drones

Article

Multi-Unmanned Aerial Vehicle Path Planning Based on Improved Nutcracker Optimization Algorithm

Chang Xiao, Huiliao Yang * and Bo Zhang

College of Artificial Intelligence and Automation, Nanjing 211106, China; xc13186412319@163.com (C.X.); bzhang@hhu.edu.cn (B.Z.)
* Correspondence: huiliao.yang@hhu.edu.cn

Abstract: For the multi-UAV path planning problem, environmental modeling and an improved swarm intelligence-based optimization algorithm are discussed in this paper. Firstly, to align with reality, specific constraints of UAVs in motions, attitudes and altitudes, real-world threats such as radars and no-fly zones, and inter-UAV collisions are considered. Thus, multi-UAV path planning is transformed into a multi-objective constrained optimization problem. Accordingly, an improved nutcracker optimization algorithm is proposed to solve this problem. Through initializing with logistic chaotic mapping and the lens imaging inverse learning strategy, a more fit elite initialization population is produced to increase the efficiency of path planning. Furthermore, by adjusting adaptive parameters and integrating an improved sine-cosine search strategy, a balance between global exploration capability and local exploitation capability during path planning is achieved. Experimental results show that the improved nutcracker optimization algorithm surpasses other algorithms with respect to both convergence speed and convergence value, making it an effective method for multi-UAV path planning.

Keywords: multi-UAV; path planning; improved nutcracker optimization algorithm

Academic Editor: Oleg Yakimenko

Received: 12 December 2024
Revised: 23 January 2025
Accepted: 1 February 2025
Published: 4 February 2025

Citation: Xiao, C.; Yang, H.; Zhang, B. Multi-Unmanned Aerial Vehicle Path Planning Based on Improved Nutcracker Optimization Algorithm. *Drones* **2025**, *9*, 116. https://doi.org/10.3390/drones9020116

1. Introduction

Owing to their compact size and light weight, strong adaptability, high concealment, and low risk factor, UAVs have been widely applied as their technology matures. However, due to the environmental complexity, task diversity, and payload limitations of UAVs, a single UAV often struggles to meet many practical mission requirements. Therefore, multiple UAVs are growing in significance and have gradually become a research hotspot.

Path planning is a significant part of the execution of the UAV mission and holds significant research value. Existing UAV path planning algorithms mainly fall into three categories. The first are traditional algorithms, including A* algorithm [1], Dijkstra's algorithm [2], artificial potential field method [3], probabilistic roadmaps [4], and rapid-exploration random trees [5], etc. These algorithms perform well in single-UAV path planning but are difficult to apply to large-scale, multi-dimensional and path planning problems. For multiple UAVs, the second are deep learning algorithms, such as deep q-learning (DQN) [6], deep deterministic policy gradient (DDPG) [7], and proximal policy optimization (PPO) algorithms [8]. Bohao Li et al. focused on the study of UAV ground target tracking in obstacle-rich environments using deep reinforcement learning and proposed an improved DDPG algorithm [9]. Xueyuan Wang et al. transformed the path planning problem into a Markov decision process (MDP) with parameterized states, permissible actions, and detailed reward functions [10]. Then, a dueling double deep Q-network (D3QN) was proposed to learn the decision-making policy of a typical UAV, without any prior knowledge of the environment. However, as the number of variables increases, the computational complexity also grows exponentially; thus, deep learning algorithms require long training times and are difficult to converge. The last are intelligent optimization algorithms, such as the ant colony algorithm (ACO) [11], particle swarm optimization (PSO) [12], genetic algorithm (GA) [13], grey wolf optimizer (GWO) [14], and nutcracker optimization algorithm (NOA) [15], etc. Compared to other algorithms, intelligent optimization algorithms can effectively solve complex combinatorial optimization problems.

Intelligent optimization algorithms are effective in handling complex path planning problems, especially those with nonlinear, multi-constraint, and multi-objective characteristics. They do not require an accurate mathematical model of the problem and can find the best or nearly the best solutions in large-scale, high-dimensional problem spaces. Tai-shan Lou proposed a hybrid strategy-based golden jackal optimizer (HGJO) algorithm [16], in which a pre-decreasing slow nonlinear energy decay strategy is utilized to balance global and local search capabilities. However, this method is only applied under two-dimensional conditions and is not suitable for three-dimensional scenarios of UAVs. Jie Zhang et al. improved the sparrow search algorithm (SSA) [17] by applying golden search optimization (GSO) and an adaptive iterative method to adjust local exploitation and global exploration, enhancing overall optimization performance. Xiaobing Yu combined GWO with differential evolution to solve the path planning problem of UAVs [18]. The position update equation of GWO was improved to enhance the search capability of grey wolves, and a rank-based mutation strategy was implemented to promote exploitation while maintaining exploration capabilities. However, these two methods are mainly applied to single-UAV path planning and have limits in multi-UAV path planning. Liang Xu et al. utilized the ideas of dynamic multi-swarm PSO (DMSPSO) and comprehensive learning PSO (CLPSO) to propose the CL-DMSPSO algorithm [19], which further improved the performance of both algorithms. Kai Meng et al. proposed an evolutionary state estimation-based multi-strategy jellyfish search (ESE-MSJS) algorithm for multi-UAV cooperative path planning to search for high-quality paths [20]. However, CL-DMSPSO has only been compared with PSO algorithms and not with other optimization algorithms, which lacks persuasiveness. The convergence speed and robustness of ESE-MSJS still have room for improvement.

NOA is a highly competitive new intelligent optimization algorithm proposed by Mohamed Abdel-Basset in 2023 [15]. This algorithm simulates the seasonal behavior of nutcrackers, demonstrating superior performance with a strong global search capability and being easy to implement. NOA can converge extremely fast and possess good robustness. It is applicable not only to single-objective optimization problems, but also effectively handles multi-objective optimization ones. Therefore, it has more potential for application to multi-UAV path planning issues compared with other swarm intelligence optimization algorithms. However, it also has issues such as uneven population initialization and an imbalance between exploration and exploitation capabilities during path planning. To address these issues, an improved nutcracker optimizer algorithm (INOA) is proposed. By using logistic chaotic mapping for initialization and a lens imaging inverse learning strategy, a more fit elite initialization population is obtained. Furthermore, through the integration of parameter adjustment and an improved sine–cosine search strategy, the balance between global exploration and local exploitation skills is achieved. With these improvements, INOA effectively improves the performance of NOA and has significant advantages in addressing multi-UAV path planning problems.

The main contributions of this paper are summarized as follows:

- A constrained optimization model for multi-UAV path planning is established, which considers realistic constraints and threats that are specific to UAVs. Taking them into account would make the resulting path points more aligned with real-world scenarios of UAVs.

- An INOA is proposed and then applied to the multi-UAV path planning problem. Specifically, elite individuals are first obtained through chaotic mapping initialization and lens imaging-based reverse learning strategy to increase the efficiency of path planning. Then, parameters in the foraging/storage phase of NOA are adjusted to balance global search and local exploitation capabilities. By integrating with the improved sine–cosine strategy, the convergence speed and precision of searching for optimal path is further enhanced.
- The effectiveness of the proposed INOA is first verified using the CEC2020 test suite. Then, simulations of multi-UAV path planning are conducted in various scenarios with the comparison of INOA and other related state-of-the-art algorithms, strongly demonstrating the merits and applicability of INOA.

The remainder of this paper is organized as follows: Section 2 provides the problem description for multi-UAV path planning. Section 3 outlines the technical details of the proposed INOA. Section 4 gives a brief introduction to the applied path fitting method. Then, Section 5 presents the results and analysis of the comparative experiment between INOA and other state-of-the-art swarm intelligence optimization algorithms. Section 6 concludes with a summary of the main contributions.

2. Problem Description of Multi-UAV Path Planning

In path planning for multiple UAVs, they are subjected to various threats such as mountains, radars and no-fly zones, and motion and attitude constraints, which were modeled and analyzed in this study. As shown in Figure 1, the radar detection range is described as a sphere based on its operating principle, the no-fly zones are modeled as cuboids since the forbidden areas of UAVs are usually regular-shaped buildings or grounds, and mountains are modeled as undulations. Multiple UAVs should avoid these threats to reach destinations safely and efficiently while satisfying motion and attitude restrictions.

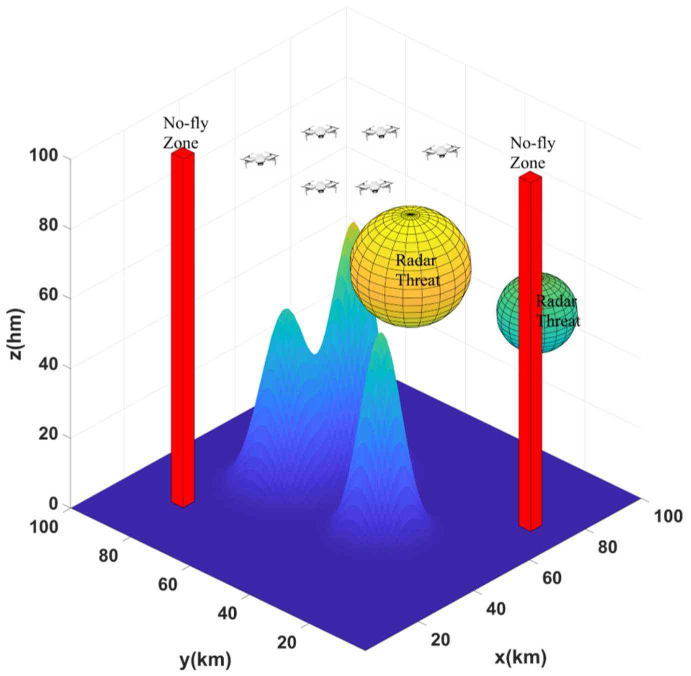

Figure 1. The diagram of the flying environment: the sphere represents the radar detection range, the red cuboid indicates the no-fly zone, and the blue undulations represent mountains.

2.1. Path Length Cost

Suppose that there are U UAVs, each of which has corresponding start and end points. Between the start and end points, there are V trajectory points, denoted as $P_{i,1}, P_{i,2}, P_{i,j}, \cdots, P_{i,V}$, respectively, and each trajectory point has a corresponding coordinate $(x_{i,j}, y_{i,j}, z_{i,j})$. By connecting the start point, trajectory points, and the end point, a corresponding path can be obtained. Thus, the path length cost F_L is defined as follows:

$$F_L = \sum_{i=1}^{U} \sum_{j=1}^{V+1} L_{i,j} \tag{1}$$

$$L_{i,j} = \sqrt{(x_{i,j} - x_{i,j-1})^2 + (y_{i,j} - y_{i,j-1})^2 + (z_{i,j} - z_{i,j-1})^2} \tag{2}$$

where $L_{i,j}$ refers to the distance of the j-th path segment of the i-th UAV.

