Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Design and verification of a transfer path optimization method for an aircraft on the aircraft carrier flight deck

Weichao SI‡, Tao SUN, Chao SONG, Jie ZHANG

Coastal Defense College, Naval Aviation University, Yantai 264001, China

E-mail: luckydevilsi@163.com; luckydevilhan@163.com; sxwxc.1984@163.com; zhangjie9886@126.com

Received May 25, 2020; Revision accepted Oct. 8, 2020; Crosschecked Aug. 8, 2021

Abstract: This paper studies the transfer path planning problem for safe transfer of an aircraft on the aircraft carrier flight deck under a poor visibility condition or at night. First, we analyze the transfer path planning problem for carrier-based aircraft on the flight deck, and define the objective to be optimized and the constraints to be met. Second, to solve this problem, the mathematical support models for the flight deck, carrier aircraft entity, entity extension, entity posture, entity conflict detection, and path smoothing are established, as they provide the necessary basis for transfer path planning of the aircraft on the aircraft carrier. Third, to enable automatic transfer path planning, we design a multi-habitat parallel chaos algorithm (called KCMPSO), and use it as the optimization method for transfer path planning. Finally, we take the Kuznetsov aircraft carrier as a verification example, and conduct simulations. The simulation results show that compared with particle swarm optimization, this method can solve the transfer path planning problem for an aircraft on the aircraft carrier flight deck better.

Key words: Carrier aircraft; Flight deck; Transfer path planning; KCMPSO algorithm; Method design and validation
https://doi.org/10.1631/FITEE.2000251 **CLC number:** TP399

1 Introduction

Aircraft carrier is an important mobile platform for high-sea combat and symbolizes a country's overall national strength. A large part of the combat effectiveness of an aircraft carrier depends on the combat capability of its carrier-based aircraft (Yang et al., 2018). To optimize the combat capability of these aircrafts to the greatest degree, the aircrafts' safety during non-combat processes, such as takeoff, landing, and flight deck movements, must be given high priority. Regardless of an aircraft's takeoff or recovery, it is necessary to move the aircraft first (He et al., 2019). Therefore, ways to ensure the safety and efficiency of the aircrafts during their movement on the flight deck is a subject that needs further study.

At present, when an aircraft moves on the flight deck, it needs to rely on its own power or tractor, but either way, it needs people's participation (Zhao, 2019). Given the limited area of the flight deck and the presence of many other aircrafts, there is no guarantee that the aircraft will not collide with other aircrafts during the transfer process due to the commander's or pilot's blind spots, especially at night or in weather with poor visibility. For example, since 2013, the US has experienced an upward trend in military aircraft accidents caused by human factors that cannot be underestimated (Chen, 2018). In addition, efficiency is low and the walking path is not necessarily the best when these aircrafts are under manual command for transfer. For this reason, this work focuses on the research related to the carrier-based aircraft transfer path optimization problem.

2 Analysis of the transfer path planning problem

Due to the high density of aircrafts parked on the limited-area flight deck, the environment is relatively

‡ Corresponding author
ORCID: Weichao SI, https://orcid.org/0000-0002-5257-7384

complex. When an aircraft is moved on the flight deck, it must maintain a certain safe distance from other aircrafts to avoid collision. All the problems to be solved involve dynamically planning an optimal path from the current starting point to the end point in real time on the premise of avoiding any collision between shipboard aircrafts (Ryan et al., 2014).

Assuming that the current starting point ***s***, the end point ***e***, and the obstacle area of the current flight deck are known, the problem to be solved is to search for a series of path points in the feasible two-dimensional (2D) area without collision within the obstacle area, so that the path composed of these points is approximately optimal in the whole region (such as the shortest distance). The mathematical model is established as follows:

Objective function:

$$\begin{aligned} F &= \min\{S(\boldsymbol{A}_i)\} \\ &= \min\left\{\sum_{j=1}^{n-1}\sqrt{(x_{i,j+1}-x_{i,j})^2+(y_{i,j+1}-y_{i,j})^2}\right\}, \end{aligned} \tag{1}$$

where

$$\begin{aligned} \boldsymbol{A}_i &= (\boldsymbol{a}_{i,1},\boldsymbol{a}_{i,2},\cdots,\boldsymbol{a}_{i,n}) \\ &= \big((x_{i,1},y_{i,1}),(x_{i,2},y_{i,2}),\cdots,(x_{i,n},y_{i,n})\big). \end{aligned}$$

The goal is to find the shortest transfer path for the total moving routes of the aircraft without collision. S is the function for the moving distance of the plane via a certain path. $\boldsymbol{A}_i$ is the i^{th} movement path. $\boldsymbol{a}_{i,j}$ is the j^{th} intermediate node in the i^{th} mobile path, whose coordinates are $(x_{i,j}, y_{i,j})$. n is the number of nodes in the path.

Restraint conditions:

$$\boldsymbol{a}_{i,1}=(x_{i,1},y_{i,1}),\quad \boldsymbol{a}_{i,n}=(x_{i,n},y_{i,n}), \tag{2}$$

$$\boldsymbol{a}_{i,j}\notin D_z,\ \ j=1,2,\cdots,n, \tag{3}$$

where D_z refers to all the obstacle areas in the current state. Eq. (3) shows that any node of the planned path cannot be located inside the obstacle area; that is, collision is not allowed.

$$l_{i,j}\geq l_{\min},\ \ j=1,2,\cdots,n-1, \tag{4}$$

where $l_{i,j}$ is the length of the j^{th} segment of the i^{th} path, and $l_{\min}$ is the minimum segment length of each movement of the aircraft, which can be defined as half of the fuselage length. Eq. (4) shows that the minimum straight-line moving distance must be maintained by the aircraft during the next turn. It is generally not expected that the aircraft will turn frequently.

$$\sum l_{i,j}\geq L_{\max},\ \ j=1,2,\cdots,n-1, \tag{5}$$

where $L_{\max}$ is the maximum length of the path between the starting point and the end point, which can be set as two times the straight-line distance between them.

$$\begin{aligned} &\frac{\boldsymbol{\Delta}_{i,j}^{\mathrm{T}}\boldsymbol{\Delta}_{i,j+1}}{\left\|\boldsymbol{\Delta}_{i,j}^{\mathrm{T}}\right\|\cdot\left\|\boldsymbol{\Delta}_{i,j+1}\right\|}\geq\cos\phi,\ \ j=1,2,\cdots,n-1, \\ &\boldsymbol{\Delta}_{i,j}=(x_{i,j}-x_{i,j-1},y_{i,j}-y_{i,j-1})^{\mathrm{T}}, \end{aligned} \tag{6}$$

where $\boldsymbol{\Delta}_{i,j}$ is the j^{th} segment vector of the i^{th} path, $\|\boldsymbol{\Delta}_{i,j}\|$ is the length of $\boldsymbol{\Delta}_{i,j}$, and ϕ is the maximum turning angle when the aircraft moves. Eq. (6) shows that the planned path must meet the requirements of the maximum turning angle of the carrier-based aircraft to avoid detours and sharp turns (Liu and Liu, 2017).

