Pigeon-Inspired Optimization Approach to Multiple UAVs Formation Reconfiguration Controller Design*

Xiaomin Zhang, Haibin Duan, *Senior Member, IEEE,* and Chen Yang

***Abstract*—This article introduces a controller for solving the problem of multiple unmanned aerial vehicles (UAVs) formation reconfiguration. Under the constraints of terminal status and of control action energy, the controller aims to find the best values of UAV's inputs (including thrust, load factor, bank angle) to accomplish the task. The basis of our controller is a method which combines PIO with CPTD. Pigeon-Inspired optimization (PIO) is a novel algorithm. It has been proposed and applied in engineering problems by following the inspirational precious works. CPTD is a method called control parameterization and time discretization. Besides, we use a mathematical model to get a function which can evaluate the reconfiguration effects. Finally, to verify the validity of our controller, comparative experiments between PIO and particle swarm optimization (PSO) are conducted. The comparative results demonstrate that our proposed controller embedded with PIO is much better than the one with PSO.**

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have demonstrate their superior advantages in many fields, ranging from civil to military [1]. With rapid developing of military, UAV plays a significant role particularly on military operation i.e. a destroyer or reconnaissance [2]. In military applications, there are two major problems for UAV to accomplish a mission in the UAV. These two problems are control system and path planning system [2]. In control system, in order to let the UAV fly in formation, we need a controller to adjust UAVs' positions to reconfigure them in an expected formation. Thus, we need to build a multi-UAVs formation flight model first and then demand a method to plan UAVs' paths so that they can fly to relatively fixed positions along these paths and finish the formation reconfiguration task.

Xiong, et al. [3] proposed a 2D model based on Genetic Algorithm (GA) for solving the problem of UAV formation reconfiguration. After that, Duan, et al.[4] proposed a 3D model based on hybrid GA and PSO [5] to solve the problem. In [4], there exits free terminal constraints and distance restraints. The free terminal constraints guarantee the relative formation of UAVs. The distance between each two UAVs should be in an interval whose lower limit ensures the safety of UAVs and the upper limit ensures the unimpeded communication among multi-UAVs. Besides these constraints, we mainly focus on the terminal time. The smaller it is, the better the controller is. In this paper, a 3D model in [4] is used based on a brand new algorithm Pigeon-Inspired optimization (PIO).

PIO, proposed by Duan, et al. [6], is an intelligent artificial algorithm which imitates the behavior of homing pigeons. It is based on the researches of pigeons [7-9]. Inspired by these researches, PIO, uses two operators to imitate the pigeons to get home. One is map and compass operator, the other is landmark operator. In this paper, we embed PIO in our controller to search for the inputs and paths of multi-UAVs.

In the meantime, since we cannot give out the inputs of UAVs at every time, we contrive to partition the whole time into several pieces. What we need to do is giving inputs' values at the beginning of every piecewise periods of time. So in our controller, PIO is evolved by combining with a method called control parameterization and time discretization (CPTD). CPTD is presented by Furakawa et al. [10]. It contains three principles. With them, the control time can be discretized.

Combining PIO with CPTD, we obtain our controller's algorithm. And in order to verify the validity of our controller, we do some experiments to observe whether UAVs can finish the formation reconfiguration task under our control. The experimental result illustrates that UAVs can reconfigure a formation that we expect. Beyond that, we also use PSO solving this problem to compare with PIO. This standard of comparison is according to the 3D model in [4]. In the model, we can get a punishing function named objective function [4]. The smaller the value of objective function algorithm can get, the better performance the algorithm does. After experiments, we find that PIO can get a smaller value of this function than PSO can, which illustrates the superiority of PIO.

This section is the introduction of the whole passage. The rest of paper is structured as follows. In section II , the mathematical model is presented. Meantime, the objective function is given. In section III, PIO is introduced. In section IV presents how to discretize our controller by CPTD. Section V includes our controller's inside programming. In section VI, the result is given out. Finally, the conclusion is drawn in section VII.