By constraining the total path length, UAVs can reach the target points with the minimum distance, thereby saving the energy consumption of UAVs.

2.2. Modeling of Flight Environment and Threats

In order to tackle the aforementioned threats properly, the associated penalties are defined as follows to help UAVs avoid the threat areas, and finally synthesized into the total objective function of path planning.

2.2.1. Threat of Mountain Collision

For natural mountains in a flight environment, their mathematical model can be described as follows:

$$Z_k(x, y) = \sum_{k=1}^{n} h_k \exp\left[-\left(\frac{x - x_k}{x_{rk}} \right)^2 - \left(\frac{y - y_k}{y_{rk}} \right)^2 \right] \tag{3}$$

where (x_k, y_k) represents the central coordinate of the i-th mountain, h_k represents the height of the mountain, x_{rk} and y_{rk} are the decay amounts along the x-axis and y-axis of the k-th mountain, and n indicates the sum of mountains.

By referring to possible collisions on a mountain peak, effective obstacle avoidance should be carried out. Then, the penalty F_C of a mountain collision is defined as follows:

$$F_C = \sum_{i=1}^{U} \sum_{j=1}^{V} C_{i,j} \tag{4}$$

$$C_{i,j} = \begin{cases} 100[z_{i,j} - Z], & if\, z_{i,j} < Z \\ 0, & otherwise \end{cases} \tag{5}$$

where $z_{i,j}$ represents the height of the corresponding point on the mountain peak, and Z represents the height of the mountain peak.

2.2.2. Radar Threat

As aforementioned, the detection range of the radar threat that UAVs may encounter is considered as a spherical space, and the corresponding penalty F_R is determined as follows based on whether UAVs fly into the detectable areas:

$$F_R = \sum_{i=1}^{U} \sum_{j=1}^{V} \sum_{k=1}^{R_{num}} R_{i,j} \tag{6}$$

$$R_{i,j} = \begin{cases} 100, & if\|P_{i,j} - O\| \le R \\ 0, & otherwise \end{cases} \tag{7}$$

where O represents the center of the radar's sphere, and R represents the radius range of the radar.

2.2.3. No-Fly Zone Threat

There always exists no-fly zones for UAVs in real applications. Therefore, this kind of threat is also considered and modeled as a cuboid-shaped space, and the penalty F_{NF} is given as follows depending on whether UAVs enter the no-fly zone:

$$F_{NF} = \sum_{i=1}^{U}\sum_{j=1}^{V}\sum_{k=1}^{NF_{num}} NF_{i,j} \tag{8}$$

$$NF_{i,j} = \begin{cases} 100, & if \quad (N_x - a \le x_{i,j} \le N_x + a)\, and\, (N_y - b \le y_{i,j} \le N_y + b) \\ 0, & otherwise \end{cases} \tag{9}$$

where $NF_{i,j}$ is a flag of the no-fly zone threat. (N_x, N_y), $2a$, and $2b$ are the center coordinate, length, and width of the no-fly zone.

2.3. Constraints of Motions and Attitudes of UAVs

Similarly, to address motion and attitude restrictions during path planning, other penalties are defined to ensure UAVs operate within these specified limitations and are also synthesized into the total objective function presented later.

2.3.1. Constraints of Path Segment Distance

When the path segment distance is too small, the UAV cannot complete some state adjustments; accordingly, the penalty F_{SL} based on the constraint of the path segment distance is defined as follows:

$$F_{SL} = \sum_{i=1}^{U}\sum_{j=1}^{V+1} SL_{i,j} \tag{10}$$

$$SL_{i,j} = \begin{cases} 100, & if\, L_{i,j} \le l_{min} \\ 0, & otherwise \end{cases} \tag{11}$$

where the path segment distance $L_{i,j}$ is defined in Equation (1). l_{min} refers to its minimum value and can be determined upon the specifications of the UAV.

2.3.2. Attitudinal Constraints

The UAVs considered in this paper are quadrotors, and their dynamic equations are as follows [21]:

$$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}\begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \tag{12}$$

$$\boldsymbol{I}\begin{bmatrix} \dot{\beta} \\ \dot{\phi} \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} \beta \\ \phi \\ \alpha \end{bmatrix} \times \left(\boldsymbol{I}\begin{bmatrix} \beta \\ \phi \\ \alpha \end{bmatrix}\right) = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} \tag{13}$$

where m is the mass of the UAV, $\ddot{\boldsymbol{p}}$ is the acceleration vector of the UAV's center of mass, g is the gravity acceleration, and $\boldsymbol{R}$ is a rotation matrix that converts vectors from the body frame to the world frame. F_1, F_2, F_3, F_4 are the upward thrust forces generated by the four rotors, respectively. $\boldsymbol{I}$ is the inertia matrix of the UAV, which is a symmetric matrix containing the moments of inertia of the UAV around the three axes. β, ϕ, α and $\dot{\beta}, \dot{\phi}, \dot{\alpha}$ are the angular velocities and accelerations of the UAV, respectively. l is the distance from the rotor to the center of mass of the UAV. M_1, M_2, M_3, M_4 are the counter-torques produced by the four rotors.

As shown in Equations (12) and (13), when UAVs pose large attitude angles, the required thrusts and torques may exceed the maximum values that motors can provide, resulting in insufficient lifts. Therefore, the maximum yaw and pitch angles of UAVs are limited, as well as the curvature of the path, which guarantees the safe operation of UAVs. The schematic diagram of the limitations of the yaw and pitch angles along the path points is shown in Figure 2. As a result, the penalties F_Y and F_A based on these angular limitations are defined as follows, respectively:

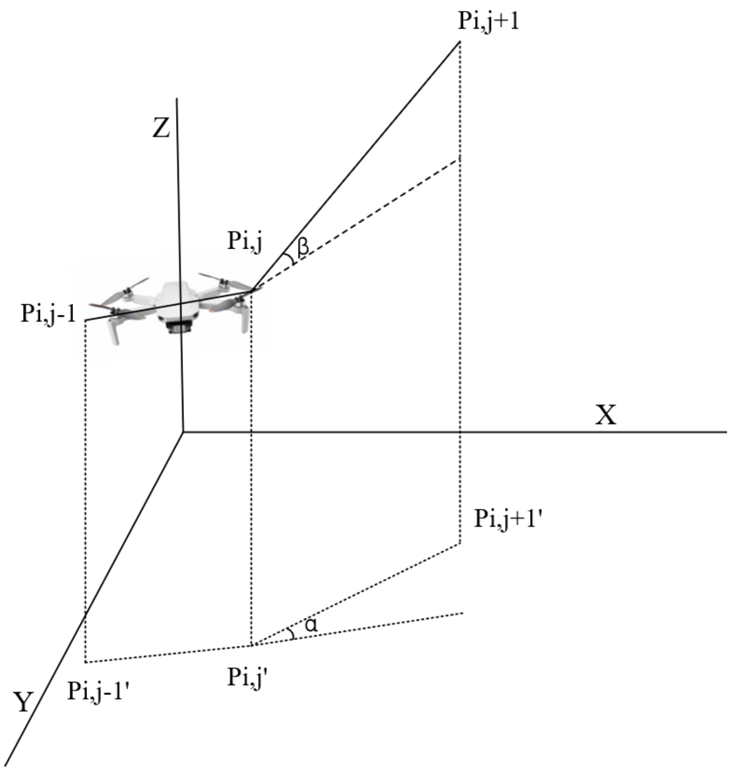

Figure 2. The diagram of the limitations of yaw and pitch angles along the path points: since the yaw angle α and pitch angle β are within the restricted range, the curvature of the path connecting the previous path point $P_{i,j-1}$, the current one $P_{i,j}$, and the next one $P_{i,j+1}$ is limited.

$$F_Y = \sum_{i=1}^{U} \sum_{j=1}^{V+1} Y_{i,j} \tag{14}$$

$$Y_{i,j} = \begin{cases} 0, & if\ \alpha_{i,j} \le \alpha_{max} \\ 100, & otherwise \end{cases} \tag{15}$$

$$\alpha_{i,j} = \arctan\left(\frac{\left\| \overrightarrow{P'_{i,j-1}P'_{i,j}} \times \overrightarrow{P'_{i,j}P'_{i,j+1}} \right\|}{\overrightarrow{P'_{i,j-1}P'_{i,j}} \cdot \overrightarrow{P'_{i,j}P'_{i,j+1}}} \right) \tag{16}$$

$$F_A = \sum_{i=1}^{U} \sum_{j=1}^{V+1} A_{i,j} \tag{17}$$

$$A_{i,j} = \begin{cases} 0, & if\ \beta_{i,j} \le \beta_{max} \\ 100, & otherwise \end{cases} \tag{18}$$

$$\beta_{i,j} = \arctan\left(\frac{z_{i,j} - z_{i,j-1}}{\sqrt{\left(x_{i,j} - x_{i,j-1}\right)^2 + \left(y_{i,j} - y_{i,j-1}\right)^2}}\right) \tag{19}$$

where α_{max}, β_{max} refer to the maximum yaw and pitch angles, respectively. $\alpha_{i,j}$, $\beta_{i,j}$ represent the yaw and pitch angle at the current point, respectively. $\overrightarrow{P'_{i,j-1}P'_{i,j}}$ is the horizontal projection of $\overrightarrow{P_{i,j-1}P_{i,j}}$.

2.3.3. Altitudinal Constraints

In real application, UAVs are usually encouraged to fly at appropriate altitudes to mitigate the effects of winds and ensure safety. Therefore, the penalty F_H depending upon the altitudinal constraint is defined as follows:

$$F_H = \sum_{i=1}^{U}\sum_{j=1}^{V} H_{i,j} \tag{20}$$

$$H_{i,j} = \begin{cases} \left|h_{i,j} - \frac{h_{lb}+h_{ub}}{2}\right|, & if\, h_{lb} \le h_{i,j} \le h_{ub} \\ 100\,, & otherwise \end{cases} \tag{21}$$

where h_{lb} and h_{ub} refer to the lowest and highest flight altitudes, respectively, and $h_{i,j} = z_{i,j}$ represents the height of the UAV.

