Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 705238, 9 pages
http://dx.doi.org/10.1155/2013/705238

Research Article

UCAV Path Planning by Fitness-Scaling Adaptive Chaotic Particle Swarm Optimization

Yudong Zhang,[1] Lenan Wu,[2] and Shuihua Wang[1,3]

[1] *School of Computer Science and Technology, Nanjing Normal University, Nanjing, Jiangsu 210023, China*
[2] *School of Information Science and Engineering, Southeast University, Nanjing, Jiangsu 210096, China*
[3] *School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu 210046, China*

Correspondence should be addressed to Shuihua Wang; shuihuaw2007@gmail.com

Received 10 April 2013; Revised 15 June 2013; Accepted 28 June 2013

Academic Editor: Saeed Balochian

Path planning plays an extremely important role in the design of UCAVs to accomplish the air combat task fleetly and reliably. The planned path should ensure that UCAVs reach the destination along the optimal path with minimum probability of being found and minimal consumed fuel. Traditional methods tend to find local best solutions due to the large search space. In this paper, a Fitness-scaling Adaptive Chaotic Particle Swarm Optimization (FAC-PSO) approach was proposed as a fast and robust approach for the task of path planning of UCAVs. The FAC-PSO employed the fitness-scaling method, the adaptive parameter mechanism, and the chaotic theory. Experiments show that the FAC-PSO is more robust and costs less time than elite genetic algorithm with migration, simulated annealing, and chaotic artificial bee colony. Moreover, the FAC-PSO performs well on the application of dynamic path planning when the threats cruise randomly and on the application of 3D path planning.

1. Introduction

Unmanned combat air vehicle (UCAV) is an experimental class of the unmanned aerial vehicle (UAV). UCAVs differ from ordinary UAVs because they are designed to deliver weapons to attach enemy targets. The elimination of the need for an onboard human crew in a UCAV that may be shot down over enemy territory has obvious advantages for personnel safety. In addition, much equipment necessary for a human pilot (such as the cockpit, flight controls, oxygen, and seat/ejection seat) can be omitted from an unmanned vehicle, resulting in a decrease in weight possibly allowing greater payloads, range, and maneuverability.

The path planning of UCAV is to generate a space path between an initial safe location and the desired dangerous destination that has an optimal or near-optimal performance under specific constraint conditions. It is always a complex research subject, so it is an imperative technology required in the design of UCAV. Series of algorithms have been proposed to solve this complicated multiconstrained optimization problem. Allaire used a genetic algorithm (GA) to realize the FPGA implementation for UAV real-time path planning [1]. Duan et al. proposed an improved particle swarm optimization to optimize the formation reconfiguration control of multiple UCAVs [2], proposed a hybrid metaheuristic ant colony optimization (ACO) and differential evolution (DE) to solve the UCAV three-dimension path-planning problem [3], and proposed a max-min adaptive ant colony algorithm for multi-UCAVs coordinated trajectory replanning [4]. Mou et al. proposed a modified ant colony algorithm as a fast and efficient approach for path planning of UCAV [5]. Zhang et al. proposed an improved artificial bee colony algorithm for UCAV path-planning problem [6]. However, these methods can easily be trapped into local minima and cannot solve the contradiction between the goal optimization and excessive information.

PSO is well known for its lower computational costs, and its most attractive feature is that it requires less computational bookkeeping and only a few lines of implementation codes. In order to improve the performance of a traditional PSO, three improvements are proposed: (I) a new power-rank fitness-scaling method, by which the scaled values are suitable for following selection; (II) adaptively varied parameters to search an expansive area at the prophase stage and

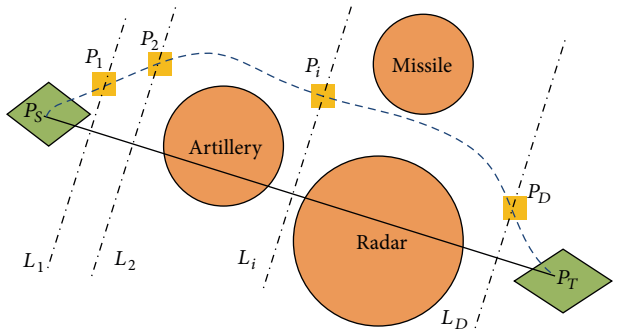

Figure 1: Typical 2D UCAV battlefield model.

a restricted area at the anaphase stage; and (III) introduction of chaos to improve the robustness of the basic PSO algorithm considering its outstanding performance of jumping out of stagnation.

The remainder of this paper is organized as follows. Section 2 introduces the encoding strategy for path planning. Section 3 discusses the performance evaluation function containing both the threat cost and the fuel cost. Section 4 introduces the basic principles of canonical PSO. Section 5 gives detailed description of our proposed method—Fitness-scaling Adaptive Chaotic PSO (FAC-PSO). Experiments in Section 6 compared our proposed method with elite genetic algorithm with migration, simulated annealing, and chaotic artificial bee colony. The statistical results on 100 different runs demonstrate that the FAC-PSO is superior to other algorithms with respect to success rate and computation time. Besides, we also applied our approach in the field of dynamic UCAV path planning. Final Section 7 is devoted to the conclusions.