II. MATHEMATICAL MODEL

Firstly, we assume that t=0 is the start time point in a reconfiguration task and t=T is the end time point. The task will be finished by N UAVs. And. Define $\boldsymbol{U}$ as the controller inputs. Because there are N UAVs, $\boldsymbol{U}$ contains N dimensional vectors i.e. $\boldsymbol{U}=(u_1,...,u_N)$.

*This work was supported by National Key Basic Research Program of China (973 Project) under grant #2014CB046401, and Aeronautical Science Foundation of China under grant No.20135851042 .

Xiaomin Zhang is with School of Advanced Engineering, Beihang University, Beijing, China (e-mail: zxm5-11@163.com).

Haibin Duan is with Science and Technology on Aircraft Control Laboratory, School of Automation Science and Electrical Engineering, Beihang University, Beijing, China (e-mail: hbduan@buaa.edu.cn; phone: 86-10-8231-7318).

Chen Yang is with School of Advanced Engineering, Beihang University, Beijing, China (e-mail: ychen3718@126.com)

978-1-4799-4699-0/14/$31.00©2014 IEEE

$$u_i = \{u_i(t) \mid \forall t \in [0,T]\} \in \Re^{r_i}, \forall i \in \{1,...,N\} \quad (1)$$

where u_i is the inputs of the i^{th} UAV which includes thrust, load factor and bank angle. The continuous form of $\boldsymbol{U}$ is as (2).

$$\boldsymbol{U} = (u_1,...,u_N) = \{\boldsymbol{U}(t) \mid \forall t \in [0,T]\} \quad (2)$$

The UAV's state can be defined as (3):

$$\boldsymbol{X}_i = [v_i, \gamma_i, \chi_i, x_i, y_i, z_i]^T \in \Re^6, \forall i \in \{1,...,N\}. \quad (3)$$

where (x_i, y_i, z_i) is the 3D coordinates. v_i is the i^{th} UAV's airspeed, γ_i is the flight path angle and χ_i is the heading angle [4]. Thus, the formation system state can be defined as $\boldsymbol{X}=(x_1^{\mathrm{T}},..., x_N^{\mathrm{T}})^{\mathrm{T}} \in \Re^{6N}$. The motion equation of formation can be described as:

$$\dot{\boldsymbol{X}}(t) = f(t, \boldsymbol{X}(t), \boldsymbol{U}(t)) \quad (3)$$

If the input and the initial state $\boldsymbol{X}(0) = \boldsymbol{X}_0$ is given out, then the status at every time point $t \in (0,T]$ can be obtained by (4).

$$\boldsymbol{X}(t) = \boldsymbol{X}(0) + \int_0^t f(\tau, \boldsymbol{X}(\tau), \boldsymbol{U}(\tau))d\tau \quad (4)$$

This is to say if the initial status is fixed, then $\boldsymbol{X}(t)$ can be determined only by $\boldsymbol{U}$ i.e. $\boldsymbol{X}(t) = \boldsymbol{X}(t \mid \boldsymbol{U})$.

Usually, the standard form of objective function can be described as follow [4].

$$J(\boldsymbol{U}) = \Phi_0(\boldsymbol{X}(T \mid \boldsymbol{U})) + \int_0^T L_0(t, \boldsymbol{X}(t \mid \boldsymbol{U}), \boldsymbol{U}(t))dt \quad (5)$$

The standard form of constraints can be depicted as [4].

$$\begin{cases} g_i(\boldsymbol{U}) = \Phi_i(\boldsymbol{X}(\tau_i \mid \boldsymbol{U})) + \int_0^{\tau_i} L_i(t, \boldsymbol{X}(t \mid \boldsymbol{U})), \boldsymbol{U}(t))dt \le 0 \\ \forall i \in \{1,...,M\} \end{cases} \quad (6)$$

For formation controller, its task is equal to searching for an input and terminal time point T to get the minima of the objective function $J(\boldsymbol{U})$ [4].