2.3.4. Collision Avoidance Between UAVs

To avoid collisions, each UAV should search for the position of other UAVs at each trajectory point. Assuming that it flies at a constant speed between trajectory points, with a speed of $V_{i,j}$, the time to pass through each trajectory segment can be calculated by $L_{i,j}$, and then the total flight time can be obtained accordingly.

Suppose that UAVs p and q have a total flight time of $t_{p,j}$ and $t_{q,j}$ at the j-th point $(p, q, j \in N_+, j > 1)$, respectively. When $t_{q,j-1} < t_{p,j} < t_{q,j}$, the current position coordinate of UAV q is given as

$$P_q = P_{q,j-1} + (t_{p,j} - t_{q,j-1})V_{q,j} \tag{22}$$

where $V_{q,j} = (v_{q,jx}, v_{q,jy}, v_{q,jz})$ indicates the velocity of UAV q in three directions.

Similarly, the positions of other UAVs can be determined. With these positions, the safety distance, denoted as u_{sd}, could be set according to the application scenarios. So far, the collision penalty F_{UC} between UAVs is defined as follows:

$$F_{UC} = \sum_{i=1}^{U}\sum_{j=1}^{V} UC_{i,j} \tag{23}$$

where $UC_{i,j}$ is a flag of collision. If the distance $\left|P_i - P_j\right|$ between UAVs i and j exceeds u_{sd}, $UC_{i,j} = 0$; otherwise, $UC_{i,j} = 100$.

2.4. Objective Function

Therefore, by integrating all the aforementioned penalties based on threats, physical limitations of UAVs, as well as the collision avoidance issue into the path length cost shown in Equation (1), the objective function of the multi-UAV path planning problem is written as follows:

$$\begin{aligned} J = {} & w_1F_L + w_2F_{SL} + w_3F_A + w_4F_Y + w_5F_H \\ & + w_6F_C + w_7F_{UC} + w_8F_R + w_9F_{NF} \end{aligned} \tag{24}$$

where J represents the total cost, i.e., the fitness, and w_i refers to the weight of each cost function.

Then, a swarm-intelligence-based optimization method is proposed and applied to find the optimal value of the above objective function, with which the trajectory points along the optimal path of UAVs can be obtained.

3. Improved Nutcracker Optimization Algorithm for Multi-UAV Path Planning

This section first introduces the principle of the standard NOA, which is the basis of solving the multi-UAV path planning problem, and then its limitations are analyzed. Accordingly, INOA is proposed to further improve the planning performance.

3.1. Standard NOA

NOA is motivated by the foraging and food recovery behavior of nutcrackers. It mimics two seasonal behaviors of the nutcracker, namely, seeking and storing seeds in summer and autumn, and foraging based on memory in winter and spring. As a result, this algorithm shows a strong global search capability, and it is suitable to implement in path planning for multiple UAVs due to its applicability to constrained multi-objective optimization problems.

3.1.1. Foraging Phase and Storage Stage

(1) Foraging stage

In this stage, a nutcracker checks the initial position first. If there exists food, it would be actuated towards the storage area; otherwise, it goes to another location for continued searching. The modeling of this behavior is given as follows:

$$\vec{X}_i^{t+1(new)} = \begin{cases} X_{a,j}^t + \gamma\left(X_{U,j}^t - X_{V,j}^t\right) + \mu\left(g^2 U_j - L_j\right), & if\ t \le T/2.0 \\ X_{W,j}^t + \mu\left(X_{U,j}^t - X_{V,j}^t\right) + \mu(g_1 < \delta)\left(g^2 U_j - L_j\right), & otherwise \end{cases} \tag{25}$$

$$\vec{X}_i^{t+1(new)} = \begin{cases} X_{i,j}^t, & if\ h_1 < h_2 \\ Equation\ (25), & otherwise \end{cases} \tag{26}$$

$$\mu = \begin{cases} h_3, if\ g_1 < g_2 \\ h_4, if\ g_2 < g_3 \\ h_5, if\ g_1 < g_3 \end{cases} \tag{27}$$

where $X_{i,j}^t$ represents the j-th position of the i-th nutcracker in the iteration t, $\vec{X}_i^{t+1(new)}$ represents the new position of the nutcracker, $X_{a,j}^t$ represents the average position of all nutcrackers, $X_{U,j}^t$, $X_{V,j}^t$ and $X_{W,j}^t$ represent three different nutcracker individuals randomly selected from the population, and U_j and L_j represent the maximum and minimum values, respectively. γ and h_5 represent the random numbers generated by a Levy flight; g, g_1, g_2, g_3, h_1, h_2, and h_3 represent random numbers in the range of [0, 1]; h_4 refers to a random number that follows a normal distribution; and δ is used to improve the local exploitation capability.

(2) Storage stage

The nutcracker transports the food obtained during the foraging phase to the storage area. This behavior can be mathematically represented as follows:

$$\vec{X}_i^{t+1(new)} = \begin{cases} \vec{X}_i^t + \mu\left(\vec{X}_{best}^t - \vec{X}_i^t\right)|\lambda| + g_1\left(\vec{X}_U^t - \vec{X}_V^t\right) & if\ h_1 < h_2 \\ \vec{X}_{best}^t + \mu\left(\vec{X}_U^t - \vec{X}_V^t\right) & if\ h_1 < h_3 \\ \vec{X}_{best}^t l & otherwise \end{cases} \tag{28}$$

where λ represents the number generated based on a Levy flight, $\vec{X}_{best}^t$ represents the current optimal solution, and l represents a factor that linearly decreases from 1 to 0.

Then, the following equation is used to adjust the conversion between the foraging phase and storage stage:

$$\vec{X}_i^{t+1} = \begin{cases} Equation\ (26), & if\ u < P_{a_1} \\ Equation\ (28), & otherwise \end{cases} \tag{29}$$

where u represents random numbers in the range of [0, 1], and P_{a_1} decreases linearly from 1 to 0.

3.1.2. Cache-Search and Recovery Strategy

(1) Cache-search stage

As winter approaches, the nutcracker would embark on its second round of exploration, transitioning from its storage mode to a search mode by using two reference points (FPs) as markers for a single storage area. These two reference points $FP_{i,1}^t$ and $FP_{i,2}^t$ are defined as follows:

$$\overrightarrow{FP}_{i,1}^t = \begin{cases} \vec{X}_i^t + \varphi cos(\alpha)\left(\left(\vec{X}_U^t - \vec{X}_V^t\right)\right) + \varphi FP, & if\ \alpha = \pi/2 \\ \vec{X}_i^t + \varphi cos(\alpha)\left(\left(\vec{X}_U^t - \vec{X}_V^t\right)\right), & otherwise \end{cases} \tag{30}$$

$$\overrightarrow{FP}_{i,2}^t = \begin{cases} \vec{X}_i^t + \left\{\varphi \cos(\alpha)\left(\left(\vec{U} - \vec{L}\right)h_3 + \vec{L}\right) + \varphi FP\right\}\vec{U}_s & if\ \alpha = \pi/2 \\ \vec{X}_i^t + \varphi cos(\alpha)\left(\left(\vec{U} - \vec{L}\right)h_3 + \vec{L}\right)\vec{U}_s, & otherwise \end{cases} \tag{31}$$

$$\vec{U}_s = \begin{cases} 1, \vec{g_2} < P_{rp} \\ 0, otherwise \end{cases} \tag{32}$$

$$\varphi = \begin{cases} \left(1 - \frac{t}{T}\right)^{2 \cdot \frac{t}{T}}, & if\ g_1 > g_2 \\ \left(\frac{t}{T}\right)^{\frac{2}{t}}, & otherwise \end{cases} \tag{33}$$

where t and T represent the current and maximum number of generations, respectively. α represents a random radian in the range of $[0, \pi]$. P_{rp} represents the percentage likelihood of investigating various areas within the exploration domain.

(2) Recovery stage

When nutcrackers search for their caches, they may encounter two possibilities. The first possibility is that the nutcracker can use the first FP to remember its cache location. The following formula characterizes this behavior:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^t, & if\ h_3 < h_4 \\ X_{i,j}^t + g_1\left(X_{best,j}^t - X_{i,j}^t\right) + g_2\left(\vec{F}P_{i,1}^t - X_{W,j}^t\right), & otherwise \end{cases} \tag{34}$$

The second possibility is that the nutcracker, using the first FP, does not remember the location of the hidden food, and then it would use the second FP to search for food. This behavior can be characterized as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t}, & if\ h_5 < h_6 \\ X_{i,j}^{t} + g_1\left(X_{best,j}^{t} - X_{i,j}^{t}\right) + g_2\left(\vec{F}P_{i,2}^{t} - X_{W,j}^{t}\right), & otherwise \end{cases} \tag{35}$$

where h_6 is also a random number in the interval of [0, 1].

Thus, the recovery behavior can be summarized by the following formula:

$$\vec{X}_i^{t+1} = \begin{cases} Equation\ (34), & if h_7 < h_8 \\ Equation\ (35), & otherwise \end{cases} \tag{36}$$

where h_7, h_8 are random numbers in the interval of [0, 1].

Update the position based on whether the nutcracker finds food, which is determined by the following formula:

$$\vec{X}_i^{t+1} = \begin{cases} \vec{X}_i^{t}, & if\ f(\vec{X}_i^{t}) < if(\overrightarrow{FP}_{i,1}^{t}) \\ \overrightarrow{FP}_{i,1}^{t}, & otherwise \end{cases} \tag{37}$$

$$\vec{X}_i^{t+1} = \begin{cases} \vec{X}_i^{t}, & if\ f(\vec{X}_i^{t}) < if(\overrightarrow{FP}_{i,2}^{t}) \\ \overrightarrow{FP}_{i,2}^{t}, & otherwise \end{cases} \tag{38}$$

To balance foraging behaviors through FP, the following formula is utilized:

$$\vec{X}_i^{t+1} = \begin{cases} Equation\ (37), & if\ f(FP_{i,1}') < f(FP_{i,2}') \\ Equation\ (38), & otherwise \end{cases} \tag{39}$$

Similarly, the following equation is defined to adjust the conversion between the caching phase and recovery phase:

$$\vec{X}_i^{t+1} = \begin{cases} Equation\ (36), & if\ v > P_{a_2} \\ Equation\ (39), & otherwise \end{cases} \tag{40}$$

where v represents a random number in the range of [0, 1], and P_{a_2} represents the probability of exchange between the cache search phase and the recovery phase.