3 Design of mathematical support models

3.1 Aircraft carrier flight deck model

The aircraft carrier plane is expanded into a rectangular area, and the unreachable area is regarded as a special obstacle area, which is filled with black as shown in Fig. 1 (Qi and Wang, 2016). By filling the irregular aircraft carrier flight deck with a regular rectangular area, the selection of intermediate nodes can be simplified.

The size of the aircraft carrier flight deck model is 1000 px×246 px (here, px is short for pixel) (Si et al., 2012).

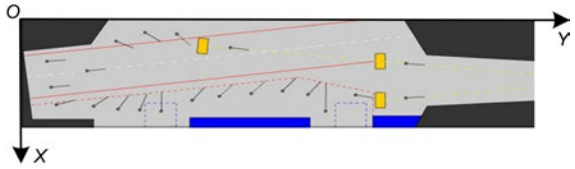

Fig. 1 Flight deck model

3.2 Carrier aircraft entity model

Different aircraft carriers carry different types of aircrafts, which have different shapes and need to be modeled separately. To reduce the computational complexity, we use a convex polygon to model the outline of the carrier plane (Si et al., 2015). According to the actual situation, this study focuses on the establishment of the following types of carrier-based aircraft entity models:

Fig. 2 shows a first-class carrier-based aircraft, which is a fixed-wing aircraft. Relative to the flight deck scale, its size is defined as G_{1a}=20.1 px, G_{1b}=48 px, and G_{1c}=33.9 px.

Fig. 3 shows the second-class carrier-based aircraft, which is a helicopter. Similarly, its size is defined as G_{1a}=9.6 px, G_{1b}=19 px, and G_{1c}=41 px.

Thus, no matter which type of carrier-based aircraft is modeled, its shape can be simplified to the outline shown in Fig. 4.

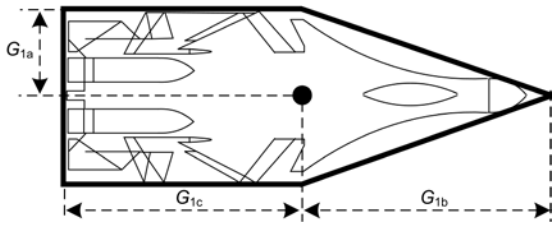

Fig. 2 Model of the first-class aircraft

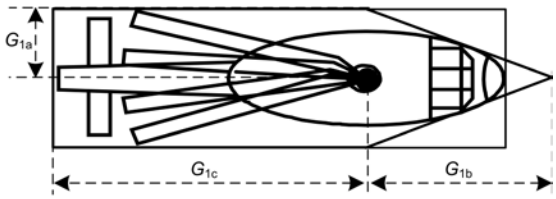

Fig. 3 Model of the second-class aircraft

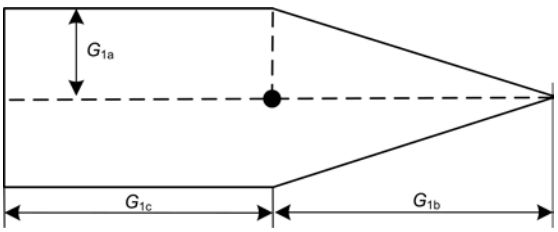

Fig. 4 Aircraft model

3.3 Entity extension model

To simplify the path planning, the carrier-based aircraft to be moved is simplified as a point A. The corresponding obstacles need to be extended according to the type of A (Luan, 2019). In extreme cases, A will not collide with obstacles when moving along the extended boundary line of the obstacles, as shown in Fig. 5.

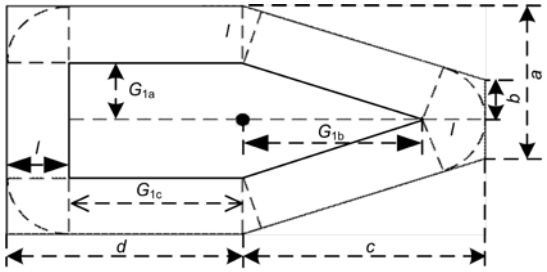

Fig. 5 Entity extension model

When transferring a first-class aircraft, the buffer distance of obstacles is l=G_{1a}=20.1 px. After calculation, a=(G_{1a}+20.1) px, b≈[(G_{1a}+20.1)/3] px, c=(G_{1b}+20.1) px, and d=(G_{1c}+20.1) px.

When transferring a second-class aircraft, the buffer distance of obstacles is l=G_{1a}=9.6 px. After calculation, a=(G_{1a}+9.6) px, b≈[(G_{1a}+9.6)/3] px, c=(G_{1b}+9.6) px, and d=(G_{1c}+9.6) px.

In addition, because a carrier-based aircraft must obey certain rules when parking on the flight deck, the obstacle area model for the aircraft can be simplified in the following two regards: a combination of carrier-based aircraft and flight deck and a combination of multiple carrier-based aircrafts that are close to each other.

3.4 Entity posture model

After determining the environment model, for each entity model, when the coordinates of its center point and the angle between its longitudinal axis and the positive direction of the X axis are known, its state (position, orientation, and etc.) can be uniquely determined in the environment model. In this case, 2D transformation is needed (Song, 2018).

To calculate the coordinates of each point of model B, whose center point is located at (x_0, y_0) and where the angle between the longitudinal axis of the fuselage and the positive direction of the X axis is θ, we can use model A whose center point is located at the original point. The coordinates of each point of model A can be calculated easily. Then, by rotating the angle and moving to (x_0, y_0) of model A, we can obtain model B (Fig. 6).

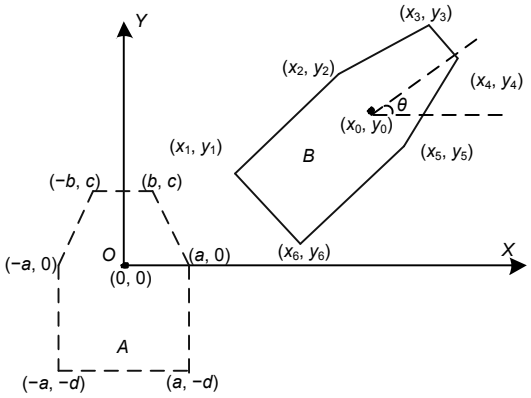

Fig. 6 Entity posture model

Suppose the coordinates of one point in model A are (x, y). Then after rotating and moving A, the new coordinate calculation process of point B is as follows:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \boldsymbol{T} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & x_0 \\ \sin\theta & \cos\theta & y_0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos\theta - y\sin\theta + x_0 \\ x\sin\theta + y\cos\theta + y_0 \\ 1 \end{pmatrix},$$

where

$$(x', y') = (x\cos\theta - y\sin\theta + x_0,\ x\sin\theta + y\cos\theta + y_0).$$

(x', y') are the new coordinates of (x, y). Thus, the coordinates of each point of model B can be given as follows:

$$\begin{aligned}
(x_1, y_1) &= (-a\cos\theta + d\sin\theta + x_0, -a\sin\theta - d\cos\theta + y_0),\\
(x_2, y_2) &= (-a\cos\theta + x_0, -a\sin\theta + y_0),\\
(x_3, y_3) &= (-b\cos\theta - c\sin\theta + x_0, -b\sin\theta + c\cos\theta + y_0),\\
(x_4, y_4) &= (b\cos\theta - c\sin\theta + x_0, b\sin\theta + c\cos\theta + y_0),\\
(x_5, y_5) &= (a\cos\theta + x_0, a\sin\theta + y_0),\\
(x_6, y_6) &= (a\cos\theta + d\sin\theta + x_0, a\sin\theta - d\cos\theta + y_0).
\end{aligned}$$

3.5 Entity conflict detection model

When planning the path, to ensure that the carrier-based aircraft does not collide with obstacles, it is necessary to perform conflict detection (Hao et al., 2018). In this study, we design two types of conflict detection, which are point conflict detection and path-segment conflict detection.