$$\min_{u_1,T} ... \min_{u_N,T} J(\boldsymbol{U}) \quad (7)$$

$$\min J(\boldsymbol{U}, T). \quad (8)$$

Since the inputs cannot be infinitely great or infinitely little, inputs are limited in an interval. The restriction of the controller input is denoted as (9).

$$\boldsymbol{U}_{\min} \le \boldsymbol{U}(t) \le \boldsymbol{U}_{\max}, \forall t \in [0,T), T > 0 \quad (9)$$

The free terminal constraint is denoted as [4].

$$\begin{aligned} g_1(\boldsymbol{U}, \Delta t) = \sum_{i=1}^{N} \{ & [(x_i(T) - x_m(T)) - x_i^m]^2 \\ & + [(y_i(T) - y_m(T)) - y_i^m]^2 \\ & + [(z_i(T) - z_m(T)) - z_i^m]^2 \} = 0 \end{aligned} \quad (10)$$

In (10), $m \in \{1,...,N\}$,and set the m^{th} UAV as the center. Then, $[x_i^m, y_i^m, z_i^m]^T$ represents the optimal relative position of i^{th} UAV from the center.

The distance between i^{th} UAV and j^{th} UAV can be described as (11).

$$d^{i,j}(x_i(t), x_j(t)) = \sqrt{(x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2 + (z_i(t) - z_j(t))^2} \quad (11)$$

where D_{safe} and D_{comm} give out the constraints of d, which can be expressed as (12) and (13).

$$d^{i,j}(x_i(t), x_j(t)) \ge D_{safe} \quad (12)$$

$$d^{i,j}(r(t), m(t)) \ge D_{comm} \quad (13)$$

where D_{safe} means the minimum of d. If $d < D_{safe}$, two UAVs will crash into each other. While D_{comm} means the maximum of d. If $d > D_{comm}$, UAVs will lose contact with each other.

III. PIO ALGORITHM

Because homing pigeons have special ability that they can find their way home themselves, people take advantages of them in many fields, for example, news communication, sports communication, marine communication and military communication.

In fact, there are lots of researches studying pigeons' special ability[7-9]. They claimed that pigeons use a combination of the sun, the earth's magnetic field and landmarks to find their way around [6]. Inspired by these researches, PIO has been proposed by Duan, et al.[4]. This algorithm has two operators imitating the pigeons' behavior, which are Map and compass operator and Landmark operator. The detailed information of these two operators are described as follow.

A. Map and compass operator

In this operator, pigeons are assumed to sense the earth field by magnetoreception and then build a map. Using the compass, the pigeons can readjust the navigation direction so that they can fly to the destination quickly.

In certain time pigeons have their position P_i and velocity V_i. In i^{th} iteration, the two parameters of i^{th} pigeon are updated by following equations [6].

$$V_i(t) = V_i(t-1)e^{-Rt} + rand \cdot (P_g - P_i(t-1)) \quad (14)$$

$$P_i(t) = P_i(t-1) + V_i(t) \quad (15)$$

where R is the map and compass factor and P_g is the current global best position.

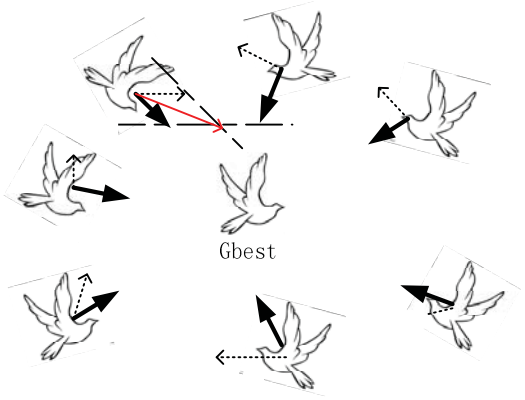

Figure 1 Map and Compass Operator

In Figure 1, pigeons' speed is decided by two parts. One is described by the heavy continuous arrow. It means a component which illustrates that pigeons try to fly toward the best one. This part can drive the pigeons to fly close to the best positions. And the best position is by now the closest to the destination. The other is a component drawn by thin dotted arrow. This component is influenced by the factor $V_i(t-1)e^{-Rt}$, which represents the compass effect and R is the map and compass factor. This component will become smaller with the increase of iteration and it can prevent pigeons from flying to local optimum to some extent. The composite vectors are the final velocities.