3.2. Improvements on NOA

Although NOA is a competitive and efficient intelligent optimization algorithm for path planning, it also has regular issues, such as uneven population initialization and an imbalance between exploration and exploitation capabilities. Consequently, the following improvements are implemented to enhance its search performance, thereby obtaining more satisfactory paths for UAVs.

3.2.1. Improvement of Initialization Based on Logistic Chaos Map

The basic population initialization is randomly distributed throughout the entire space, which has high randomness and uneven distribution, leading to problems such as a lack of population diversity and low search efficiency. Utilizing the chaotic mapping mechanism to increase the variety of the population can boost the efficiency of the algorithm. Its nonlinear characteristics and periodic nature enable it to generate more complex and random sequences, which helps to enhance the variety of the population and stops the population from converging on local optimal solutions. Commonly used chaotic mapping methods include the logistic map and the tent map. Compared to the other one, the logistic map offers a more flexible parameter adjustment, higher sensitivity to chaos, and advantages in simulating complex systems. Therefore, the logistic map was used for population initialization in this study, which is a nonlinear difference equation used to describe the change in a variable across time. The formula is as follows:

$$x_{n+1} = \mu x_n(1 - x_n) \tag{41}$$

where x_n represents the state variable value at time step n, which is a number between 0 and 1, x_{n+1} is the state variable value at the next time step $n+1$. μ is a positive control parameter, which is typically in the range of [0, 4]. Here, we set it as 4, placing the map in a fully mapped state with ergodicity.

As can be seen in Figures 3 and 4, the sequence produced by the logistic map initialization is more uniform compared to a random initialization, which can find the optimal solution more quickly and avoid being stuck in local minima.

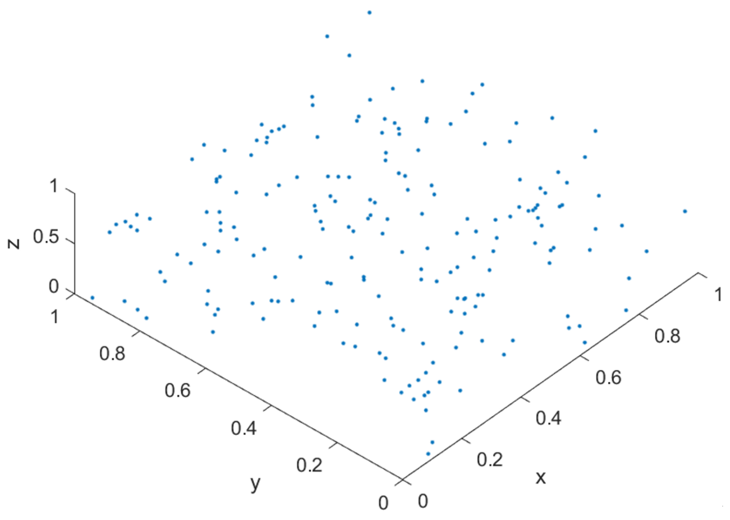

Figure 3. Random initialization adopted in NOA: the initialization sequence is uneven.

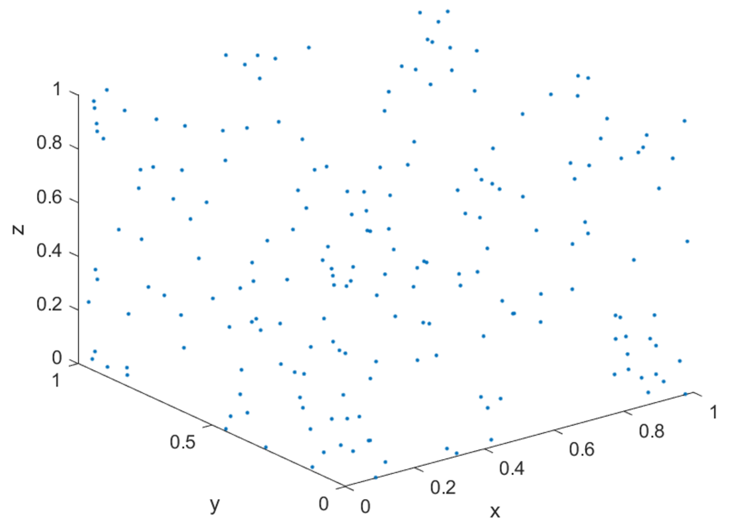

Figure 4. The logistic map initialization employed in INOA: the initialization sequence is more uniform.

3.2.2. Lens Imaging Inverse Learning Strategy

A lens imaging inverse learning strategy is also introduced in this study, which further enhances the diversity of the initial solutions. Compared to the ordinary backward learning strategy, the lens imaging inverse learning introduces a scaling factor k, which is more conducive to obtaining the global optimal solution. The schematic diagram is shown in Figure 5. By generating the inverse solution of the current initial solution through inverse learning, and comparing it with the fitness of the original solution, the initial solutions are updated, thereby further increasing the probability of the algorithm escaping from local optima.

The lens imaging inverse learning strategy is as follows:

$$\vec{X}_{i,j}^{\mathrm{r}} = \left(\vec{U}_j + \vec{L}_j\right)/2 + \left(\vec{U}_j + \vec{L}_j\right)/2k - \vec{X}_{i,j}^{t}/k \tag{42}$$

$$\vec{X}_{i,j}^{t} = \begin{cases} \vec{X}_{i,j}^{\mathrm{r}}, if\ f(\vec{X}_{i,j}^{\mathrm{r}}) < f(\vec{X}_{i,j}^{\mathrm{t}}) \\ \vec{X}_{i,j}^{t}, if\ f(\vec{X}_{i,j}^{\mathrm{r}}) \geq f(\vec{X}_{i,j}^{t}) \end{cases} \tag{43}$$

where $\vec{X}_{i,j}^{\mathrm{r}}$ is the generated inverse solution, $\vec{X}_{i,j}^{t}$ is the current solution, and k is a random value between 1 and 10. When the fitness of the inverse solution is lower than that of the current solution, the inverse solution swaps out the current solution; otherwise, it remains unchanged.

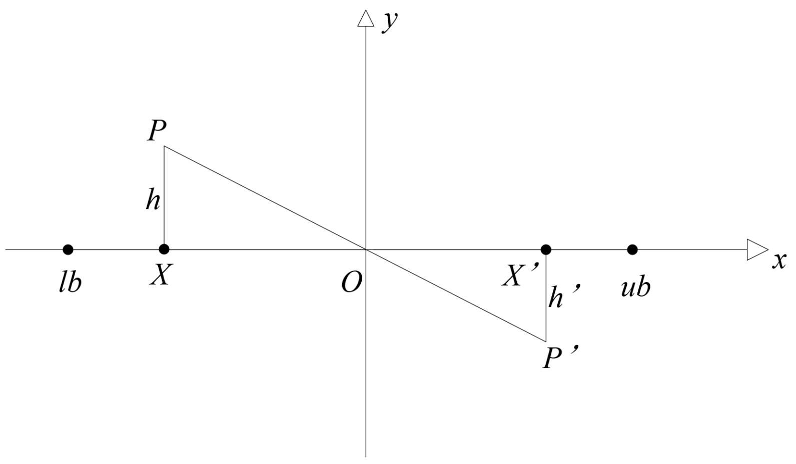

Figure 5. The diagram of the lens imaging inverse learning strategy: if P is the current point within the bounds of $[lb, ub]$, then P' is the inverse solution of P obtained through by the lens imaging inverse learning strategy.

3.2.3. Parameter Adjustment

The exploration phase reflects the algorithm's capability to explore the global search space, determining whether the algorithm can find the optimal solution, while the exploitation phase demonstrates the algorithm's capability to mine in the local search space, affecting the efficiency of the algorithm in obtaining the optimal solution. However, a limitation of NOA is that its transition between exploration and exploitation is not balanced. In the foraging/storage phase of NOA, the parameter P_{a1} is responsible for regulating the conversion between exploration and exploitation, and it decreases linearly. But the actual foraging/storage behavior of the nutcracker may exhibit nonlinear characteristics. Moreover, in the search for the caching/recovery food phase, the parameter P_{a2} that controls exploration and exploitation is set to a fixed value of 0.2, which cannot fully describe the complex behavior of the nutcracker in the process of searching for the caching areas and recovery food. Obviously, these parameters (P_{a1} and P_{a2}) do not accurately reflect the changes in the actual process. Therefore, the following improvements are made, as shown in Figure 6. Specifically, the original parameters are set as follows:

$$P_{a1} = (T - t)/T \tag{44}$$

$$P_{a2} = 0.2 \tag{45}$$

Then, to better balance the exploration and exploitation, the parameters are adjusted as follows:

$$P_{a1new} = (cos(\pi t/T) + 1)/2 \tag{46}$$

$$P_{a2new} = (t/T - 1)^2/10 + 0.2 \tag{47}$$

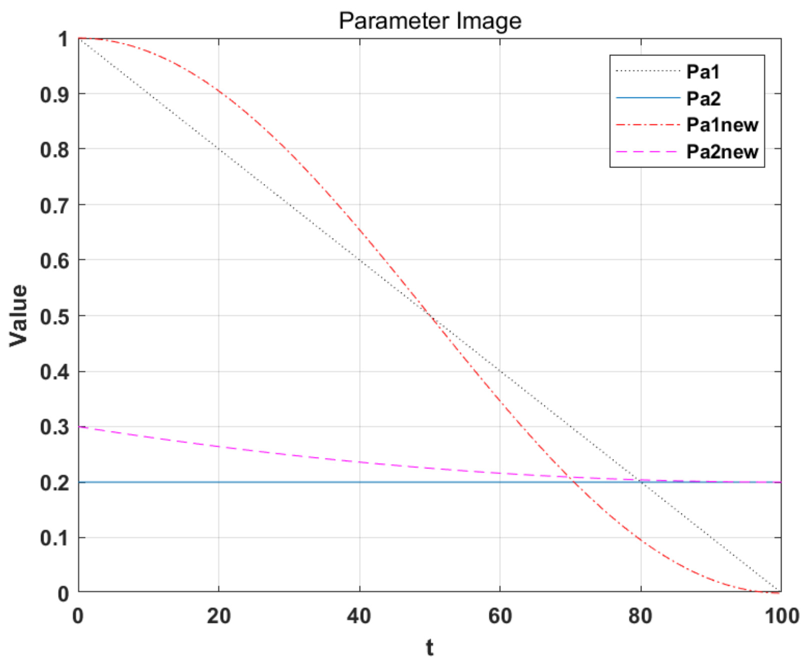

Figure 6. Parameter adjustment: INOA improves parameters P_{a1new} and P_{a2new} in the foraging/storage and the caching/recovery phases as trigonometric and quadratic functions, respectively, to better balance the exploration and exploitation.