3.5.1 Point conflict detection

Point conflict detection is used mainly to determine whether the nodes of the planned path are located in other obstacles.

1. Identify the status of each obstacle in the current environment. Using the entity posture model, we can calculate the coordinates of each point of the obstacle. Furthermore, we can obtain its leftmost point coordinates $(x_{\text{left}}, y_{\text{left}})$, rightmost point coordinates $(x_{\text{right}}, y_{\text{right}})$, uppermost point coordinates $(x_{\text{upper}}, y_{\text{upper}})$, and lowermost point coordinates $(x_{\text{lower}}, y_{\text{lower}})$. Note that $x_{\text{left}}<x_{\text{right}}$ and $y_{\text{upper}}<y_{\text{lower}}$.

2. Obtain the current path node $M(a, b)$ which needs to be detected.

3. Preliminary screening of obstacles. To reduce the amount of calculation, we remove obstacles that are not in the detection range by screening. The principle is as follows:

If $x_{\text{left}}<a<x_{\text{right}}$ and $y_{\text{upper}}<b<y_{\text{lower}}$, then the $x=a$ and $y=b$ lines are passing through the obstacle. So, the obstacle must participate in further point conflict detection.

4. Perform point conflict detection between node $M(a, b)$ and each obstacle screened out. We take obstacle A as an example, as shown in Fig. 7.

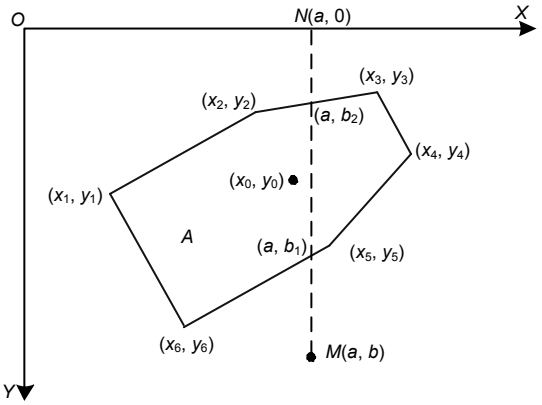

Fig. 7 Point collision detection

(1) Judge whether each line of obstacle A has an intersection with line $x=a$, as shown in Fig. 8. Assume that the end points of line L of obstacle A are (x_i, y_i) and (x_j, y_j). The judgment method is as follows:

If $\max\{x_i, x_j\}\geq a$ and $\min\{x_i, x_j\}\leq a$, then the $x=a$ line passes through the segment L or its rightmost

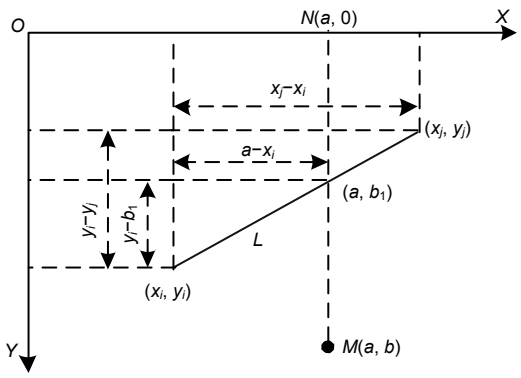

Fig. 8 Proportional intersection calculation

endpoint or its leftmost endpoint. From this we can know that there is an intersection point.

(2) Calculate the intersection point.

According to the proportion formula

$$\frac{a-x_i}{x_j-x_i}=\frac{y_i-b_1}{y_i-y_j},$$

we can obtain $b_1=y_i-(a-x_i)(y_i-y_j)/(x_j-x_i)$. Therefore, the intersection point is (a, b_1).

(3) Repeat the above judgment and intersection operation for each line segment of the obstacle, and finally give all the intersection point sets without duplicate points. Note that since the obstacle models are convex polygons, when a line passes through the obstacle, there are at most two intersection points, such as (a, b_1) and (a, b_2).

(4) Perform the last point conflict detection to show whether node $M(a, b)$ is located inside the obstacle.

If $\min\{b_1, b_2\}<y<\max\{b_1, b_2\}$, then point $M(a, b)$ is located on the line of two intersection points. According to the properties of a convex polygon, point $M(a, b)$ must be located in the interior of the obstacle. That is to say, there is conflict; otherwise, there is no conflict.

3.5.2 Path-segment conflict detection

Path-segment conflict detection is mainly to judge whether the path segment composed of two adjacent path nodes in the planned path intersects with obstacles.

1. Identify the status of each obstacle in the current environment through point conflict detection.

2. Define the coordinates of two adjacent path nodes $M(a, b)$ and $N(c, d)$ which compose the MN path segment.

3. Perform preliminary screening of obstacles. By comparing the relationship between the leftmost, rightmost, uppermost, and lowermost points of the obstacle and the two path nodes of the MN path segment, we can come to a preliminary judgment. For example, assuming that $a<c$ and $b<d$, we can judge as follows:

(1) If $a<x_{\text{right}}$, $c>x_{\text{left}}$, $b<y_{\text{lower}}$, and $d>y_{\text{upper}}$, then the obstacle may intersect with the MN path segment.

(2) If $a\geq x_{\text{right}}$ or $c\leq x_{\text{left}}$, $b\geq y_{\text{lower}}$, and $d\leq y_{\text{upper}}$, then the obstacle can be excluded.

4. Perform path-segment conflict detection between the MN segment and each obstacle screened out. We take obstacle A as an example, as shown in Fig. 9.