B. *Landmark operator*

In this operator, some pigeons which are closer to the final goal are reserved. They will count on landmarks near them to fly directly to the destination if they are familiar with the landmarks enough.

Based on the map and compass operator model, pigeons have already flown near the destination. Then, they go more directly and precisely to where they want with the landmark. The center of the present pigeons $P_c(t)$ which is regarded as a landmark is defined by equation (17) (See Figure 2). In each generation, the number of pigeons $M(t)$ reduces by half. The remaining pigeons are more closer to the center compared to the excluded pigeons. And then pigeons fly directly to the center. Equations (16), (17) and (18) show the landmark operator model process [6].

$$M(t) = \frac{M(t-1)}{2} \tag{16}$$

$$P_c(t) = \frac{\sum P_i(t) \cdot J(P_i(t))}{\sum J(P_i(t))} \tag{17}$$

$$P_i(t) = P_i(t-1) + rand \cdot (P_c(t) - P_i(t-1)) \tag{18}$$

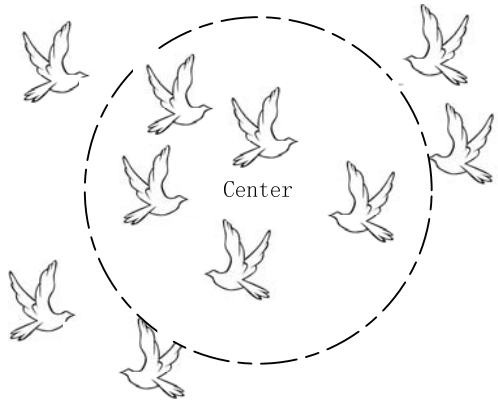

Figure 2 Landmark Operator

In Figure 2, there is a center pigeon defined by equation (17). Pigeons in the circle are included to continue their trip. However, pigeons out of the circle are excluded because they are far away from the center.

IV. DISCRETIZING CONTROLLER WITH CPTD

If we still use a continuous model to solve the formation reconfiguration problem, the controller need to give out the inputs at every iteration. It is infeasible for PIO to do this and it is also too complicated. In order to solve the problem easily with PIO, we need to discretize the input. With the simplified input, we can get the status and calculate the cost function so that the minima of the costs with different inputs can be obtained.

A. *Principles in CPTD*

There are three principles in CPTD.

- Divided the total time T of the input into n_p parts in average. Let the piecewise function replace the continuous inputs. In each part, the inputs are regarded as constants in each interval Δt_p.
- Set T a function of Δt_p.
- Through combining with nonlinear algorithm, get the minima of cost function and search for the value of Δt_p. Further, the inputs u_i corresponding to every Δt_p become the values that we search for of the minimum cost function.

Thus, with the method of CPTD, our control means is divided into three steps.

- **Division of control time *T*** Divide the control time T into n_p parts in average. $n_p = \{1, 2, ...\}$. Every part is equal to $\Delta t_p \in \Re^+$. Thus, equation (19) can be given out.

$$T = n_p \Delta t_p \tag{19}$$

In each Δt_p, do numerical integration and interval operation based on the homologous inputs and equation (3).