2. Path Encoding

In this model, the starting point and the target point are defined as P_S and P_T, respectively. There are threatening areas in the task region, such as artillery, radar, and missile, which all are presented in the form of a circle. Inside of the threatening areas, the UCAV should be vulnerable to the threat with a certain probability proportional to the distance away from the threat center, while outside of the threatening areas, the UCAV should be safe without being attacked. The task of path planning is to design an optimal path between start point and target point considering all these threatening areas as shown in Figure 1.

We connect the starting point and target point and then divide the straight line P_SP_T into $(D + 1)$ equal portions. At each segment point, draw the vertical line of P_SP_T, which can be labeled with $L_1, L_2, \ldots, L_i, \ldots, L_D$. Select discrete points P_i at each L_i. In this way, the path from the starting node to the target node can be described as follows:

$$\text{path} = \{P_S, P_1, P_2, \ldots, P_i, \ldots, P_D, P_T\}. \quad (1)$$

The location of P_S and P_T is known for UCAV, and the location of line L_i $(i = 1, 2, \ldots, D)$ can be easily calculated. Therefore, each point P_i $(i = 1, 2, \ldots, D)$ can be expressed using only 1 parameter, namely, its distance to the straight line P_SP_T. In a word, there are total D parameters in (1), so our task is a D-dimensional optimization problem. In the following section, we let $P_0 = P_S$ and $P_{D+1} = P_T$ in order to simplify the expression.

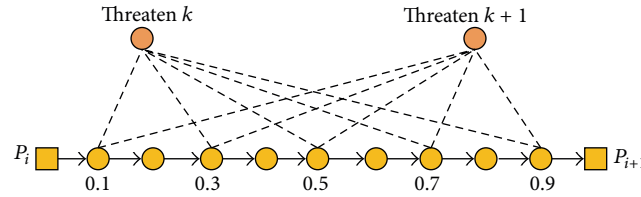

Figure 2: Computation of threat cost.

3. Performance Function

The performance indicators of the planned path mainly consist of the threat cost J_T and the fuel cost J_F. Their calculation formulas are expressed as follows:

$$J_T = \sum_{i=0}^{D} J_T(i),$$
$$J_F = \sum_{i=0}^{D} J_F(i). \quad (2)$$

Here, $J_T(i)$ and $J_F(i)$ denote the threat cost and fuel cost at the ith subpath from P_i to P_{i+1}, respectively. The threat cost of subpath is calculated by an approximation based on five discrete points along the subpath as shown in Figure 2. If the ith subpath (P_i, P_{i+1}) is within the effect range, the threat cost is given as [7]

$$J_T(i) = \frac{L_i}{5} \sum_{k=1}^{N_T} T_k \left(\frac{1}{d_{0.1,i,k}^4} + \frac{1}{d_{0.3,i,k}^4} + \frac{1}{d_{0.5,i,k}^4} + \frac{1}{d_{0.7,i,k}^4} + \frac{1}{d_{0.9,i,k}^4} \right). \quad (3)$$

Here, N_T denotes the number of threatening areas, L_i denotes the length of ith subpath, T_k denotes the degree of threatening, and $d_{0.1,i,k}$ denotes the distance from the 1/10 point on the ith subpath to the kth threat area.

Suppose that the velocity of UCAV is constant, the fuel cost of the ith subpath $J_F(i)$ can be considered proportional to L_i. Therefore, the total cost of the path is proportional to the total length of the path L.

The total cost for traveling along the trajectory comes from a weighted sum of the threat and fuel costs, as defined in the following formula

$$J = \omega J_T + (1 - \omega) J_F, \quad (4)$$

where ω is a variable between 0 and 1, giving the designer certain flexibility to dispose relations between the threat exposition degree and the fuel consumption. If ω approaches 1, a safer path is needed and less attention is paid to the fuel. Alternatively, if ω approaches 0, a shorter path is needed even on the cost of sacrifice the safety. In this study, we determine it as 0.5 by the suggestion from [8], indicating that the threat is as important as the fuel.

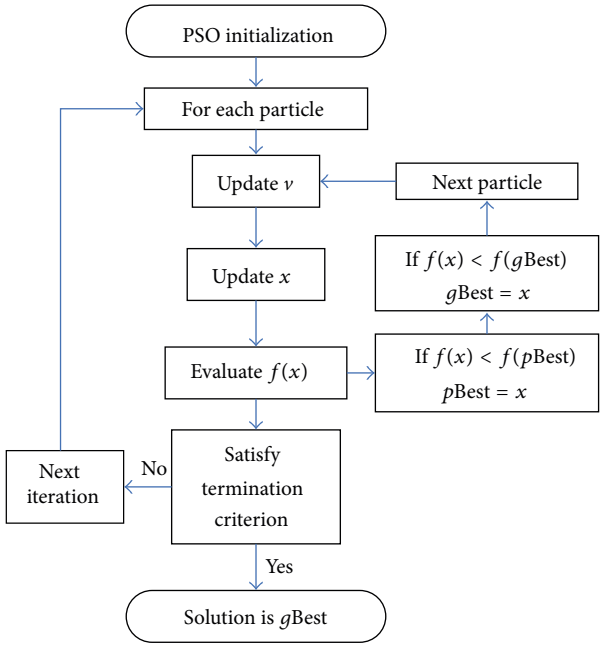

Figure 3: Flow chart of the PSO algorithm.