3.2.4. Improved Sine–Cosine Strategy

The speed at which the nutcracker finds the food source affects the convergence velocity of the algorithm. By introducing the sine–cosine strategy and utilizing the oscillating characteristics of sine and cosine to search for food sources, the global exploration speed of the algorithm can be effectively improved. The sine–cosine algorithm formula is as follows:

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^t + z_1 \sin(z_2)|z_3 X_{best,j}{}^t - X_{i,j}^t|, z_4 > 0.5 \\ \\ X_{i,j}^t + z_1 \cos(z_2)|z_3 X_{best,j}{}^t - X_{i,j}^t|, z_4 \le 0.5 \end{cases} \tag{48}$$

where $z_1 = 2(1 - t/T)$, $z_2 \in [0, 2\pi]$, $z_3 \in [0, 2]$, and $z_4 \in [0, 1]$.

However, this algorithm has a weak local search capability; therefore, the following improvements are made on it:

$$X_{i,j}^{t+1} = \begin{cases} z_1 \sin(z_2) + X_{best,j}{}^{t}, z_4 > 0.5 \\ \\ z_1 \cos(z_2) + X_{best,j}{}^{t}, z_4 \leq 0.5 \end{cases} \tag{49}$$

To guarantee both the global exploration in the early stage and the local exploitation in the later stage, the sine–cosine strategy is finally modified as follows:

$$X_{i,j}^{t+1} = \begin{cases} Equation\ (48), & if\ t/T < 0.3 \\ Equation\ (49), & otherwise \end{cases} \tag{50}$$

By performing the aforementioned improvements, the flowchart of INOA is displayed in Figure 7, and the main steps are as follows:

Step 1: Set the size of the nutcracker population as N and initialize it using the logistic chaotic mapping.

Step 2: Apply the lens imaging inverse learning strategy to the initialized nutcracker population and select better elite initialization solutions through fitness comparison.

Step 3: Create p and q as random values between 0 and 1. If $p < q$, update the solution using the foraging and storing strategy and optimize the parameter P_{a1}.

Step 4: If $p \geq q$, update the solution using the caching search and recovery strategy and optimize the parameter P_{a2}.

Step 5: Integrate the improved sine–cosine strategy to improve the algorithm's abilities of balancing global exploration and local exploitation.

Step 6: Find the best fitness solution through iterations.

At this point, by applying the proposed INOA, the trajectory points in an optimal path for UAVs can be obtained.

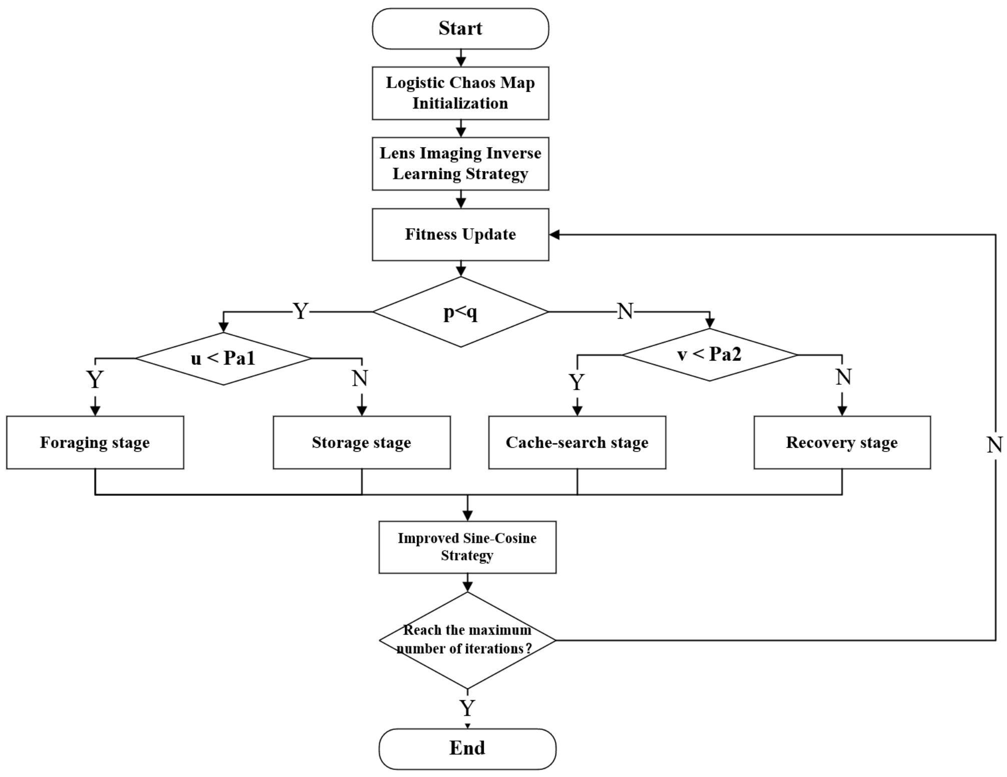
Start
Logistic Chaos Map Initialization
Lens Imaging Inverse Learning Strategy
Fitness Update
p<q
Y
N
u < Pa1
v < Pa2
Y
N
Y
N
Foraging stage
Storage stage
Cache-search stage
Recovery stage
N
Improved Sine-Cosine Strategy
Reach the maximum number of iterations?
Y
End

Figure 7. The flowchat of INOA.

4. Path Fitting

To improve the smoothness of the optimal path and ensure the stability of a UAV flight, it is necessary to fit the path. Cubic spline interpolation was used in this study to construct a smooth curve on a given set of data points such that the curve was a cubic polynomial on each segment. For a set of ordered data points $(x_0, y_0, z_0), (x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$, the cubic spline curve is defined as the cubic polynomial $S_{ijk}(x, y)$ between each pair of adjacent points (x_i, y_i, z_i) and $(x_{i+1}, y_{i+1}, z_{i+1})$. The entire curve is composed of these polynomials spliced together. Each polynomial can be written as follows:

$$\begin{aligned} S_{ijk}(x,y) &= a_{ijk} + b_{ijk}(x - x_i) + c_{ijk}(y - y_j) \\ &+ d_{ijk}(x - x_i)^2 + e_{ijk}(x - x_i)(y - y_j) + f_{ijk}(y - y_j)^2 \\ &+ g_{ijk}(x - x_i)^3 + h_{ijk}(x - x_i)^2(y - y_j) \\ &+ i_{ijk}(x - x_i)(y - y_j)^2 + j_{ijk}(y - y_j)^3 \end{aligned} \tag{51}$$

where $a_{ijk}, b_{ijk}, \ldots, j_{ijk}$ represent the coefficients of the polynomial.

Forming a system of linear equations through interpolation conditions and the continuity of the first and the second derivatives, we can obtain all the coefficients of the polynomials by solving this system. Then, the interpolation function is constructed in the entire three-dimensional space, thereby ensuring that the connection between trajectory points is a smooth curve.

5. Simulation Results and Analysis

In this section, we report simulations of the proposed INOA, including those based on the test set, applications in multi-UAV path planning, and comparative studies, conducted on the MATLAB R2020b environment to comprehensively demonstrate the merits of INOA.

5.1. Simulation Based on Test Set

The effectiveness of the proposed INOA was confirmed through the CEC2020 test suite. The CEC2020 test suite is a set of standard test functions used to evaluate and compare the performance of optimization algorithms, which was proposed at the 2020 congress on evolutionary computation (CEC), and it consists of a total of 10 functions.

To show the superiority of INOA, it was compared to NOA, Harris hawks optimization (HHO), and sea horse optimization (SHO). The population size was set to 50, the maximum number of iterations to 500, and the dimensionality to 20. The algorithms were run 30 times, and the average values were taken for the simulation. The simulation results are shown in Figure 8. It is seen that INOA achieved the best results on all 10 functions during the entire search process, except for functions F2 and F6, where the global search performance does not outperform but still keeps pace with NOA in the early stage.