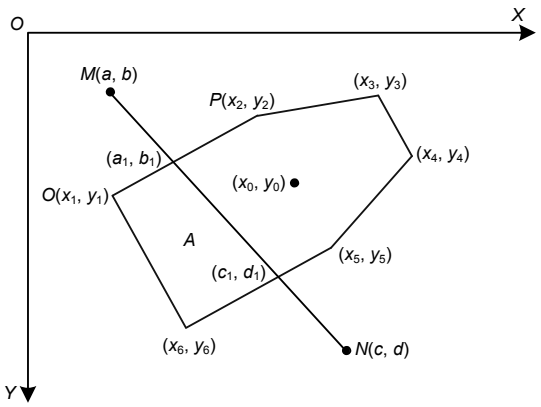

Fig. 9 Path-segment conflict detection

We must judge whether each line segment of obstacle A has an intersection with the MN path segment. Assume an OP line of some obstacles whose end points are $O(x_i, y_i)$ and $P(x_j, y_j)$, $i, j=1, 2, \ldots, 6$. Then the judgment method is as follows:

1. We give the linear equations of the MN path segment and OP line as

$$f(MN):\begin{cases}x=a, & \text{if } MN//Y \text{ axis},\\ y=\dfrac{b-d}{a-c}(x-a)+b, & \text{otherwise.}\end{cases}$$

$$f(OP):\begin{cases}x=x_i, & \text{if } OP//Y \text{ axis},\\ y=\dfrac{y_i-y_j}{x_i-x_j}(x-x_i)+y_i, & \text{otherwise.}\end{cases}$$

2. According to different situations, we judge whether the two line segments intersect with each other:

(1) If MN//Y axis and OP//Y axis, then there is no need to judge the intersection, because the two lines are either parallel or partially or completely coincident.

(2) If MN//Y axis and not OP//Y axis, then:

(a) $[b-(a-x_i)(y_i-y_j)/(x_i-x_j)-y_i][d-(c-x_i)(y_i-y_j)/(x_i-x_j)-y_i]\leq 0$, meaning that M and N are on both sides of the OP line or that one of them is on the OP line.

(b) $(x_i-a)(x_j-a)\leq 0$, meaning that O and P are on both sides of the MN segment or one of them is on the MN segment.

If the above two conditions are met at the same time, then the MN and OP lines intersect with each other and the number of intersections should be increased by 1. In particular, if $(x_i-a)(x_j-a)=0$, the coordinates of the point (O or P) located on the MN segment need to be recorded.

(3) If not MN//Y axis and OP//Y axis, then:

(a) $(a-x_i)(c-x_i)\leq 0$, meaning that M and N are on both sides of the OP line or that one of them is on the OP line.

(b) $[y_i-(b-d)(x_i-a)/(a-c)-b][y_j-(b-d)(x_j-a)/(a-c)-b]\leq 0$, meaning that O and P are on both sides of the MN segment or that one of them is on the MN segment.

If the above two conditions are met at the same time, then the MN line and OP line intersect with each other and the number of intersections should be increased by 1. In particular, if $[y_i-(b-d)(x_i-a)/(a-c)-b][y_j-(b-d)(x_j-a)/(a-c)-b]=0$, the coordinates of the point (O or P) located on the MN segment need to be recorded.

(4) If not MN//Y axis and not OP//Y axis, then:

(a) $[b-(a-x_i)(y_i-y_j)/(x_i-x_j)-y_i][d-(c-x_i)(y_i-y_j)/(x_i-x_j)-y_i]\leq 0$, meaning that M and N are on both sides of the OP line or that one of them is on the OP line.

(b) $[y_i-(b-d)(x_i-a)/(a-c)-b][y_j-(b-d)(x_j-a)/(a-c)-b]\leq 0$, meaning that O and P are on both sides of the MN segment or that one of them is on the MN segment.

If the above two conditions are met at the same time, then the MN and OP lines intersect with each other and the number of intersections should be increased by 1. In particular, if $[y_i-(b-d)(x_i-a)/(a-c)-b][y_j-(b-d)(x_j-a)/(a-c)-b]=0$, the coordinates of the point (O or P) located on the MN segment need to be recorded.

3. Judge each edge of the obstacle repeatedly, merge the recorded point coordinates, and remove the repeated intersections. If the number of intersections is 2, then the MN path segment and the current obstacle intersect with each other, meaning that there is a conflict; otherwise, there is no conflict. Note that it is impossible to have one intersection point here, because the path node which is located inside the obstacle has been excluded during the point conflict detection.

3.6 Path smoothing model

Through planning, we can obtain a series of relatively discrete path points, and the planning path formed by connecting these points may be tortuous, which cannot truly reflect the movement path of the aircraft on the flight deck. Therefore, it is necessary to smooth the planning path (Huang, 2018).

Here, we use a "cutting the sharp angle" method to remove the protrusion point of the path. In Fig. 10, path L contains path points A, B, C, and D, and it is tortuous. Here, we take point B as an example.

1. We insert new nodes $m(a_1, b_1)$ and $n(a_2, b_2)$ at the midpoint of the two path segments connected with node B of the path.

2. We perform conflict detection for new nodes m and n. If there is no conflict, $m(a_1, b_1)$ and $n(a_2, b_2)$ are retained and the initial path node B is deleted. Now, $L=\{A, m, n, C, D\}$.

3. Repeat the above process of inserting nodes until we obtain the expected smooth path.

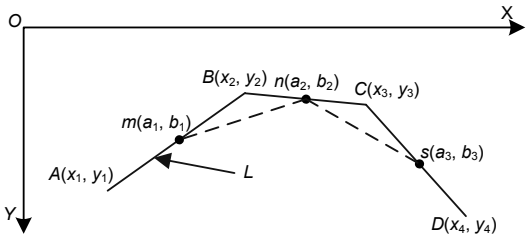

Fig. 10 Path smoothing model

4 Design and verification of the KCMPSO algorithm

In this section, we first analyze the basic particle swarm optimization (PSO) algorithm and its shortcomings, then design an improved KCMPSO algorithm based on PSO, and finally verify the effectiveness of the improved KCMPSO algorithm.

4.1 Analysis of the basic PSO algorithm

PSO is a computing technology based on the swarm intelligence theory, which has the advantages of strong generality, simple principle, and easy implementation (Shao, 2017). PSO has been developed in discrete optimization (Ma et al., 2018), multi-objective optimization (Lu, 2019), constraint optimization (Liang, 2017), and dynamic optimization (Helbig and Engelbrecht, 2014).

PSO is a stochastic optimization algorithm in the multi-dimensional search space. First, the initial positions and velocities of M particles in the particle swarm are initialized randomly in the feasible solution space and velocity space, respectively. For the n-dimensional search space, the expression of the initial position and velocity of the i^{th} particle is

$$\begin{cases} \boldsymbol{X}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,n}), \\ \boldsymbol{V}_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,n}), \end{cases}$$

where for i=1, 2, …, m, j=1, 2, …, n, $x_{i,j}$ is the j^{th} dimension position of the i^{th} particle in the population, and $v_{i,j}$ is the j^{th} dimension velocity of the i^{th} particle in the population.

In each iteration, by evaluating the objective function of each particle (generally, the objective function of the required solution), the historical optimal position of each particle $\boldsymbol{P}_i$=($p_{i,1}$, $p_{i,2}$, …, $p_{i,n}$) and the global optimal position $\boldsymbol{P}_{\text{g}}$=($p_{\text{g},1}$, $p_{\text{g},2}$, …, $p_{\text{g},n}$) found by the group are determined, where $p_{i,j}$ is the j^{th} historical optimal position of the i^{th} particle in the population, and $p_{\text{g},j}$ is the j^{th} global optimal position in the population.