4. Particle Swarm Optimization

PSO is a population-based stochastic optimization technique, which simulates the social behavior of a swarm of birds, flocking bees, and fish schooling [9]. By randomly initializing the algorithm with candidate solutions, the PSO successfully leads to a global optimum. This is achieved by an iterative procedure based on the processes of movement and intelligence in an evolutionary system. Figure 3 shows the flow chart of a PSO algorithm.

In PSO, each potential solution is represented as a particle. Two properties (position x and velocity v) are associated with each particle. Suppose x and v of the ith particle are given as

$$\begin{aligned} x &= \left(x_{i1}, x_{i2}, \ldots, x_{iN}\right), \\ v &= \left(v_{i1}, v_{i2}, \ldots, v_{iN}\right), \end{aligned} \tag{5}$$

where N stands for the dimensions of the problem. In each iteration, a fitness function is evaluated for all the particles in the swarm. The velocity of each particle is updated by keeping track of the two best positions. One is the best position a particle has traversed so far and called "pBest". The other is the best position that any neighbor of a particle has traversed so far. It is a neighborhood best called "nBest". When a particle takes the whole population as its neighborhood, the neighborhood best becomes the global best and is accordingly called "gBest". Hence, a particle's velocity and position are updated as follows:

$$\begin{aligned} v &= \omega \cdot v + c_1 r_1 \left(p\text{Best} - x\right) + c_2 r_2 \left(n\text{Best} - x\right), \\ x &= x + v\Delta t, \end{aligned} \tag{6}$$

where ω is called the "*inertia weight*" that controls the impact of the previous velocity of the particle on its current one. The parameters c_1 and c_2 are positive constants called "*acceleration coefficients*". The parameters r_1 and r_2 are random numbers uniformly distributed in the interval $[0, 1]$. These random numbers are updated every time they occur. The parameter Δt stands for the given time step.

The population of particles is then moved according to (6) and tends to cluster together from different directions. However, a maximum velocity $v_{\max}$ should not be exceeded by any particle to keep the search within a meaningful solution space. The PSO algorithm runs through these processes iteratively until the termination criterion is satisfied [10].

5. Principle of FAC-PSO

The PSO has proven to perform better than GA, DE, and ACO [11]. However, we can make further improvements from the following three aspects.

5.1. Fitness Scaling. Fitness scaling converts the raw fitness scores that are returned by the fitness function to values in a range that is suitable for the selection function [12]. The selection function uses the scaled fitness values to select the particles of the next generation. Then, the selection function assigns a higher probability of selection to particles with higher scaled values.

There exist bundles of fitness-scaling methods. One of the most common scaling techniques is traditional linear scaling, which remaps the fitness values of each particle using the following equation:

$$f_{\text{linear}} = a + b \times f_{\text{raw}}, \tag{7}$$

where a and b are constants defined by users. Another option is the rank scaling, which is obtained by sorting all the bees by their raw fitness values

$$f_{\text{rank}} = r, \tag{8}$$

where r denotes the rank of the individual particle. The third option is the power scaling method which is instead computed with

$$f_{\text{power}} = f_{\text{raw}}^{k}, \tag{9}$$

where k is a problem-dependent exponent that might require change during a run to stretch or shrink the range as needed. Top scaling is the 4th option and probably the simplest scaling method. Using this approach, several of the top individuals have their fitness set to the same value, with all remaining individuals having their fitness values set to zero. This simple concept yields

$$f_{\text{top}} = \begin{cases} s & f_{\text{raw}} \geq c \\ 0 & f_{\text{raw}} < c, \end{cases} \tag{10}$$

where s is the user-defined constant, c is the threshold.

Among those fitness-scaling methods, the power scaling finds a solution nearly the most quickly due to improvement of diversity but it suffers from instability [13]; meanwhile,

the rank scaling shows stability on different types of tests [14]. Therefore, a new power-rank scaling method was proposed combing both power and rank strategies as follows:

$$fit_i = \frac{r_i^k}{\sum_{i=1}^{N} r_i^k}, \tag{11}$$

where r_i is the rank of ith individual bee, N is the number of population. Our strategy contains a three-step process. First, all bees are sorted to obtain the corresponding ranks. Second, powers are computed for exponential values k. Third, the scaled values are normalized by dividing the sum of the scaled values over the entire population.

5.2. Adaptive Parameters. Another improvement lies in changing the parameters (ω, c_1, c_2) adaptively. In the search process of PSO, the search space will gradually reduce as the generation increases. Therefore, we hope to search an expansive area with low precision at the prophase stage while searching a restricted area with high precision at the anaphase stage as listed in Table 1. The detailed formulas of those adaptive parameters are as follows:

$$\begin{aligned}
\omega &= \omega_i - \frac{\omega_i - \omega_f}{\text{MaxGeneration}} * \text{Generation} \left(\omega_i > \omega_f\right), \\
c_1 &= c_{1i} - \frac{c_{1i} - c_{1f}}{\text{MaxGeneration}} * \text{Generation} \left(c_{1i} > c_{1f}\right), \\
c_2 &= c_{2i} - \frac{c_{2i} - c_{2f}}{\text{MaxGeneration}} * \text{Generation} \left(c_{2i} < c_{2f}\right).
\end{aligned} \tag{12}$$

Here, the indexes i and f denote "initial" and "final", respectively.