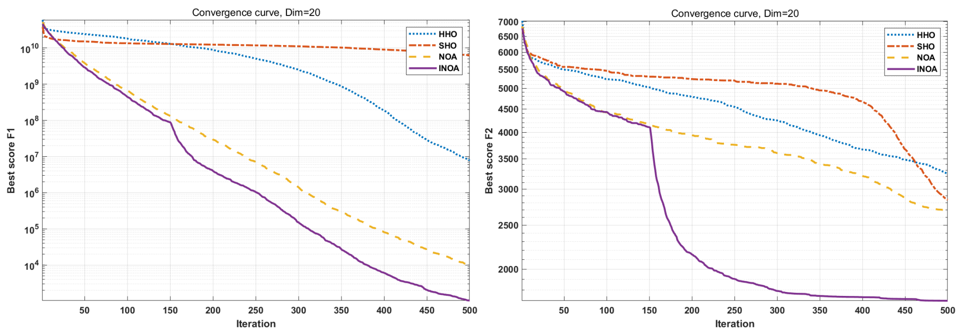

Figure 8. *Cont.*

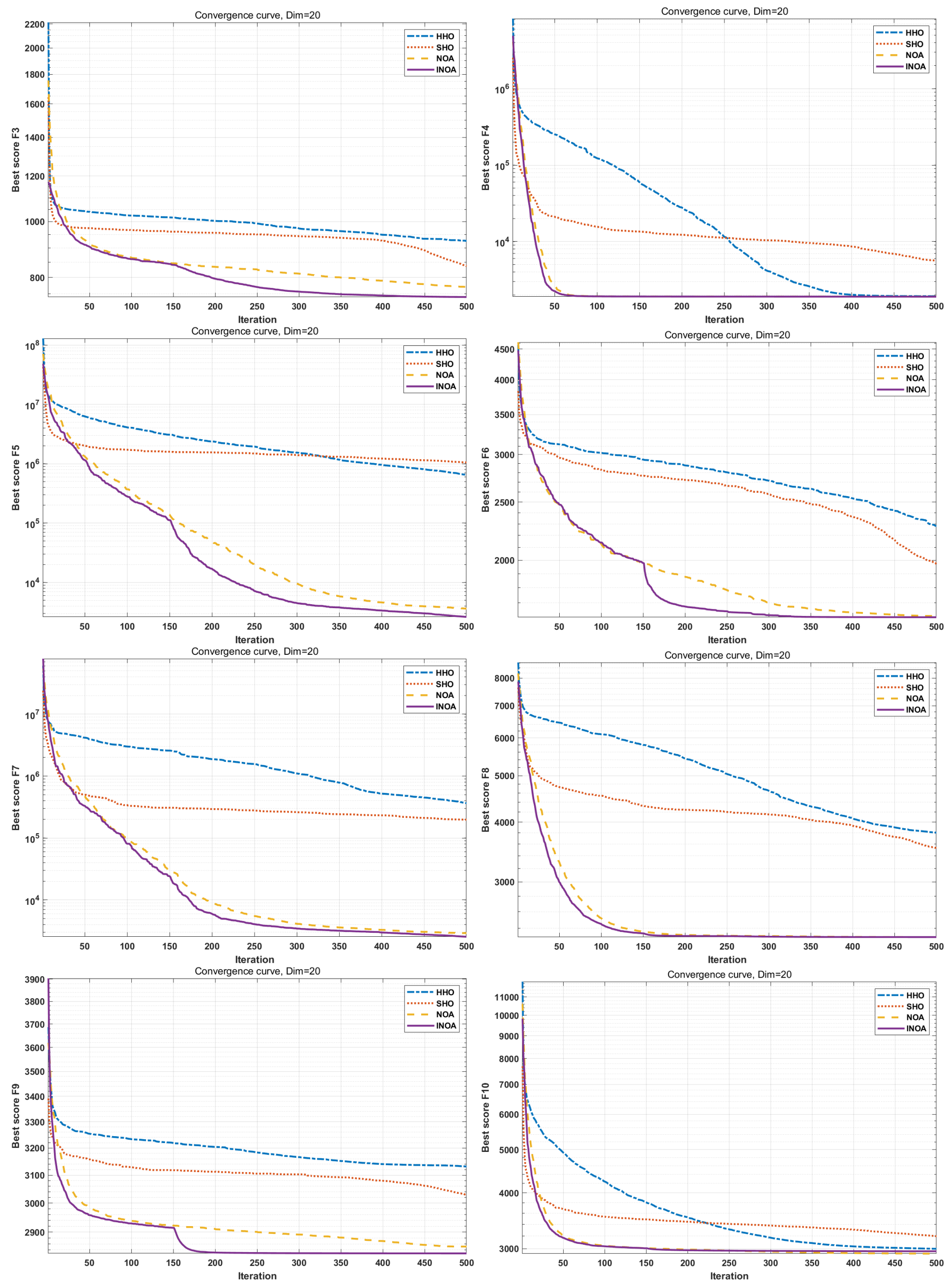

Figure 8. Simulation based on CEC2020: INOA obtains the best results on all 10 functions.

5.2. *Simulation for Multi-UAV Path Planning*

To further verify the effectiveness and applicability of INOA in multi-UAV path planning, simulation scenarios are designed for the following four different flight situations.

In the first scenario, we set the number of UAVs as 6, with 20 waypoints for each. Suppose that there are 2 no-fly zones, 3 mountains, and 2 radar threats in the flying environment. The start points of the UAVs were set as (0, 0, 0), (0, 5, 0), (5, 0, 0), (5, 5, 0), (0, 10, 0), and (10, 0, 0), respectively. The end points were (90, 90, 100), (90, 100, 100), (100, 90, 100), (100, 100, 90), (85, 85, 100), and (90, 85, 100), respectively. In the second scenario, based on the first one, the number of mountains increased to 6. In the third scenario, building upon the first one, the number of radar threats increased to 4.

To enhance the persuasiveness of the algorithm, the fourth scenario considered a more complex situation, in which the number of UAVs was set as 8, with 20 waypoints for each. The start points for each UAV were set as (0, 0, 0), (0, 0, 5), (0, 5, 0), (0, 5, 5) , (5, 0, 0), (5, 0, 5), (5, 5, 0), and (5, 5, 5), while the end points were (95, 95, 100), (95, 95, 95), (95, 100, 95), (95, 100, 100), (100, 95, 95), (100, 95, 100), (100, 100, 95), and (100, 100, 100). Additionally, the number of mountains and radar threats increased to 6 and 4, respectively.

The parameter setting for the problem description is presented in Table 1. In the simulation, we set the population size as 60 and allow for a maximum of 100 iterations.

Table 1. Parameter setting for the problem description.

Parameter	Symbol	Value	Unit
Minimum path segment distance	l_{min}	500	m
Safety distance	u_{sd}	100	m
Maximum yaw angle	α_{max}	60	°
Maximum pitch angle	β_{max}	45	°

Figures 9–16 show the path planning results in all scenarios. It can be seen that INOA always performs well for multi-UAV path planning tasks. Whether there exists an increase in the mountainous terrain or radar threats does not affect the performance of INOA in complex scenarios such as the fourth scenario. As the safe operating space for multiple UAVs is further compressed, the difficulty of finding feasible paths for multiple UAVs greatly increases. Nevertheless, it is evident that INOA can effectively overcome these challenges and successfully find the optimal paths.

To address the advantages of the proposed INOA, a comparative analysis was performed between PSO, GWO, NOA, and INOA based on the first scenario. The parameters of PSO can be found in [12], and those of other algorithms are properly selected and shown in Table 2. To avoid randomness, all algorithms were run 30 times.

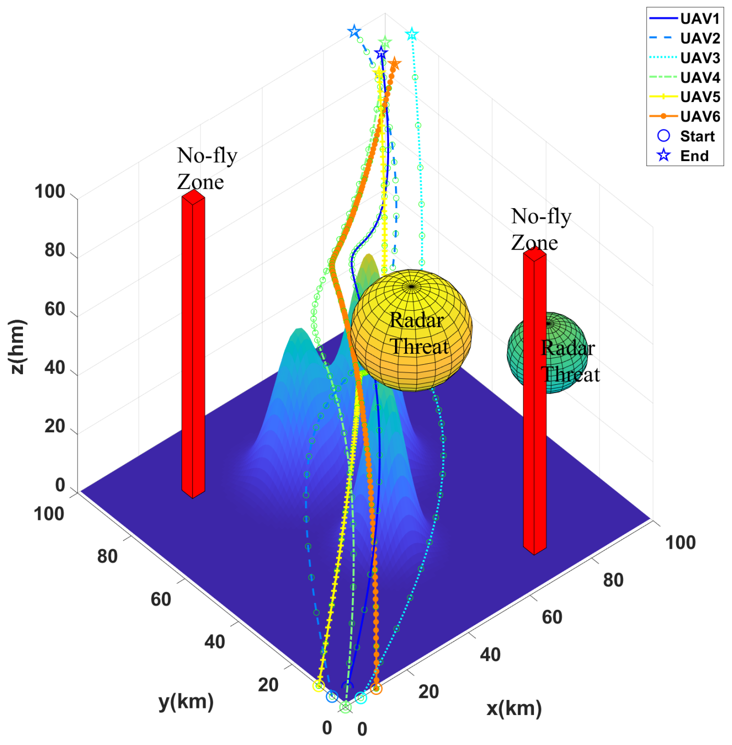

Figure 9. The result of multi-UAV path planning in the first scenario with 6 UAVs, 3 mountains, 2 radars, and 2 no-fly zones.

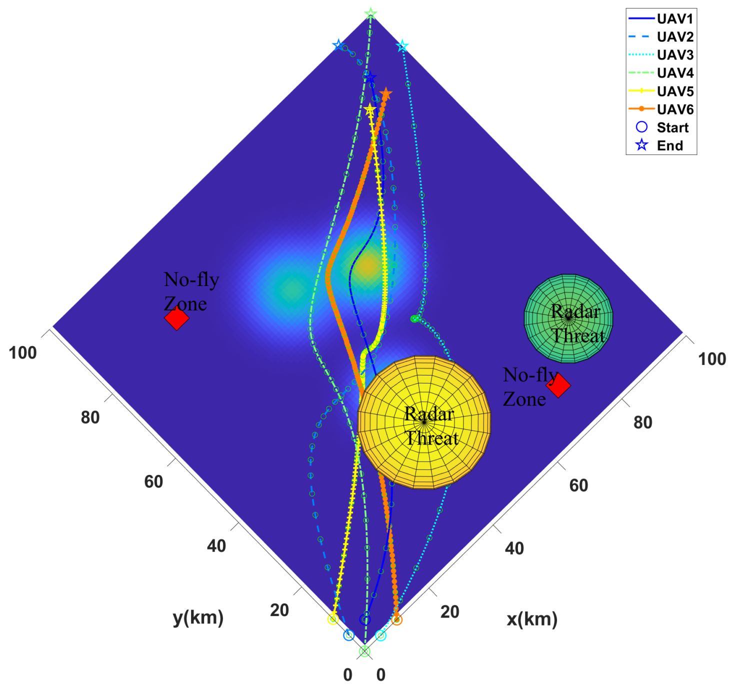

Figure 10. Top–down view of multi-UAV path planning in the first scenario with 6 UAVs, 3 mountains, 2 radars, and 2 no-fly zones.

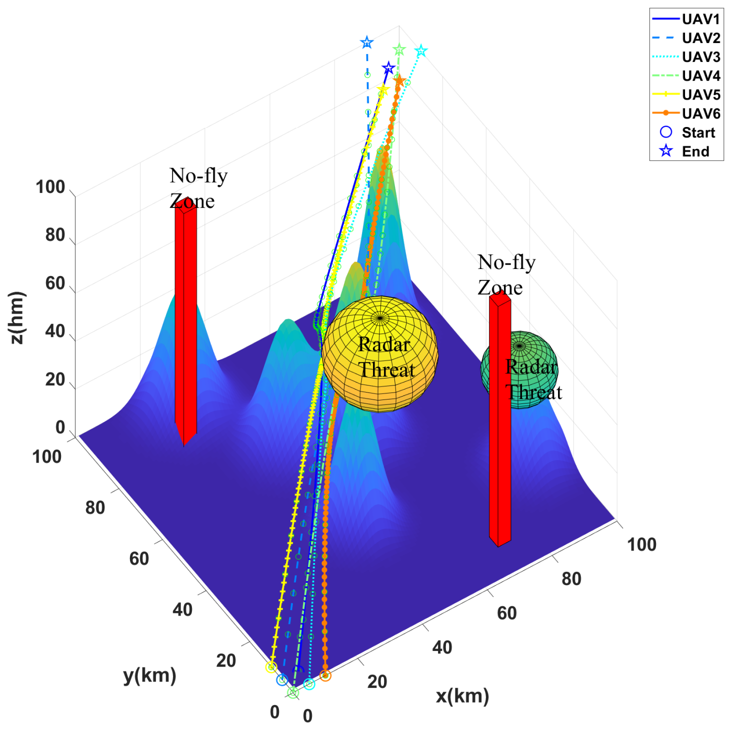

Figure 11. The result of multi-UAV path planning in the second scenario with 6 UAVs, 6 mountains, 2 radars, and 2 no-fly zones: compared with the first scenario, the number of mountains increases to 6.