- **Piecewise linearization of inputs** Because control time is divided into n_p parts, we can define a collection whose elements are all constants. This collection is named $\boldsymbol{Q}$. And $\boldsymbol{Q} = \{\boldsymbol{Q}_1, ..., \boldsymbol{Q}_N\}$

  $\boldsymbol{Q}_i$ is the i^{th} UAV's inputs.

$$\boldsymbol{Q}_i \triangleq \{q_j^i \in \boldsymbol{R}^{r_i} \mid \forall j\}$$

Therefore, the inputs of the i^{th} UAV can be described with piecewise function as (20) [4].

$$\hat{\boldsymbol{u}}_i\left(t; n_p, \boldsymbol{Q}_i\right) = \sum_{j=1}^{n_p} q_j^i \varepsilon_j(t), \boldsymbol{Q}_i \cong \boldsymbol{u}_i(t) \quad (20)$$

where $\varepsilon_j(t)$ is defined as (21) [4].

$$\varepsilon_j(t) = \begin{cases} 1 & (j-1)\Delta t_p \le t \le j\Delta t_p \\ 0 & otherwise \end{cases} \quad (21)$$

Further, the piecewise linearization constant collection of the formation can be defined as (22). The approximate input collection of the formation can be defined as (23) [4].

$$\boldsymbol{Q}_i = \begin{bmatrix} q_{11}^i & q_{21}^i \cdots & q_{n_p 1}^i \\ \cdots & \cdots & \cdots \\ q_{1r_i}^i & q_{2r_i}^i \cdots & q_{n_p r_i}^i \end{bmatrix} \quad (22)$$

$$\hat{\boldsymbol{U}}(\mathrm{t}; n_p, \boldsymbol{Q}) = \{\hat{\boldsymbol{u}}_1\left(t; n_p, \boldsymbol{Q}_1\right), ..., \hat{\boldsymbol{u}}_N\left(t; n_p, \boldsymbol{Q}_N\right)\} \quad (23)$$

Our goal is to find a $\hat{\boldsymbol{U}}$ which can let the value of objective function be the smallest. That is to say we contrive to find $\boldsymbol{Q}$ and n_p which can minimize the value of objective function. In fact, the value of n_p influences the result the most. With the increasing of n_p, the discrete inputs are more approximate to the continuous one but the calculation time will increase at the same time. Thus, a proper n_p should be chosen.

- **Parameterization** The problem of searching for input $\hat{\boldsymbol{U}}$ and T which can minimize the objective function is translated into searching for the best value of $\boldsymbol{Q}$ and Δt_p. Thus the objective function can be redefined as (24) [4].

$$J_{\text{extend}} = \min_{\boldsymbol{Q}, \Delta t_p} \{(n_p \Delta t_p)\} \quad (24)$$

And the free terminal constriction is approximate to (25) [4].

$$\begin{aligned} \hat{g}_1(\boldsymbol{Q}, \Delta t) = \sum_{i=1}^{N} \{ & [(x_i(T) - x_m(T)) - x_i^m]^2 \\ & + [(y_i(T) - y_m(T)) - y_i^m]^2 \\ & + [(z_i(T) - z_m(T)) - z_i^m]^2\} = 0 \end{aligned} \quad (25)$$

The status equation can be approximate to (26) [4].

$$\dot{\boldsymbol{X}}(t) = f(t, \boldsymbol{X}(t), \hat{\boldsymbol{U}}(\mathrm{t}; n_p, \boldsymbol{Q})) \quad (26)$$

As for other constraints, keep the constraint conditions as mentioned before.

B. Time-Optimization controller based on CPTD

- **Redefining Pigeons in PIO** According to the definition of $\boldsymbol{Q}$ and Δt_p, we can redefine pigeons as (27).

$$P = \{\boldsymbol{Q}_1, ..., \boldsymbol{Q}_N\} \quad (27)$$

where $\boldsymbol{Q}_i$ is described as (22).