Then, update the speed and position of each particle as follows:

$$\begin{aligned} v_{i,j}(t+1) = {} & \omega v_{i,j}(t) + c_1 r_1 \left[p_{i,j} - x_{i,j}(t) \right] \\ & + c_2 r_2 \left[p_{\text{g},j} - x_{i,j}(t) \right], \end{aligned} \tag{7}$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t), \tag{8}$$

where ω is the inertia weight factor, c_1 and c_2 are the positive acceleration constants, and r_1 and r_2 are the random numbers evenly distributed between 0 and 1. For constrained problems, we generally limit $v_{i,j}(t)$ within [$v_{\min}$, $v_{\max}$] and $x_{i,j}(t)$ within [$x_{\min}$, $x_{\max}$].

However, PSO inevitably has many defects, such as poor local search ability (Wu et al., 2015) and low search accuracy (Jordehi et al., 2015), and it can easily fall into local minima (Osaba et al., 2016). Aimed at the shortcomings of the PSO algorithm, this study designs the KCMPSO algorithm, which divides the population into several subpopulation habitats according to the clustering and executes the iteration without a migration mechanism in parallel.

4.2 Design ideas for the KCMPSO algorithm

Taking into account the shortcomings of the basic PSO algorithm, we make some improvements and obtain the KCMPSO algorithm. The parameters are described in Table 1.

Table 1 Parameter description of the KCMPSO

Parameter	Definition
N	Number of particles in the whole population
K	Number of subpopulations
n	Dimension of the particle position or velocity
ω	Inertia weight factor
c_1, c_2, c_3	Acceleration constants
r_1, r_2, r_3	Random numbers between 0 and 1
$x_{i,j}(t)$	The j^{th} dimension position of the i^{th} particle in the t^{th} iteration
$v_{i,j}(t)$	The j^{th} dimension velocity of the i^{th} particle in the t^{th} iteration
$P_{\text{best}(i,j)}$	The j^{th} dimension historical optimal position of the i^{th} particle
$f_{\text{best}(i)}$	Historical optimal fitness of the i^{th} particle
$P_{\text{worst}(i,j)}$	The j^{th} dimension historical worst position of the i^{th} particle
$f_{\text{worst}(i)}$	Historical worst fitness of the i^{th} particle
$g_{\text{best}(k,j)}$	The j^{th} dimension local optimal position of the k^{th} subpopulation
$f_{\text{gbest}(k)}$	Local optimal fitness of the k^{th} subpopulation
$g^z_{\text{best}(k,j)}$	The z^{th} chaotic local search decision variable
$\text{cg}^z_{\text{best}(k,j)}$	Chaotic variable corresponding to the decision variable of the z^{th} chaotic local search

KCMPSO is executed as follows:

Step 1: population initialization. According to the specific search space, the population with N particles is randomly generated.

(1) Initialize the position and speed of each particle in the population.

(2) Evaluate all particles in each subpopulation. Then, store the current position $\boldsymbol{P}$ (and fitness f) of each particle in its historical optimal position $\boldsymbol{P}_{\text{best}(i)}$ (and historical optimal fitness $f_{\text{best}(i)}$) and also in its

historical worst position $\boldsymbol{P}_{\text{worst}(i)}$ (and historical worst fitness $f_{\text{worst}(i)}$).

Step 2: use the K-means clustering algorithm to cluster particles into K subpopulations (outer loop beginning).

(1) According to a certain distance interval, randomly generate cluster centers.

(2) Calculate the distance between all particles and each cluster center, and classify them into the group represented by the nearest cluster center.

(3) For each population, calculate the mean value of population particles, and use it as the new clustering center of the subpopulation.

(4) Repeat the clustering until the particles of each subpopulation no longer change their population.

(5) Return K clustering centers and their subpopulations.

Step 3: perform independent iterative search for the K populations without a migration mechanism (inner loop beginning).

(1) Calculate the local optimal location $\boldsymbol{g}_{\text{best}(k)}$ and the local optimal fitness $f_{\text{gbest}(k)}$ of subpopulation k ($0\leq k\leq K$).

(2) Update the speed and position of each particle:

$$\begin{aligned} v_{i,j}(t+1)&=\omega v_{i,j}(t)+c_1r_1[p_{\text{best}(i,j)}-x_{i,j}(t)]\\ &\quad+c_2r_2[g_{\text{best}(k,j)}-x_{i,j}(t)]+c_3r_3[x_{i,j}(t)-p_{\text{worst}(i,j)}], \end{aligned} \tag{9}$$

$$\begin{aligned} &x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t),\\ &\quad i=1,2,\cdots,m_k,\ \ j=1,2,\cdots,n,\ \ k=1,2,\cdots,K. \end{aligned} \tag{10}$$

(3) Calculate the fitness of every particle in each subpopulation.

(4) Compare the current fitness of each particle in each subpopulation with its historical optimal fitness $f_{\text{best}(i)}$ and historical worst fitness $f_{\text{worst}(i)}$. Choose the best particle to update $\boldsymbol{P}_{\text{best}(i)}$ and $f_{\text{best}(i)}$; choose the worst one to update $\boldsymbol{P}_{\text{worst}(i)}$ and $f_{\text{worst}(i)}$.

(5) Select the particle H with the best fitness from all particles in each subpopulation. Compare H with the last local optimal particle of the same subpopulation. If H is better, then update the local optimal $g_{\text{best}(k)}$ and $f_{\text{gbest}(k)}$ of the subpopulation.

Step 4: search for the local optimal particle of each subpopulation in step 3 by dimension-chaos search, which uses the logistic chaotic sequence (Tian and Shi, 2018). Chaotic sequences have ergodicity; that is, chaotic sequences can go through all states in a specific region without repetition. The search algorithm with chaotic sequences can generate many neighboring points of the local optimal solution in iteration, to improve the search ability and quickly find the optimal solution. The main idea of chaos search is based on Eq. (11) to generate chaotic sequences:

$$y_d^{z+1}=\mu y_d^z(1-y_d^z), \tag{11}$$

where z is the number of iterations, d is the dimension of the optimal value, and μ is the control parameter of the chaotic state. When μ=4, the logistic equation enters into a chaotic state completely (Luo and Li, 2010).

According to the logistic chaotic sequence, in this study we perform a dimension-chaos search for each dimension of $\boldsymbol{g}_{\text{best}(k)}=(g_{\text{best}(k,1)},\ g_{\text{best}(k,2)},\ \cdots,\ g_{\text{best}(k,n)})$ as follows:

(1) Initialize the chaos number of iterations z=0.

(2) Map the decision variables $g^z_{\text{best}(k,j)}$ into the chaotic variables $\text{cg}^{z+1}_{\text{best}(k,j)}$ between 0 and 1 according to Eq. (12):

$$\text{cg}^z_{\text{best}(k,j)}=\frac{g^z_{\text{best}(k,j)}-D_{\min,j}}{D_{\max,j}-D_{\min,j}},\ \ j=1,2,\cdots,n, \tag{12}$$

where $D_{\max,j}$ and $D_{\min,j}$ are the maximum and maximum values of the j^{th} dimension variables in the search range, respectively.