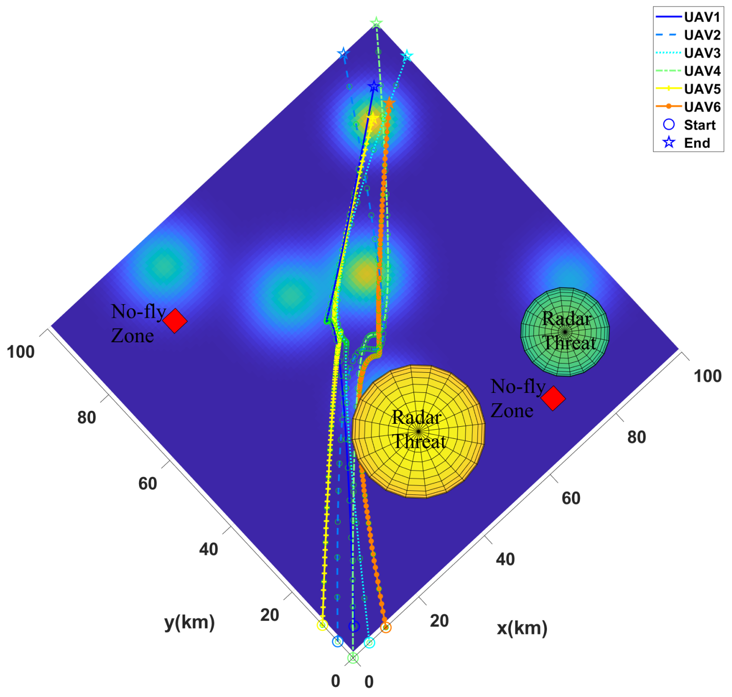

Figure 12. Top–down view of multi-UAV path planning in the second scenario with 6 UAVs, 6 mountains, 2 radars, and 2 no-fly zones: compared with the first scenario, the number of mountains increases to 6.

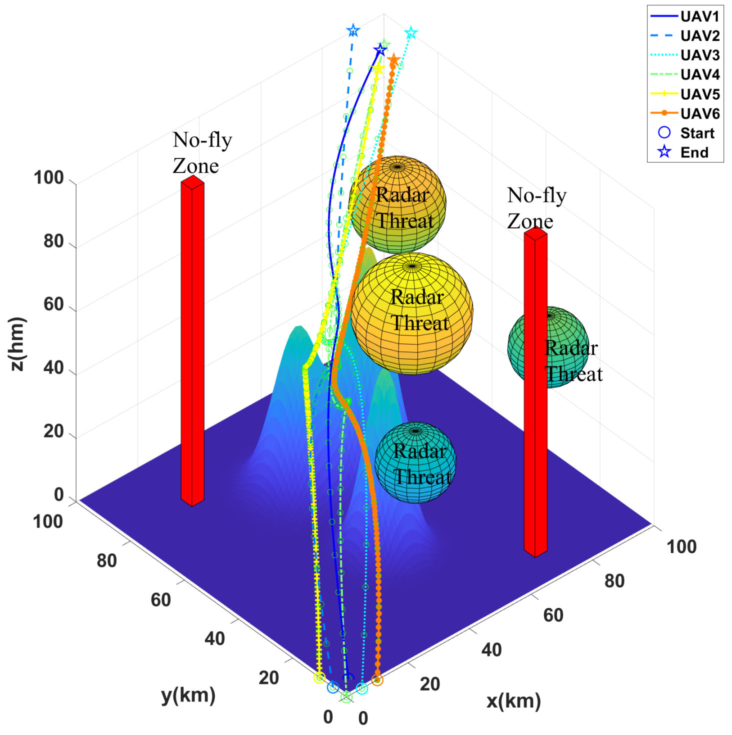

Figure 13. The result of multi-UAV path planning in the third scenario with 6 UAVs, 3 mountains, 4 radars, and 2 no-fly zones: compared with the first scenario, the number of radar threats increases to 4.

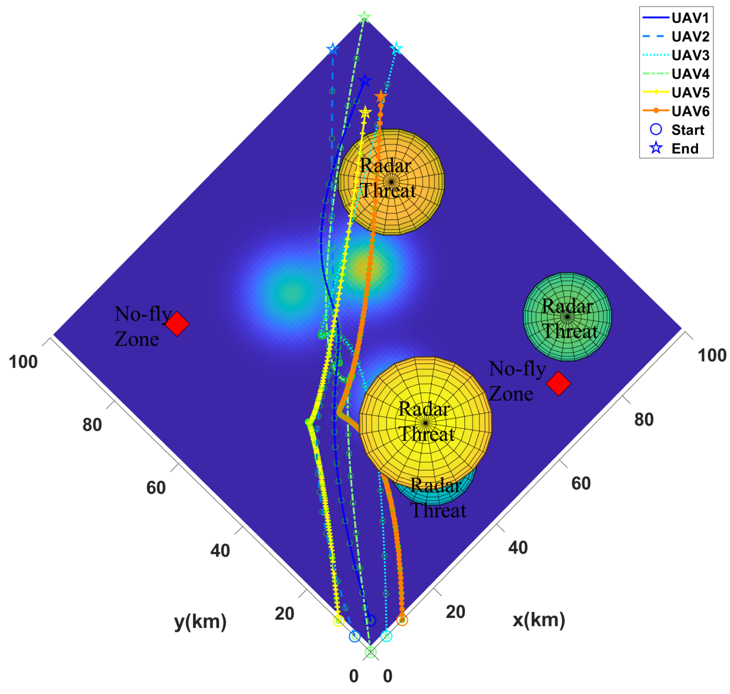

Figure 14. Top–down view of multi-UAV path planning in the third scenario with 6 UAVs, 3 mountains, 4 radars, and 2 no-fly zones: compared with the first scenario, the number of radar threats increases to 4.

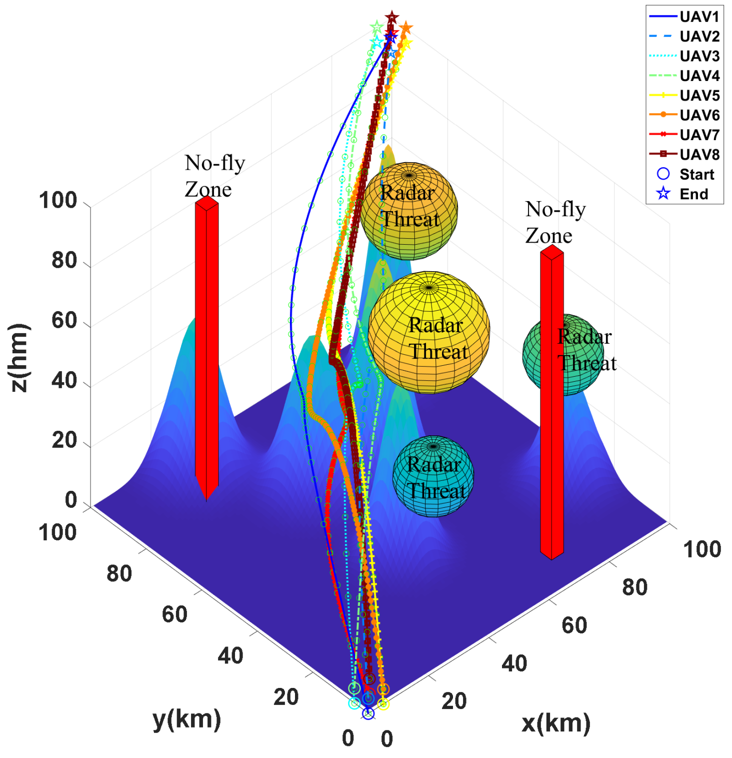

Figure 15. The result of multi-UAV path planning in the fourth scenario with 8 UAVs, 6 mountains, 4 radars, and 2 no-fly zones: compared with the first scenario, the numbers of UAVs, mountains, and radars all increased.

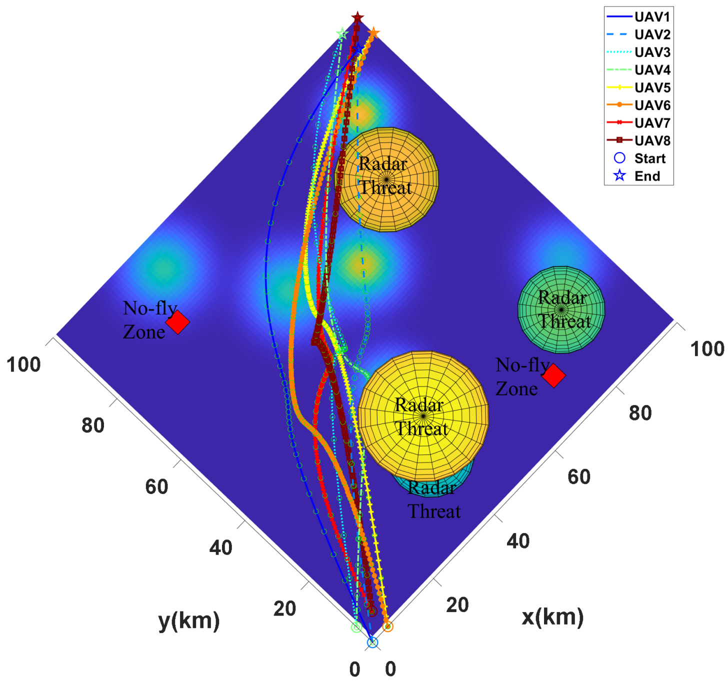

Figure 16. Top–down view of multi-UAV path planning in the fourth scenario with 8 UAVs, 6 mountains, 4 radars, and 2 no-fly zones: compared with the first scenario, the numbers of UAVs, mountains, and radars are all increased.