- **Initialization** According to equation (14) to (18), initialize the positions and velocities of pigeons.
- **Fitness Calculation** The redefined objective function named JE can be expressed as (28) [4].

$$\begin{aligned} J_{\text{extend}} \triangleq \min_{\boldsymbol{Q}, \Delta t_p} \{ & (n_p \Delta t_p) + \delta_1 \hat{g}_1(\boldsymbol{Q}, \Delta t) + \\ & \delta_2 \cdot \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} [\max(0, D_{\text{safe}} - d^{i,j}(\boldsymbol{x}_i(t), \boldsymbol{x}_j(t))) \\ & + \max(0, d_{i,j}(\boldsymbol{x}_i(t), \boldsymbol{x}_j(t)) - D_{\text{comm}})] \\ & + \delta_3 \sum_{k=1}^{N} \sum_{l=1}^{n_p} [\max(0, \max(\mathrm{q}_k^{\,l} - (\boldsymbol{u}_{\max})_k)) \\ & + \max(0, \max((\boldsymbol{u}_{\min})_k - q_k^{\,l}))] \} \end{aligned} \quad (28)$$

where δ_1, δ_2, δ_3 are respectively the punishment coefficient of terminal constraint, distance restriction and piecewise linearization inputs' restriction. And δ_1, δ_2, δ_3 should be large enough to punish the behavior of violating the constraints.

V. Programming Description

For programming, PIO is basically divided into four parts as following.

A. Initialization

Initialize: $T_1 = 950$, $T_2 = 50$, $M = 160$, $R = 0.003$.

Set the searching range.

Set initial pigeons' positions P_i and velocities V_i.

Set the expected relative positions.

Set $gbest = \min(JE(P_1), JE(P_2), ..., JE(P_M))$.

$P_g = P_i$

Calculate fitness.

B. Map and compass operator model

FOR $t = 1$ to T_1

FOR $i = 1$ to M

Update P_i and V_i according to equation (14) and (15).

Calculate fitness value of P_i

IF $JE(P)_i < JE(P_g)$

Set $P_g = P_i, gbest = JE(P_g)$

END IF

END FOR

END FOR

C. Landmark operator model:

FOR $t = T_1 + 1$ to $T_1 + T_2$

Rank all pigeons according to their fitness values

Set $M(t) = \frac{M(t-1)}{2}$

Abandon the worse half of pigeons.

Calculate the present center of the pigeons $P_c(t)$ according to equations (10).

FOR $i = 1$ to M

Update P_i and V_i according to equations (7) and equations (8).

Calculate fitness values of P_i

IF $JE(P)_i < JE(P_g)$

Set $P_g = P_i$, $gbest = JE(P_g)$.

END IF

END FOR

END FOR

D. Output

Output the best value P_g.

VI. Experimental Results

To verify the validity of our controller embedded with PIO, simulation experiment has been carried out and Figure 3 and 4 show the results.