5.3. Chaotic Random Number. The parameters (r_1, r_2) were generated by a pseudorandom number generator (RNG) in classical PSO. The RNG cannot ensure the optimization's ergodicity in solution space because it is absolutely random; therefore, a chaotic operator was employed to generate parameters (r_1, r_2) by the following formula:

$$r_i(t+1) = 4.0 * r_i(t) * \left[1 - r_i(t)\right], \quad i = 1, 2, \tag{13}$$

where $r_0 \in (0, 1)$ and $r_0 \notin \{0.25, 0.5, 0.75\}$. A very small difference in the initial value of x would give rise to a large difference in its long-time behavior as shown in Figure 4. The track of chaotic variable x_n can travel ergodically over the whole space of interest.

Figure 5 shows that the series is in the cycle (0.75) when initial points are 0.25 or 0.75 since $4 * 0.25 * (1 - 0.25) = 0.75$ and $4 * 0.75 * (1 - 0.75) = 0.75$. The series is in cycle (0) when initial point is 0.5 since $4 * 0.5 * (1 - 0.5) = 1$ and $4 * 1 * (1 - 1) = 0$. Therefore, those three points (0.25, 0.5, and 0.75) will make the series lose chaotic property.

6. Experiments

The experiments were carried out on the platform of P4 IBM with 3 GHz main frequency and 2 GB memory, running under Windows XP operating system. The algorithm was developed via the global optimization toolbox of MatLab 2011a.

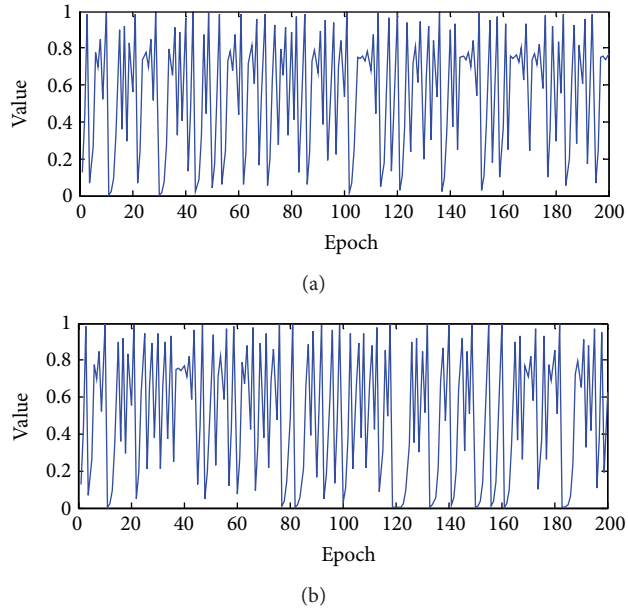

Figure 4: Chaotic Property of logistic equation: (a) $x_0 = 0.12345678$; (b) $x_0 = 0.12345679$.

6.1. Threat Setting. Set the coordinates of the starting point as (5, 5) and the target point as (100, 100). In the flight course, there exist eight threat areas listed in Table 2. Suppose that the codes of initial path are all zeros, which corresponds to a straight line from starting point directly to the target point as shown in Figure 6. Traditional gradient-based methods will guide the 12th–20th nodes of the path to search the upper-left area, and they will finally be misled into the local minima. However, the evolutionary algorithms including GA, PSO, and our proposed FAC-PSO are able to jump from the local minima and search the bottom-right area, where the global minimal point locates in.

6.2. Algorithm Comparison. We compared the proposed FAC-PSO method with elite genetic algorithms with migration (EGAM) [15], simulated annealing (SA) [16], chaotic artificial bee colony (CABC) [8], and standard PSO. The parameters are obtained through trial-and-error method and shown in Table 3. Here, *NP* means the number of populations/bees/particles corresponding to different algorithms, and *MaxEpoch* means the maximum iterative epochs. P_C, P_m, and P_e of EGAM stand for the crossover probability, mutation probability, and elite probability, respectively. TDF, T_I, and T_F of SA denote the temperature decrease function, initial temperature, and final temperature, respectively. N_F in CABC represents the number of foods.

Each algorithm ran 100 times, and the success rate was calculated and shown in Table 4. It indicates that the proposed FAC-PSO show slight superiority to other algorithms when

Table 1: Parameters variation.