(3) According to $\text{cg}^{z+1}_{\text{best}(k,j)}$, Eq. (13) is used to calculate the chaotic variable $\text{cg}^{z+1}_{\text{best}(k,j)}$ of the next iteration:

$$\text{cg}^{z+1}_{\text{best}(k,j)}=4\text{cg}^z_{\text{best}(k,j)}\left(1-\text{cg}^z_{\text{best}(k,j)}\right). \tag{13}$$

(4) Transform the chaotic variables into decision variables $g^{z+1}_{\text{best}(k,j)}$ according to Eq. (14):

$$g^{z+1}_{\text{best}(k,j)}=D_{\min,j}+\text{cg}^{z+1}_{\text{best}(k,j)}(D_{\max,j}-D_{\min,j}). \tag{14}$$

(5) According to the decision variables $g^{z+1}_{\text{best}(k,j)}$ (j=1, 2, …, n), the performance of the new solution is

evaluated. If the new solution is better than the original one or the chaos search has reached the preset maximum number of iterations, the new solution will be returned as the search result; otherwise, the next chaos search will be carried out (z=z+1).

In this part, the correlation between each dimension of the local global optimal position is separated, and a dimension-chaos search is performed for each dimension. The advantage of this method is that some dimensions may be improved to the optimal direction in some cases, but the overall target value is poor and negation is carried out. The dimension-chaos search can improve the accuracy to a certain extent.

Step 5: repeat steps 3 and 4 until the preset maximum number of iterations for the inner loop is reached.

Step 6: if the solution in step 5 meets the requirements or reaches the preset number of iterations for the outer loop, the local optimal positions and fitness values of each subpopulation will be outputted. They will be sorted according to the order of advantages and disadvantages. Thus, the particle with serial number 1 will be taken as the global optimal particle. If the solution does not meet the requirements or fails to reach the number of iterations for the outer loop, all subpopulation particles are merged and return to step 2.

4.3 Experiment and analysis

To verify the effectiveness of KCMPSO, we use the standard test function of BR-Branin to test and compare the PSO and KCMPSO algorithms.

The BR-Branin (n=2) is defined as follows (Rosli et al., 2017):

$$\begin{aligned} &f_{\mathrm{BR}}(x)\\ &=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10,\\ &-5\le x_1\le 10,\ 0\le x_2\le 15. \end{aligned} \tag{15}$$

The global minimum value of this function is 0.398, which has three best points (−3.142, 12.275), (3.142, 2.275), and (9.425, 2.425). When testing the KCMPSO and standard PSO algorithms, we cluster the population according to the above three best points, and form three subhabitats.

The common parameter settings are as follows: the population size is set to 60, $v_{\max}$ (i.e., the max moving distance of the particle) is set to 15% of the search space, the number of iterations for the inner loop is set to 50, the number of iterations for the outer loop is set to 5, and the final result is the average value after 50 independent operations. For the standard PSO algorithm, c_1 and c_2 are set to 2.0, and ω is a linearly changing inertia weight factor changing from 1.2 to 0.2. For the KCMPSO, the adaptive inertia weight factor shown in Eq. (16) is used (here, $\omega_{\max}$=1.2 and $\omega_{\min}$=0.2), and acceleration constants are c_1=1.7, c_2=2, and c_3=0.6 (Trelea, 2003).

$$\omega=\begin{cases}\omega_{\min}+\dfrac{(\omega_{\max}-\omega_{\min})(f-f_{\min})}{f_{\mathrm{avg}}-f_{\min}}, & f\le f_{\mathrm{avg}},\\ \omega_{\max}, & f>f_{\mathrm{avg}},\end{cases} \tag{16}$$

where $\omega_{\max}$ and $\omega_{\min}$ represent the maximum and minimum values of ω, respectively, f is the current fitness value of the particle, f_{avg} is the average fitness of all particles, and $f_{\min}$ is the minimum fitness of all particles.

In this study, PSO and KCMPSO algorithms are compared in terms of the average minimum function value, total standard deviation, and convergence process.

The average minimum function value and its difference from the standard global minimum value of 0.398 are shown in Table 2.

The difference between the best points given by the different algorithms and the standard best points of each subpopulation are shown in Table 3.

The comparisons of the convergence process for each subpopulation between these two algorithms are shown in Figs. 11–13.

From Tables 2 and 3, it can be seen that the best point and the optimal value calculated by the KCMPSO algorithm are better than those of the PSO algorithm, indicating that the former is better in its search quality and search results than the latter. From Figs. 11–13, it can be seen that the average value of the KCMPSO algorithm decreases faster than that of the PSO algorithm for each subpopulation. Furthermore, the average value of KCMPSO's objective function is better than that of PSO algorithm's objective function. Thus, the KCMPSO algorithm is shown to be more effective than the PSO algorithm.

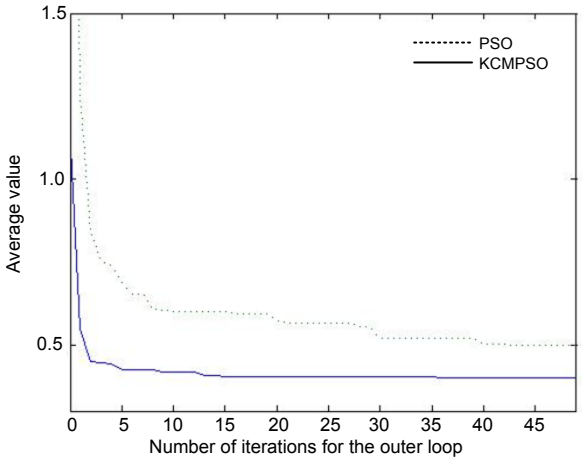

Fig. 11 Comparison of convergence processes of subpopulation 1

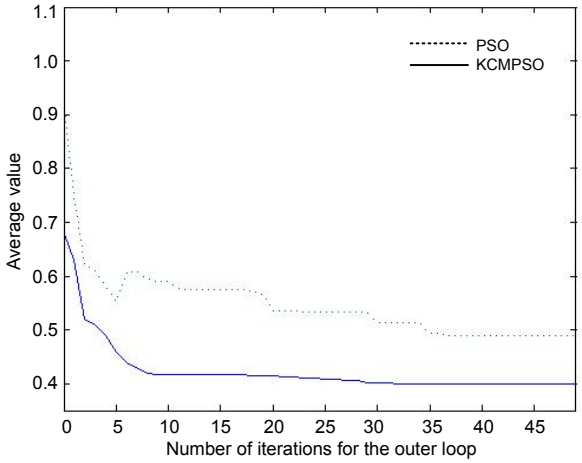

Fig. 12 Comparison of convergence processes of subpopulation 2

5 Simulation verification using KCMPSO

To verify the feasibility of the path planning method designed in this study, we take the Kuznetsov aircraft carrier as an example for simulation verification (Zeng, 2017). The process of applying the KCMPSO algorithm to the path planning problem for the carrier plane is as follows:

1. Establish the aircraft carrier flight deck environment model to define the current transportation environment (Fig. 1).

2. Establish the carrier aircraft model (Fig. 4).

3. Use the entity posture model to adjust the entity of each aircraft, so that it can simulate the actual situation on the aircraft carrier flight deck well.