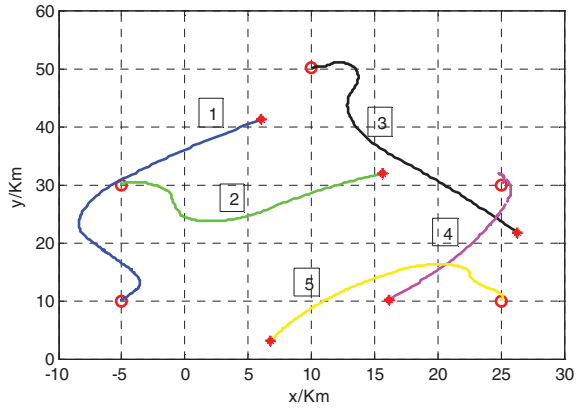

Figure 3 Result of xy plane by PIO

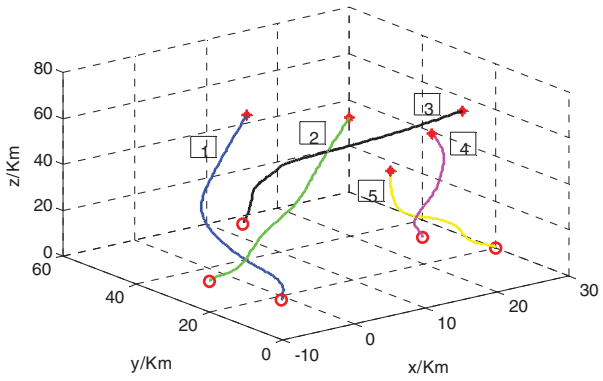

Figure 4 UAV paths of 3D by PIO

In Figure 3 and 4, the result of xy plane has been given out. Points drawn of '○' represents the start point. Points drawn of '∗' represents the end point. Initially, we set the formation style as a pentagon. Five UAVs' coordinates are (x1,y1,z1)=(-5,10,0),(x2,y2,z2)=(-5,30,0),(x3,y3,z3)=(10,50,0),(x4,y4,z4)=(25,30,0),(x5,y5,z5)=(25,10,0). Regard the 3rd UAV as the center and we look forward other UAVs arriving at relative positions as follows: (x1',y1',z1')= (-20,20,0), (x2',y2',z2')=(-10,10,0),(x3',y3',z3')=(0,0,0),(x4',y4',z4')=(-10,-10,0), (x5',y5',z5')=(-20,-20,0). The final positions of our controller has achieved are: (x1,y1,z1)= (6.0807,41.1775,61.6058),(x2,y2,z2)=(16.1690,31.8742,61.3051),(x3,y3,z3)=(26.2639,21.7390,61.2483), (x4,y4,z4)=(16.1657,11.2001,61.2156),(x5,y5,z5)= (6.1058,1.2703,61.1198). The final relative positions of five UAVs from the 3rd UAV are: (-20.1832,20.0385,0.0574),(-10.0949,10.1353,0.0568),(0,0,0),(-10.0983,-10.5389,-.0327),(-20.1581,-20.3686,-0.1285). So the UAVs finally fly very approximately to the places where they are expected. This illustrates our controller can manage to control UAVs to let them arrive at the final relative positions and form a style almost like '>'. This experiment verifies the validity of our controller embedded with PIO.

Besides, in order to investigate the superiority of PIO, we try PSO in our controller. Beyond this, we carry out the experiment in the same condition. Although the initial start points are different, the relative positions are the same. The absolute value of positions won't influence the performance of algorithm. The result is depicted in Figure 5 and 6.

From Figures 5 and 6, we can see the result of PSO controller cannot arrive at the positions as expected. In 3D figures, PSO cannot make UAVs keep in the same height while PIO manage to do it. In xy plane, PIO can get a shape of '>' while PSO cannot. This illustrates PIO's superiority.