	Ω	c_1	c_2	Performance
Prophase	Larger	Larger	Smaller	PSO searches for global optima in an expansive area with low precision
Anaphase	Smaller	Smaller	Larger	PSO searches for local optima in a limited area with high precision

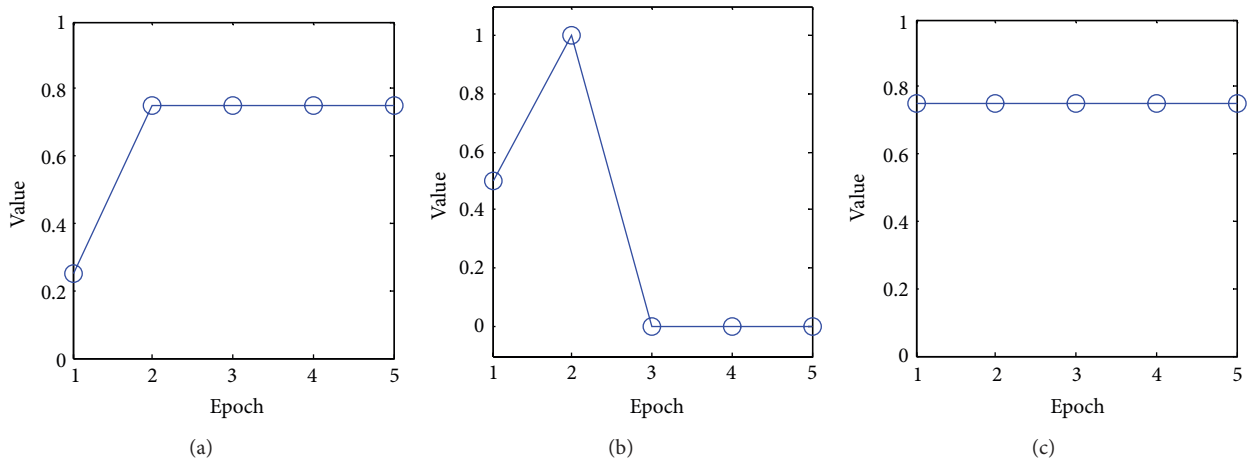

Figure 5: Loss of chaotic property at specific initial values: (a) $x_0 = 0.25$; (b) $x_0 = 0.5$; and (c) $x_0 = 0.75$.

Table 2: Information of 2D threatening objects.

Index	Position	Radius
1	(10, 30)	14
2	(10, 50)	10
3	(20, 80)	20
4	(40, 15)	12
5	(40, 50)	15
6	(50, 70)	12
7	(75, 70)	14
8	(80, 40)	12

$D = 10$. As the D increases, the FAC-PSO shows more robustness compared to other algorithms. It should be noted that a larger D makes the search space larger, which leads to the success rate of all the algorithms decreasing.

We take $D = 20$ as an example, choose a typical run and show the convergence plot in Figure 7. It indicates that the proposed FAC-PSO was trapped into local minima at about 42 epochs but it jump out at about 45th epoch. Conversely, the EGAM, SA, CABC, and PSO were stagnated in the local minima over all 100 epochs.

The final searched paths of the four algorithms are shown in Figure 8. It indicates that our proposed FAC-PSO found the global best path of the algorithm, while the other four algorithms failed at this run.

6.3. Time Comparison. Computation time is another important factor used to evaluate the algorithm. Since a failed run usually takes little time due to early stagnation, we only consider the success runs. The average computation times of each algorithm of different sizes of D are shown in Table 5. It indicates that the proposed algorithm takes the least time in spite of the size of D; moreover, the SA takes the second least time for calculation. The most time-consuming algorithm is the CABC.

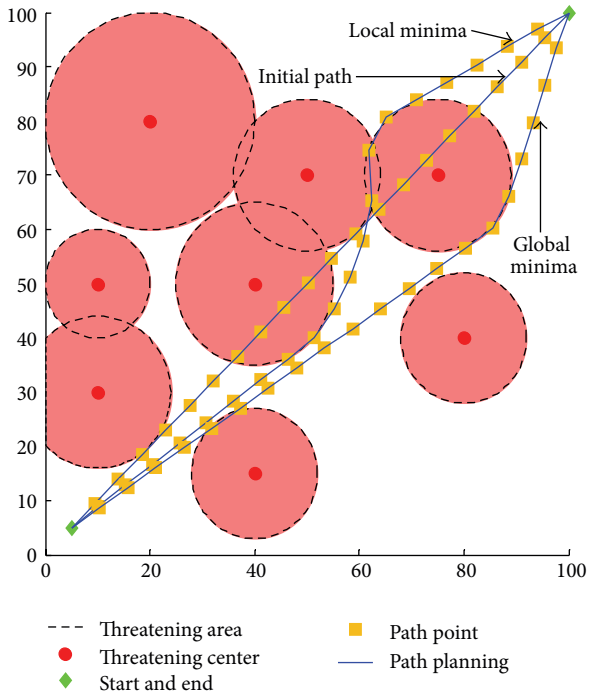

Figure 6: Local minima and global minima.

Table 3: Parameters of algorithms.

Algorithm	Parameter setting
EGAM	$NP = 20$; $MaxEpoch = 100$; $P_C = 0.8$; $P_m = 0.1$; $P_e = 0.1$; and *migration interval* = 20.
SA	$NP = 20$; $MaxEpoch = 100$; TDF = "exponential"; $T_I = 100$, and $T_F = 0$.
CABC	$NP = 20$; $MaxEpoch = 100$; and $N_F = 10$.
PSO	$NP = 20$; $MaxEpoch = 100$; $v_{\max} = 1$; $\omega = 0.6$; $c_1 = 1$; and $c_2 = 1$.
FAC-PSO	$NP = 20$; $MaxEpoch = 100$; $v_{\max} = 1$; $\omega_i = 0.9$; $\omega_f = 0.4$; $c_{1i} = 2.5$; $c_{1f} = 0.5$, $c_{2i} = 0.5$; and $c_{2f} = 2.5$.

