponsored by the 5th and 6th National College Students' Innovation and Entrepreneurial Training Programs in China.

References

[1] Y. Zhang, Y. Jun, G. Wei, and L. Wu, "Find multi-objective paths in stochastic networks via chaotic immune PSO," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1911–1919, 2010.

[2] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 132–141, 2013.

[3] Y. Zhang, P. Agarwal, V. Bhatnagar, S. Balochian, and J. Yan, "Swarm intelligence and its applications," *The Scientific World Journal*, vol. 2013, Article ID 528069, 3 pages, 2013.

[4] B. Li, L. G. Gong, and C. H. Zhao, "Unmanned combat aerial vehicles path planning using a novel probability density model based on Artificial Bee Colony algorithm," in *Proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP '13)*, pp. 620–625, Beijing, China, June 2013.

[5] D. G. MacHaret, A. A. Neto, and M. F. M. Campos, "Feasible UAV path planning using genetic algorithms and Bézier curves," *Proceedings of the Advances in Artificial Intelligence Conference (SBIA '10)*, Springer, Berlin, Germany, vol. 6404, pp. 223–232, 2010.

[6] H. H. John, *Adaptation in Natural and Artificial Systems*, University of Michigan, Ann Arbor, Mich, USA, 1975.

[7] R. Storn and K. Price, "Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[8] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the 1st European conference on artificial life*, pp. 134–142, Paris, France, 1991.

[9] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, October 1995.

[10] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[11] L. Guo, G. Wang, H. Wang, and D. Wang, "An effective hybrid firefly algorithm with harmony search for global numerical optimization," *The Scientific World Journal*, vol. 2013, Article ID 125625, 9 pages, 2013.

[12] C. Xu, H. Duan, and F. Liu, "Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning," *Aerospace Science and Technology*, vol. 14, no. 8, pp. 535–541, 2010.

[13] Z. Cheng, Y. Sun, and Y. Liu, "Path planning based on immune genetic algorithm for UAV," in *Proceedings of the International Conference on Electric Information and Control Engineering (ICEICE '11)*, pp. 590–593, Wuhan, China, April 2011.

[14] Y. Bao, X. Fu, and X. Gao, "Path planning for reconnaissance UAV based on particle swarm optimization," in *Proceedings of the 2nd International Conference on Computational Intelligence and Natural Computing (CINC '10)*, pp. 28–32, Wuhan, China, September 2010.

[15] Y. Fu, M. Ding, and C. Zhou, "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV," *IEEE Transactions on Systems, Man, and Cybernetics A: Systems and Humans*, vol. 42, no. 2, pp. 511–526, 2012.

[16] D. M. Pierre, N. Zakaria, and A. J. Pal, "Master-slave parallel vector-evaluated genetic algorithm for unmanned aerial vehicle's path planning," in *Proceedings of the 11th International Conference on Hybrid Intelligent Systems (HIS '11)*, pp. 517–521, Melacca, Malaysia, December 2011.

[17] H. Duan, S. Liu, and J. Wu, "Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning," *Aerospace Science and Technology*, vol. 13, no. 8, pp. 442–449, 2009.

[18] Z. Yin, X. Liu, and Z. Wu, "A multiuser detector based on artificial bee colony algorithm for DS-UWB systems," *The Scientific World Journal*, vol. 2013, Article ID 547656, 8 pages, 2013.

[19] B. Li and Y. Yao, "An edge-based optimization method for shape recognition using atomic potential function," *Engineering Applications of Artificial Intelligence*. Submitted.

[20] H. Sun, H. Lus, and R. Betti, "Identification of structural models using a modified Artificial Bee Colony algorithm," *Computers and Structures*, vol. 116, pp. 59–74, 2013.

[21] B. Li, Y. Li, and L. Gong, "Protein secondary structure optimization using an improved artificial bee colony algorithm based on AB off-lattice model," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 70–79, 2014.

[22] R. Irani and R. Nasimi, "Application of artificial bee colony-based neural network in bottom hole pressure prediction in underbalanced drilling," *Journal of Petroleum Science and Engineering*, vol. 78, no. 1, pp. 6–12, 2011.

[23] B. Li and Y. Li, "BE-ABC: hybrid artificial bee colony algorithm with balancing evolution strategy," in *Proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP '12)*, pp. 217–222, Dalian, China, June 2012.

[24] F. Kang, J. Li, and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," *Information Sciences*, vol. 181, no. 16, pp. 3508–3531, 2011.

[25] H.-B. Duan, C.-F. Xu, and Z.-H. Xing, "A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems," *International Journal of Neural Systems*, vol. 20, no. 1, pp. 39–50, 2010.

[26] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[27] D. Haijun and F. Qingxian, "Artificial bee colony algorithm based on boltzmann selection strategy," *Computer Engineering and Applications*, vol. 45, no. 32, pp. 53–55, 2009.

[28] Y. Li, Y. Wang, and B. Li, "A hybrid artificial bee colony assisted differential evolution algorithm for optimal reactive power flow," *International Journal of Electrical Power and Energy Systems*, vol. 52, pp. 25–33, 2013.

[29] M. S. Kiran and M. Gündüz, "A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems," *Applied Soft Computing*, vol. 13, pp. 2188–2203, 2013.

[30] H. Duan, Z. Xing, and C. Xu, "An improved quantum evolutionary algorithm based on artificial bee colony optimization," in *Advances in Computational Intelligence*, vol. 116, pp. 269–278, Springer, Berlin, Germany, 2009.

[31] A. Banharnsakun, B. Sirinaovakul, and T. Achalakul, "Job shop scheduling with the Best-so-far ABC," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 583–593, 2012.

[32] G. Li, P. Niu, and X. Xiao, "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization," *Applied Soft Computing Journal*, vol. 12, no. 1, pp. 320–332, 2012.

[33] B. Li, L. G. Gong, and Y. Yao, "On the performance of internal feedback artificial bee colony algorithm (IF-ABC) for protein secondary structure prediction," in *presented in Proceedings of the 6th International Conference on Advanced Computational Intelligence*, pp. 33–38, Hangzhou, China, 2013.

[34] B. Li, "Research on WNN modeling based on an improved artificial Bee Colony algorithm for gold price prediction," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 270658, 10 pages, 2014.

[35] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.

[36] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, "A Bat algorithm with mutation for UCAV path planning," *The Scientific World Journal*, vol. 2012, Article ID 418946, 15 pages, 2012.

[37] Y. Zhang, L. Wu, and S. Wang, "UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2013, Article ID 705238, 9 pages, 2013.

[38] D. Rodic and A. P. Engelbrecht, "Social networks in simulated multi-robot environment," *International Journal of Intelligent Computing and Cybernetics*, vol. 1, pp. 110–127, 2008.

[39] B. Li and R. Chiong, "A new artificial bee colony algorithm based on a balance-evolution strategy for numerical optimization," Submitted.