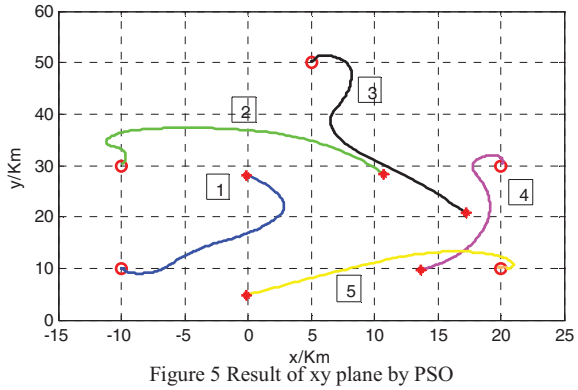

Figure 5 Result of xy plane by PSO

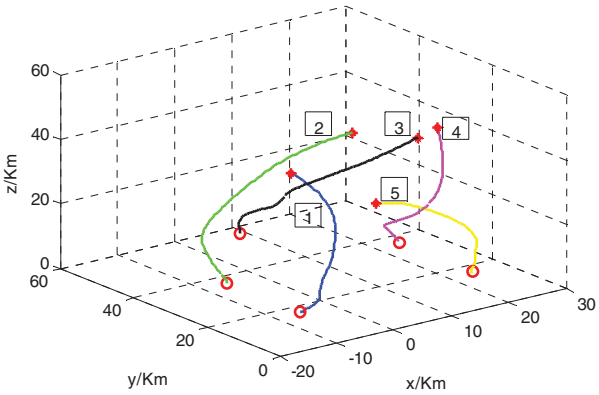

Figure 6 UAV paths of 3D by PSO

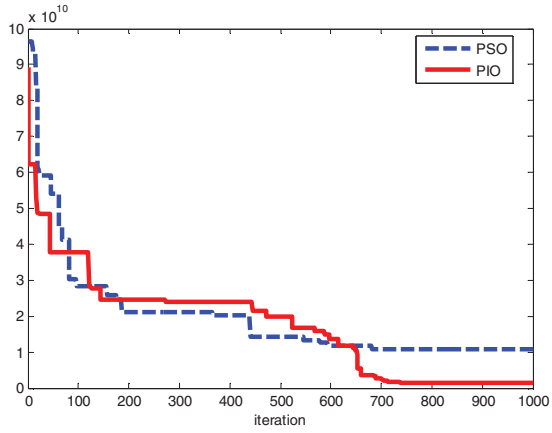

Figure 7 The convergence curves of PIO and PSO

Figure 7 shows the convergence curves of PIO and PSO. PIO has stronger ability to search for the minima of objective function ***J***. It has the ability to find smaller value in large range. In 600 to 800 iterations, PSO cannot get a smaller value but PIO can. Finally PIO find a value of punish function ***J*** smaller than PSO. It reaches the value of $1.27e10^9$ at last. Thus, our controller embedded with PIO is better.

VII. Conclusion

In this paper, we designed a UAV formation reconfiguration controller based on PIO algorithm. A 3D model is adopted and an objective function is established. In order to simplify the problem of formation reconfiguration, we use CPTD to discretize our controller. By comparative experimental results, it is obvious that our newly designed controller is feasible and effective.

References

[1] Z. Sarris and S. Atlas, "Survey of UAV applications in civil markets," in *Proceedings of the 9th Mediterranean Conference on Control and Automation*, Dubrovnik, Croatia, 2001, pp. 1–11.

[2] H. H. Triharminto, T. B. Adji, N. A. Setiawan, "Dynamic UAV path planning for moving target intercept in 3D," *Proceedings of the 2nd International Conference on Instrumentation Control Automation,* Bandung, Indonesia., 2011, pp. 15-17.

[3] W. Xiong, Z. Chen and R. Zhou, "Optimization of multiple flight vehicle formation reconfiguration using hybrid genetic algorithm,"in *Proceedings of the 1st Chinese Guidance, Navigation Control Conference*, Beijing, China, 2007, pp. 501 - 506.

[4] H. Duan, Q. Luo, G. Ma, Y. Shi, "Hybrid particle swarm optimization and genetic algorithm for multi-UAV formation reconfiguration," *IEEE Computational Intelligence Magazine,* 2013, vol.8, no.3, pp.16-27.

[5] J. Kennedy, R.Eberhart "Particle Swarm optimization". *Proceedings of IEEE International Coference on Neural Networks*, 1995:1942-1948.

[6] H. Duan, P. Qiao, "Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning," *International Journal of Intelligent Computing and Cybernetics*, 2014, vol. 7, no.1, pp.24-37.

[7] T Guilford, S Roberts, D Biro. Positional entropy during pigeon homing II: navigational interpretation of Bayesian latent state models. *Journal of Theoretical Biology*, 2004, vol. 227, no. 1, pp. 25-38.

[8] Mora, C Davison, J Wild, M Walker. Magnetoreception and its trigeminal mediation in the homing pigeon, *Nature*, 2004, vol. 432, no. 7016, pp. 508 -511.

[9] Whiten. Operant study of sun altitude and pigeon navigation. *Nature*, 1972, vol. 237, no.5355, pp. 405-406.

[10] T. Furukawa, H. F. Durrant-Whyte, F. Bourganlt, and G. Dissanayake, "Timeoptimalcoordinated control of the relative formation of multiple vehicles," in *Proceedings of IEEE International Symposium Computational Intelligence Robotics Automation*, Kobe, Japan, 2003, pp.259 - 264.