Table 4: Success rates of different algorithms for 2D UCAV.

D	EGAM	SA	CABC	PSO	FAC-PSO
10	78%	22%	80%	75%	**87%**
15	66%	12%	67%	71%	**85%**
20	53%	3%	59%	56%	**80%**

Table 5: Average computation time (s).

D	EGAM	SA	CABC	PSO	FAC-PSO
10	12.3	11.6	14.6	13.0	**10.2**
15	12.9	13.8	15.3	14.7	**11.3**
20	14.5	13.6	16.8	14.9	**13.7**

Table 6: Information of 3D threatening objects.

Index	Position	Radius
1	(10, 30, 30)	14
2	(10, 50, 20)	10
3	(20, 80, 40)	20
4	(40, 15, 70)	12
5	(40, 50, 50)	15
6	(50, 70, 40)	12
7	(75, 70, 35)	14
8	(80, 40, 50)	12
9	(40, 55, 30)	10
10	(30, 40, 40)	15

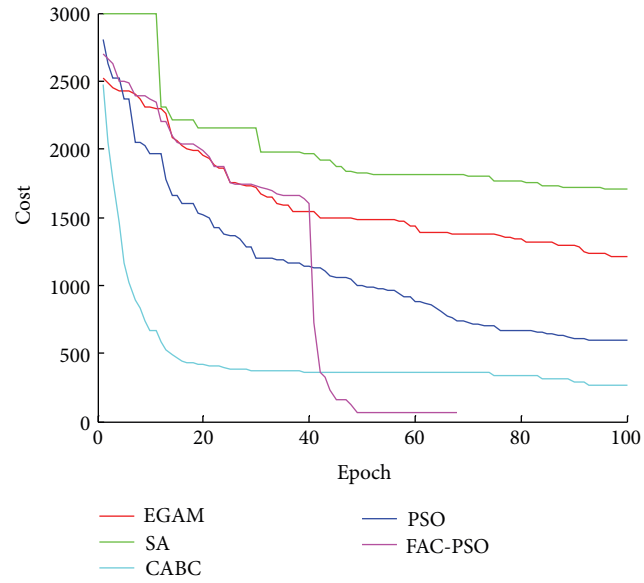

Figure 7: A typical convergence plot of different algorithms.

Table 7: Success rates of different algorithms for 3D UCAV.

D	EGAM	SA	CABC	PSO	FAC-PSO
15	65%	9%	64%	65%	**74%**
20	41%	2%	43%	42%	**68%**
25	16%	0%	21%	20%	**53%**

6.4. Dynamic Path Planning. The aforementioned paths are static ones which are determined by a beforehand-known map and threat information, but the UCAV usually meets unforeseen threats in actual flight course, so they must possess dynamic path-planning ability. When UCAV detects instantaneous or moving threat, it must replan the path so as to avoid the new arisen threat by revising the former path.

Suppose that all the threatening areas can move randomly, then, the flight path of UCAV should change after each move according to the current threat positions. The UCAV paths by the proposed FAC-PSO at steps 0, 5, 10, and 15 are shown in Figure 9, which imply the feasibility of FAC-PSO under moving threatening conditions.

6.5. 3D Path Planning. We applied our method to 3D UCAV path planning. First, we generate a cube ($100 \times 100 \times 100$). Second, we generated 10 threats, and their coordinates and radii are listed in Table 6.

We set the coordinates of the starting point as (5, 5, and 5) and the target point as (100, 100, and 100). We compared the proposed FAC-PSO method with EGAM, SA, CABC, and PSO. All parameters are the same as Table 3 except that the maximal epoch is changed to 1000. Each algorithm ran 100 times, and the success rate was calculated and shown in Table 7. We found that the FAC-PSO performs best among all algorithms for 3D UCAV path planning.

7. Conclusions

In this study, a novel FAC-PSO approach for UCAV path planning was proposed. We first investigate the path encoding strategy and then construct the cost function which combines the threat cost and fuel cost simultaneously. The FAC-PSO algorithm was proposed utilizing the fitness-scaling, the adaptive mechanism, and the ergodicity and irregularity of the chaos. Compared to standard PSO, the FAC-PSO is more