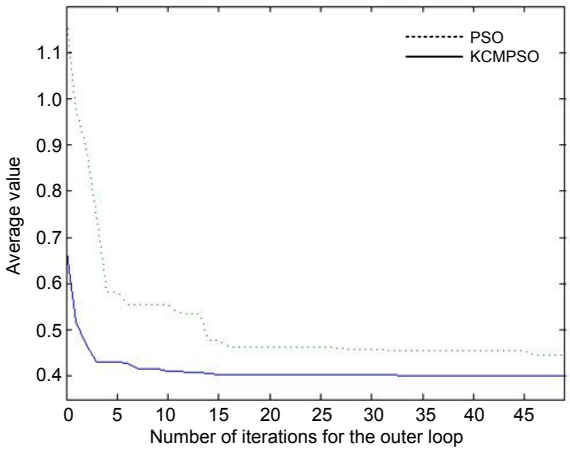

Fig. 13 Comparison of convergence processes of subpopulation 3

Table 2 Comparison of average minimum function values

Algorithm	Calculation result		Standard minimum value		Difference from the standard minimum value	
	PSO	KCMPSO	PSO	KCMPSO	PSO	KCMPSO
Subpopulation 1	0.419 856 016	0.399 888 381	0.398	0.398	0.021 856 016	0.001 888 381
Subpopulation 2	0.409 104 068	0.399 415 025	0.398	0.398	0.011 104 068	0.001 415 025
Subpopulation 3	0.444 800 522	0.400 127 866	0.398	0.398	0.046 800 522	0.002 127 866

Table 3 Comparison of average best points

Algorithm	Calculation result		Standard point		Distance difference	
	PSO	KCMPSO	PSO	KCMPSO	PSO	KCMPSO
Subpopulation 1	(−3.136 401 832, 12.255 320 370)	(−3.134 712 097, 12.262 301 030)	(−3.142, 12.275)	(−3.142, 12.275)	0.020	0.015
Subpopulation 2	(3.103 316 197, 2.356 760 262)	(3.139 768 603, 2.285 789 338)	(3.142, 2.275)	(3.142, 2.275)	0.090	0.011
Subpopulation 3	(9.400 875 392, 2.511 280 741)	(9.426 078 953, 2.480 731 617)	(9.425, 2.425)	(9.425, 2.425)	0.089	0.055

4. In addition to the moving aircraft, expand the remaining obstacles for the carrier-based aircraft and establish the obstacle expansion model.

5. Define the starting point O (138, 198) and the end point B (658, 183) for path planning in Fig. 14.

6. Use the KCMPSO algorithm to do a path optimization search.

(1) Set the algorithm's parameters as follows: the population size is set to 30, and each particle in the population represents a path. The number of iterations of the outer loop is set to 3, the number of iterations of the inner loop is set to 50, and the number of iterations of their operation is set to 50. The acceleration constants are c_1=1.7 and c_2=2. ω is adopted according to Eq. (16), $\omega_{\max}$=1.2, and $\omega_{\min}$=0.2.

(2) Initialize the particle position and velocity (Fig. 15). In the initialization of the particle position $\boldsymbol{A}_i$=($\boldsymbol{a}_{i,1}$, $\boldsymbol{a}_{i,2}$, …, $\boldsymbol{a}_{i,n}$), the intermediate node $\boldsymbol{a}_{i,j}$ is not randomly initialized. We use the entity conflict detection mechanism to initialize the feasible node, so that the path corresponding to the position vector of each particle is feasible as far as possible, so as to improve the efficiency of the algorithm. When the particle velocity vector $\boldsymbol{V}_i$=($v_{i,1}$, $v_{i,2}$, …, $v_{i,n}$) is initialized, it is necessary to note that $v_{i,1}$ corresponding to the starting point and $v_{i,n}$ corresponding to the end point must be 0, because they are fixed. The velocities of other intermediate nodes should not be too large, and can be initialized randomly as

$$v_{i,j} = \text{Random}(v_{i,\min}, v_{i,\max}) = \text{Random}(-10, 10).$$

(3) Use Eq. (1) as the fitness function to search for the shortest path.

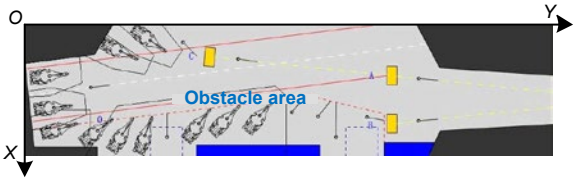

Fig. 14 Obstacle area on the deck

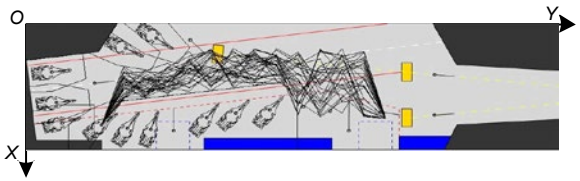

Fig. 15 Initialization of the original routes

(4) Use the K-means clustering algorithm to cluster particles into K subpopulations. Suppose that the position of a particle is

$$\begin{aligned}\boldsymbol{A}_i &= (\boldsymbol{a}_{i,1}, \boldsymbol{a}_{i,2}, \cdots, \boldsymbol{a}_{i,n}) \\ &= \big((x_{i,1}, y_{i,1}), (x_{i,2}, y_{i,2}), \cdots, (x_{i,n}, y_{i,n})\big).\end{aligned}$$

The position of a cluster center is

$$\begin{aligned}\boldsymbol{A}_o &= (\boldsymbol{a}_{o,1}, \boldsymbol{a}_{o,2}, \cdots, \boldsymbol{a}_{o,n}) \\ &= \big((x_{o,1}, y_{o,1}), (x_{o,2}, y_{o,2}), \cdots, (x_{o,n}, y_{o,n})\big).\end{aligned}$$

The distance is defined as

$$\|\boldsymbol{A}_i - \boldsymbol{A}_o\| = \frac{1}{n}\left(\sum_{j=1}^{n}\sqrt{(x_{i,j} - x_{o,j})^2 + (y_{i,j} - y_{o,j})^2}\right).$$

The distance between each particle and each cluster center is calculated, and the particle is classified into the subpopulation of the nearest cluster center.

(5) Perform a multi-group search. A certain number of searches are performed for each subpopulation, where each particle updates the velocity and position according to Eqs. (9) and (10), respectively. After a search for each subpopulation, it is necessary to determine the local global optimal individual of the subpopulation, and then perform a dimension-chaos search on it to improve its quality.

(6) Evaluate the particles. According to the principle of "the shorter the length, the better the path," the K optimal paths calculated by the K subpopulation are sorted, and the optimal path sequence is given.

7. The final path is smoothed to reflect the real moving route of the aircraft on the flight deck (Fig. 16).

Based on the above analysis, it can be seen that the path planning method based on the KCMPSO algorithm can solve the flight deck path planning problem for carrier-based aircraft better.