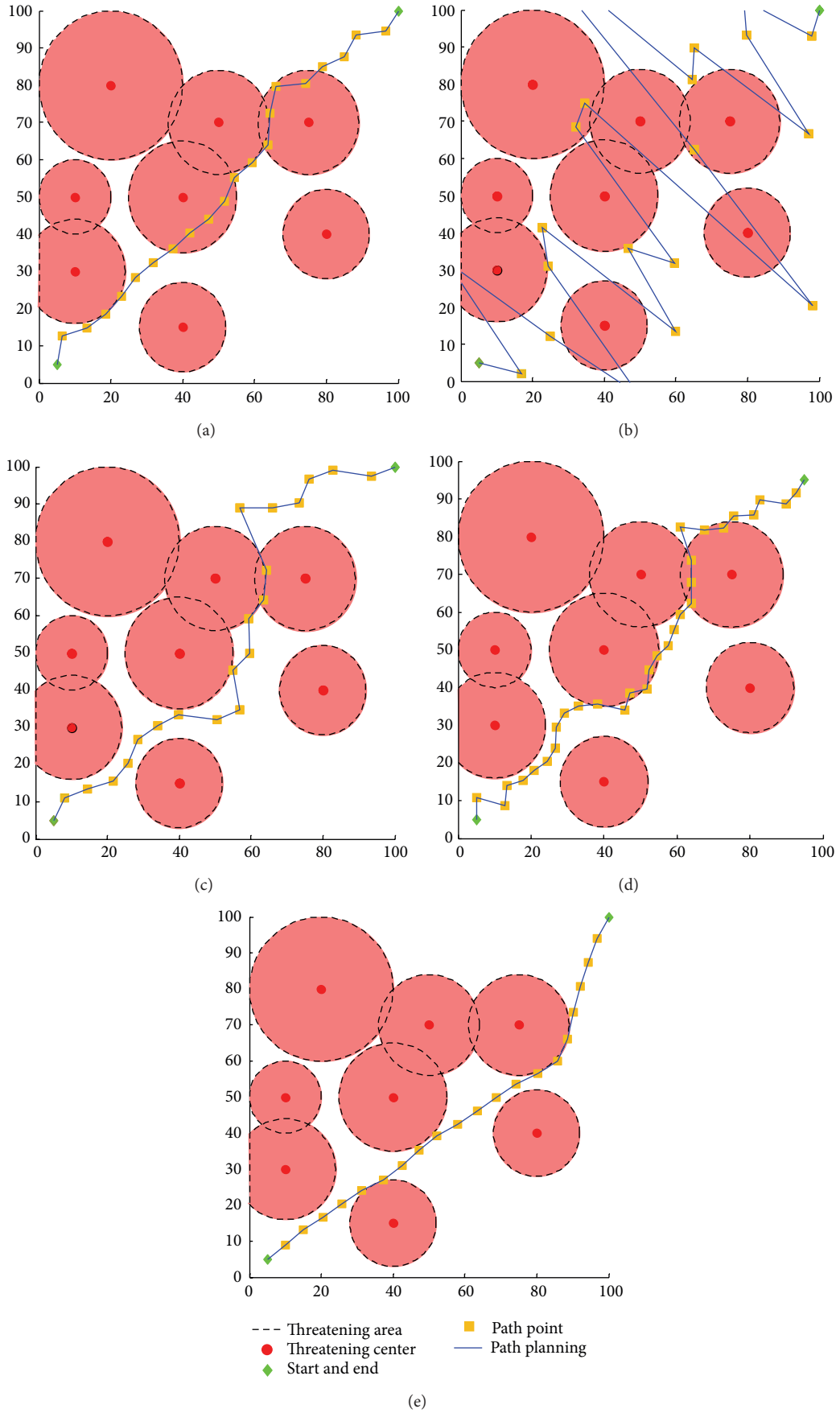

Figure 8: Planned path of (a) EGAM; (b) SA; (c) CABC; (d) PSO; and (e) FAC-PSO.

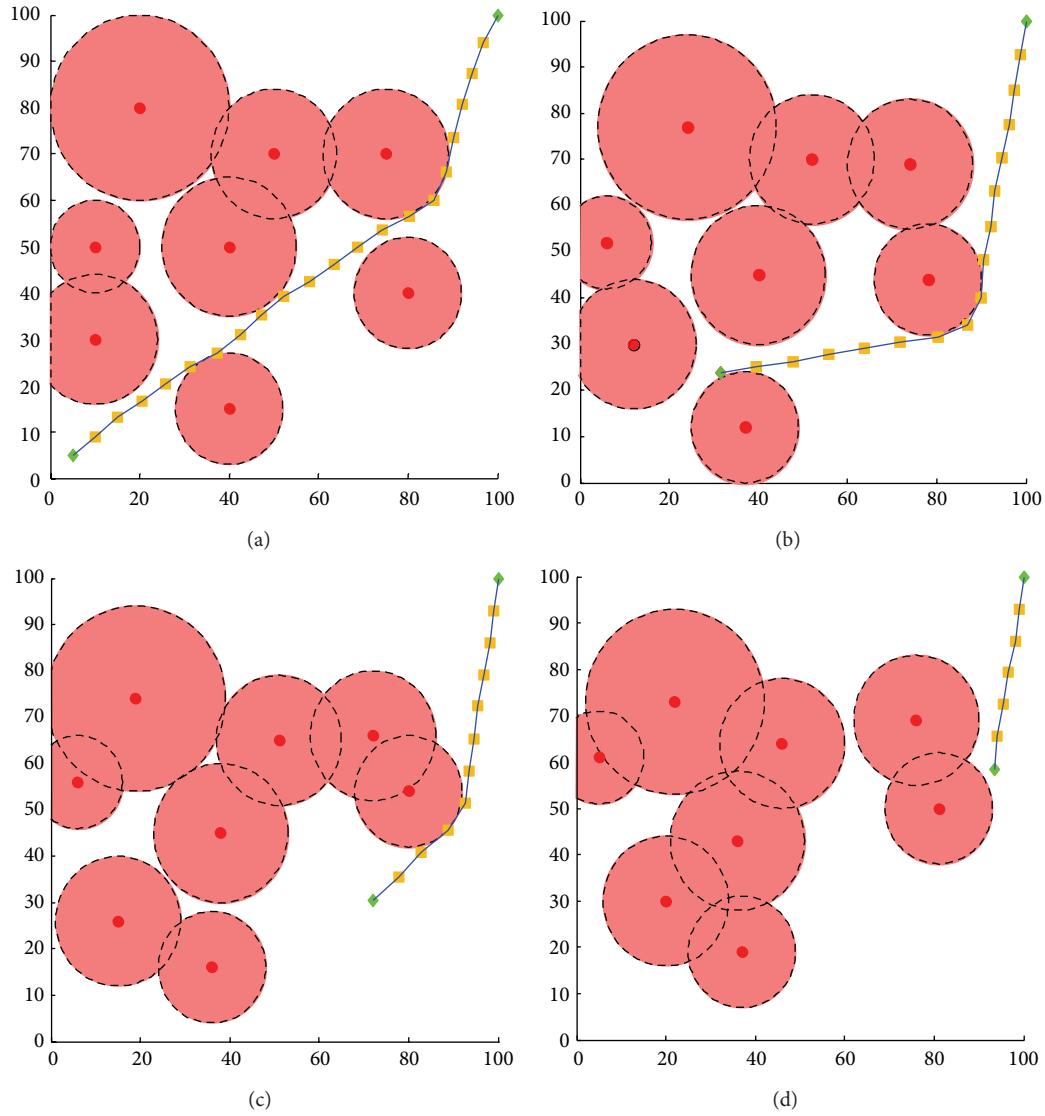

Figure 9: All threatening obstacles are moving for dynamic path planning: (a) step 0; (b) step 5; (c) step 10; and (d) step 15.

powerful at jumping out of local minima as well as speeding up the procedures of finding the global optimal minima.

The simulation results show that the proposed FAC-PSO excels EGAM, SA, CABC, and PSO algorithms with respect to success rate and computation time. We extended our experiment to 2D dynamic path planning and 3D path planning. All prove the superiority of FAC-PSO. Therefore, it is a feasible and effective way for UCAV path planning.

References

[1] F. C. J. Allaire, M. Tarbouchi, G. Labonté, and G. Fusina, "FPGA implementation of genetic algorithm for UAV real-time path planning," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1–3, pp. 495–510, 2009.

[2] H.-B. Duan, G.-J. Ma, and D.-L. Luo, "Optimal formation reconfiguration control of multiple UCAVs using improved particle swarm optimization," *Journal of Bionic Engineering*, vol. 5, no. 4, pp. 340–347, 2008.

[3] H. Duan, Y. Yu, X. Zhang, and S. Shao, "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm," *Simulation Modelling Practice and Theory*, vol. 18, no. 8, pp. 1104–1115, 2010.

[4] H.-B. Duan, X.-Y. Zhang, J. Wu, and G.-J. Ma, "Max-Min adaptive ant colony optimization approach to multi-UAVs coordinated trajectory replanning in dynamic and uncertain environments," *Journal of Bionic Engineering*, vol. 6, no. 2, pp. 161–173, 2009.

[5] C. Mou, W. Qing-xian, and J. Chang-sheng, "A modified ant optimization algorithm for path planning of UCAV," *Applied Soft Computing Journal*, vol. 8, no. 4, pp. 1712–1718, 2008.

[6] Y. Zhang, L. Wu, and S. Wang, "UCAV path planning based on FSCABC," *Information*, vol. 14, no. 3, pp. 687–692, 2011.

[7] D. Rodic and A. P. Engelbrecht, "Social networks in simulated multi-robot environment," *International Journal of Intelligent Computing and Cybernetics*, vol. 1, no. 1, pp. 110–127, 2008.

[8] 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.

[9] 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.

[10] Y. Zhang, L. Wu, G. Wei, J. Yan, and Q. Zhu, "Chaotic immune particle swarm optimization for multi-exponential fitting," *Journal of Southeast University (Natural Science Edition)*, vol. 39, no. 4, pp. 678–683, 2009.

[11] O. Begambre and J. E. Laier, "A hybrid Particle Swarm Optimization—simplex algorithm (PSOS) for structural damage identification," *Advances in Engineering Software*, vol. 40, no. 9, pp. 883–891, 2009.

[12] E. R. Tkaczyk, K. Mauring, A. H. Tkaczyk et al., "Control of the blue fluorescent protein with advanced evolutionary pulse shaping," *Biochemical and Biophysical Research Communications*, vol. 376, no. 4, pp. 733–737, 2008.

[13] A. M. Korsunsky and A. Constantinescu, "Work of indentation approach to the analysis of hardness and modulus of thin coatings," *Materials Science and Engineering A*, vol. 423, no. 1-2, pp. 28–35, 2006.

[14] Y. Wang, B. Li, and T. Weise, "Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems," *Information Sciences*, vol. 180, no. 12, pp. 2405–2420, 2010.

[15] T. Kellegöz, B. Toklu, and J. Wilson, "Elite guided steady-state genetic algorithm for minimizing total tardiness in flowshops," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 300–306, 2010.

[16] B. Agard and B. Penz, "A simulated annealing method based on a clustering approach to determine bills of materials for a large product family," *International Journal of Production Economics*, vol. 117, no. 2, pp. 389–401, 2009.