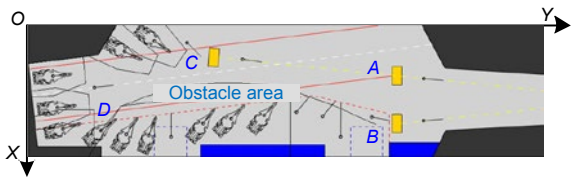

Fig. 16 Best smoothing route based on KCMPSO

6 Conclusions

Through the above discussion, several conclusions are obtained as follows:

1. The problem of transfer path planning for moving and repositioning carrier-based aircraft on the flight deck is studied, and the mathematical model established is in accordance with the actual circumstances.

2. On the premise of meeting the actual needs, the designed mathematical support models for the carrier aircraft entity, entity extension, entity posture, entity conflict detection, and path smoothing can facilitate the subsequent operation.

3. To obtain the optimal route, an algorithm called KCMPSO is designed based on the improvements over the basic PSO algorithm. Through comparative analysis, KCMPSO is shown to be superior to the PSO algorithm in terms of search quality and convergence rate.

4. Finally, the mathematical support models are used to simulate and verify KCMPSO's performance on the Kuznetsov aircraft carrier. The results meet the actual basic requirements and show that the designed method is feasible.

Contributors

Weichao SI and Tao SUN designed the research. Weichao SI and Chao SONG processed the data. Jie ZHANG drafted the manuscript. Weichao SI and Chao SONG revised and finalized the paper.

Compliance with ethics guidelines

Weichao SI, Tao SUN, Chao SONG, and Jie ZHANG declare that they have no conflict of interest.

References

Chen XY, 2018. The U.S. DoD Publish total aviation mishaps in recent years. *Int Aviat*, 6:25-28.

Hao SQ, Cheng SW, Zhang YP, 2018. A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation. *Chin J Aeronaut*, 31(7):1579-1593. https://doi.org/10.1016/j.cja.2018.04.017

He SH, Yan SW, Xu JW, 2019. Path designing for aircrafts' taxiing on flight deck while launching. *J Nav Aeronaut Astron Univ*, 34(1):126-132 (in Chinese). https://doi.org/10.7682/j.issn.1673-1522.2019.01.005

Helbig M, Engelbrecht AP, 2014. Population-based metaheuristics for continuous boundary-constrained dynamic multi-objective optimisation problems. *Swarm Evol Comput*, 14:31-47. https://doi.org/10.1016/j.swevo.2013.08.004

Huang C, 2018. Study on Path Planning and Location of Mobile Robot based on Intelligent Optimization Algorithm. MS Thesis, Dalian Jiaotong University, Dalian, China (in Chinese).

Jordehi AR, Jasni J, Wahab NA, et al., 2015. Enhanced leader PSO (ELPSO): a new algorithm for allocating distributed TCSC's in power systems. *Int J Electron Power Energy Syst*, 64:771-784. https://doi.org/10.1016/j.ijepes.2014.07.058

Liang WJ, 2017. The Research and Implementation of MES based on Hybrid PSO Multi-objective Job Shop Scheduling. MS Thesis, Guangdong University of Technology, Guangzhou, China (in Chinese).

Liu A, Liu K, 2017. Advances in carrier-based aircraft deck operation scheduling. *Syst Eng Theory Pract*, 37(1):49-60 (in Chinese). https://doi.org/10.12011/1000-6788(2017)01-0049-12

Lu W, 2019. Design and Application of Dynamic Multi-objective Particle Swarm Optimization Algorithm. MS Thesis, Beijing University of Technology, Beijing, China (in Chinese).

Luan TT, 2019. Research on Model and Evaluation Method Capacity for Sortie Process of Carrier Aircraft. MS Thesis, Harbin Engineering University, Harbin, China (in Chinese).

Luo J, Li Y, 2010. Artificial bee colony algorithm with chaotic-search strategy. *Contr Dec*, 25(12):1913-1916 (in Chinese). https://doi.org/10.13195/j.cd.2010.12.156.luoj.013

Ma JC, Yao DK, Zhao GH, et al., 2018. Research on airspace planning of tactical training based on discrete particle swarm optimization. *Fire Contr Commun Contr*, 43(12):94-98 (in Chinese). https://doi.org/10.3969/j.issn.1002-0640.2018.12.020

Osaba E, Yang XS, Diaz F, et al., 2016. An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. *Eng Appl Artif Intell*, 48:59-71. https://doi.org/10.1016/j.engappai.2015.10.006

Qi C, Wang D, 2016. Dynamic aircraft carrier flight deck task planning based on HTN. *IFAC-PapersOnline*, 49(12): 1608-1613. https://doi.org/10.1016/j.ifacol.2016.07.810

Rosli NS, Ibrahim R, Ismail I, 2017. Intelligent prediction system for gas metering system using particle swarm optimization in training neural network. *Proc Comput Sci*, 105:165-169. https://doi.org/10.1016/j.procs.2017.01.197

Ryan JC, Banerjee AG, Cummings ML, et al., 2014. Comparing the performance of expert user heuristics and an integer linear program in aircraft carrier deck operations. *IEEE Trans Cybern*, 44(6):761-773. https://doi.org/10.1109/TCYB.2013.2271694

Shao Q, 2017. Particle Swarm Optimization and its Application in Engineering. MS Thesis, Jilin University, Jilin, China (in Chinese).

Si WC, Han W, Shi WW, 2012. Research on deck-disposed scheduling method of carrier planes based on PSO

algorithm. *Acta Aeron Astron Sin*, 33(11):2048-2056 (in Chinese).

Si WC, Qi YD, Han W, 2015. Hangar-exporting optimization schedule of multi-carrier plane based on NGA. *Fire Contr Commun Contr*, 40(11):13-19 (in Chinese). https://doi.org/10.3969/j.issn.1002-0640.2015.11.004

Song T, 2018. Design and Implementation of Aircraft Landing Information Processing and Display System. MS Thesis, Harbin Engineering University, Harbin, China (in Chinese).

Tian DP, Shi ZZ, 2018. MPSO: modified particle swarm optimization and its applications. *Swarm Evol Comput*, 41:49-68. https://doi.org/10.1016/j.swevo.2018.01.011

Trelea IC, 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inform Process Lett*, 85(6):317-325. https://doi.org/10.1016/S0020-0190(02)00447-7

Wu Y, Jin YC, Liu XX, 2015. A directed search strategy for evolutionary dynamic multiobjective optimization. *Soft Comput*, 19(11):3221-3255. https://doi.org/10.1007/s00500-014-1477-4

Yang DS, Wang KF, Chen DW, et al., 2018. Parallel carrier fleets: from digital architectures to smart formations. *J Commun Contr*, 4(2):101-110 (in Chinese). https://doi.org/10.3969/j.issn.2096-0204.2018.02.0101

Zeng SY, 2017. Aircraft Carrier Deck Task Plan Repair and Replanning Research under Dynamic Environment. MS Thesis, Huazhong University of Science and Technology, Wuhan, China (in Chinese).

Zhao Q, 2019. Analysis and Research on Driving Behavior of Carrier-based Aircraft Tractor. MS Thesis, Shenyang University of Technology, Shenyang, China (in Chinese).