Table 2. Parameters in all algorithms.

Algorithm	Parameter	Value
PSO	w	1.2
	c_1	2
	c_2	2
	L	0
	U	100
GWO	L	0
	U	100
NOA	L	0
	U	100
INOA	δ	0.05
	P_{rp}	0.2
	L	0
	U	100

The comparison of fitness values is displayed in Figure 17, and the corresponding statics are given in Table 3. As shown in them, due to its lack of a mechanism to escape local optima, PSO is trapped in a local optimal solution and cannot escape, resulting in the least ideal performance. GWO lacks convergence velocity and does not converge to the global optimal solution within the number of iterations. NOA converges quickly and has strong multi-objective optimization capabilities such that it is able to find high-quality solutions in a relatively small number of iterations, showing obvious advantages over the other two algorithms, but there is still room for improvement. Based on it, INOA further increases the search efficiency by obtaining a higher-fitness initial solution through logistic mapping; then, it improves the quality of the initial solution through lens imaging inverse learning and subsequently enhances convergence speed and precision through parameter adjustment and the improved sine and cosine strategy. As a result, INOA can apparently improve the capability of both global search and local exploitation, strongly confirming its effectiveness in multi-UAV path planning.

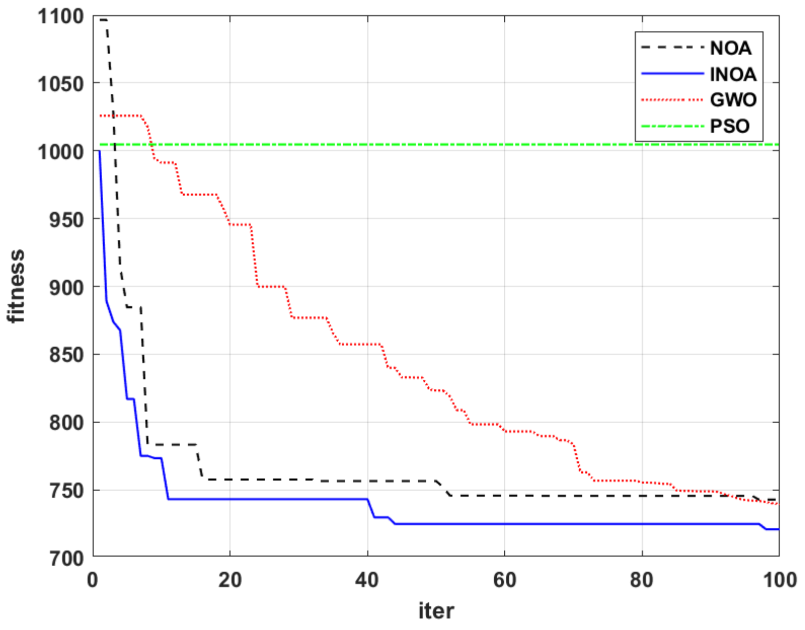

Figure 17. Fitness comparison: INOA achieves the best fitness compared with PSO, GWO, and NOA, with the improved search efficiency.

Table 3. Fitness statistics.

Algorithm	Maximum Fitness	Minimum Fitness	Average Fitness	Standard Deviation	Confidence Interval (95%)
NOA	759.5	743.0	752.1	5.8	(750.0, 754.2)
INOA	741.3	733.3	737.8	5.2	(735.9, 739.6)
GWO	759.9	749.1	756.1	8.8	(752.9, 759.2)
PSO	899.0	977.1	951.7	37.8	(938.2, 965.2)

6. Conclusions

The multi-UAV path planning problem faces many challenges. It was effectively modeled in this study with constraints such as path length, segment distance, smoothness, altitude, collision with mountains, inter-UAV collisions, radar detection, and no-fly zones, being converted into a multi-objective restricted optimization problem. To solve this problem, INOA is proposed with a more fit elite initialization population through logistic chaotic mapping initialization and a lens imaging inverse learning strategy. Furthermore, it has a balanced global search capability with a local exploitation capability through parameter adjustments and the integration of an improved sine–cosine search strategy. Simulation results on multi-UAV path planning showed that INOA significantly surpasses other algorithms with regard to convergence velocity and solution precision.

INOA addresses global path planning and can design a feasible general route for each UAV in practical applications. However, from the perspective of absolute safety, the research on the local planning of UAVs is necessary but not adequate enough. Further research will delve into local planning algorithms. The synthesis of these two methods would be applied into the real-world flight of UAVs with a more satisfactory performance.

Author Contributions: Conceptualization, H.Y. and C.X.; methodology, H.Y. and C.X.; software, C.X.; validation, C.X. and B.Z.; formal analysis, H.Y. and C.X.; investigation, C.X.; data curation, C.X.; writing—original draft preparation, C.X. and H.Y.; writing—review and editing, C.X. and H.Y.; visualization, C.X.; supervision, H.Y.; project administration, H.Y.; funding acquisition, H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundations of China (62003130) and Fundamental Research Funds for the Central Universities (B230201055).

Data Availability Statement: All data are presented in the article.

Acknowledgments: The authors would like to thank everyone who provided valuable suggestions and support to improve the content, quality, and presentation of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAV	Unmanned aerial vehicle
DQN	Deep q-learning
DDPG	Deep deterministic policy gradient
PPO	Proximal policy optimization
MDP	Markov decision process
D3QN	Dueling double deep Q-network
ACO	Ant colony algorithm
PSO	Particle swarm optimization
GA	Genetic algorithm
GWO	Grey wolf optimizer
NOA	Nutcracker optimization algorithm
HGJO	Hybrid strategy-based golden jackal optimizer
SSA	Sparrow search algorithm
GSO	Golden search optimization
DMSPSO	Dynamic multi-swarm PSO
CLPSO	Comprehensive learning PSO
INOA	Improved nutcracker optimization algorithm
CEC	Congress on evolutionary computation
HHO	Harris hawks optimization
SHO	Sea horse optimization

References

1. Han, L.; Wu, X.; Sun, X. Hybrid path planning algorithm for mobile robot based on a* algorithm fused with dwa. In Proceedings of the 2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 26–28 May 2023; Volume 3, pp. 1465–1469.
2. Luo, M.; Hou, X.; Yang, J. Surface Optimal Path Planning Using an Extended Dijkstra Algorithm. *IEEE Access* **2020**, *8*, 147827–147838. [CrossRef]
3. Yang, F.; Su, H. Research on Improving Artificial Potential Field for Drone Obstacle Avoidance. In Proceedings of the 2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT), Yichang, China, 20–22 September 2024; pp. 1–4. [CrossRef]
4. Jang, K.; Baek, J.; Park, S.; Park, J. Motion Planning for Closed-Chain Constraints Based on Probabilistic Roadmap with Improved Connectivity. *IEEE/ASME Trans. Mechatronics* **2022**, *27*, 2035–2043. [CrossRef]
5. Wang, W.; Li, J.; Bai, Z.; Wei, Z.; Peng, J. Toward Optimization of AGV Path Planning: An RRT*-ACO Algorithm. *IEEE Access* **2024**, *12*, 18387–18399. [CrossRef]
6. Jiang, W.; Bao, C.; Xu, G.; Wang, Y. Research on Autonomous Obstacle Avoidance and Target Tracking of UAV Based on Improved Dueling DQN Algorithm. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 5110–5115. [CrossRef]

7. Li, P.; Ding, X.; Ren, W. Research on Path Planning of Cloud Robot in Dynamic Environment Based on Improved DDPG Algorithm. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 3561–3566. [CrossRef]
8. Li, L.; Li, W.; Wang, J.; Chen, X.; Peng, Q.; Huang, W. UAV Trajectory Optimization for Spectrum Cartography: A PPO Approach. *IEEE Commun. Lett.* **2023**, *27*, 1575–1579. [CrossRef]
9. Li, B.; Wu, Y. Path Planning for UAV Ground Target Tracking via Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 29064–29074. [CrossRef]
10. Wang, X.; Gursoy, M.C.; Erpek, T.; Sagduyu, Y.E. Learning-Based UAV Path Planning for Data Collection with Integrated Collision Avoidance. *IEEE Internet Things J.* **2022**, *9*, 16663–16676. [CrossRef]
11. Chen, Y.; Bai, G.; Zhan, Y.; Hu, X.; Liu, J. Path Planning and Obstacle Avoiding of the USV Based on Improved ACO-APF Hybrid Algorithm with Adaptive Early-Warning. *IEEE Access* **2021**, *9*, 40728–40742. [CrossRef]
12. Tao, B.; Kim, J.H. Mobile robot path planning based on bi-population particle swarm optimization with random perturbation strategy. *J. King Saud Univ.-Comput. Inf. Sci.* **2024**, *36*, 101974. [CrossRef]
13. Pehlivanoglu, Y.V.; Pehlivanoğlu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Appl. Soft Comput.* **2021**, *112*, 107796. [CrossRef]
14. Zhang, W.; Zhang, S.; Wu, F.; Wang, Y. Path Planning of UAV Based on Improved Adaptive Grey Wolf Optimization Algorithm. *IEEE Access* **2021**, *9*, 89400–89411. [CrossRef]
15. Abdel-Basset, M.; Mohamed, R.; Jameel, M.; Abouhawwash, M. Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems. *Knowl.-Based Syst.* **2023**, *262*, 110248. [CrossRef]
16. Lou, T.S.; Yue, Z.P.; Jiao, Y.Z.; He, Z.D. A hybrid strategy-based GJO algorithm for robot path planning. *Expert Syst. Appl.* **2024**, *238*, 121975. [CrossRef]
17. Zhang, J.; Chen, D.; Han, G.; Qian, Y. Formation Path Planning for Collaborative Autonomous Underwater Vehicles Based on Consensus-Sparrow Search Algorithm. *IEEE Internet Things J.* **2023**, *11*, 13810–13823. [CrossRef]
18. Yu, X.; Jiang, N.; Wang, X.; Li, M. A hybrid algorithm based on grey wolf optimizer and differential evolution for UAV path planning. *Expert Syst. Appl.* **2023**, *215*, 119327. [CrossRef]
19. Xu, L.; Cao, X.; Du, W.; Li, Y. Cooperative path planning optimization for multiple UAVs with communication constraints. *Knowl.-Based Syst.* **2023**, *260*, 110164. [CrossRef]
20. Meng, K.; Chen, C.; Wu, T.; Xin, B.; Liang, M.; Deng, F. Evolutionary State Estimation-Based Multi-Strategy Jellyfish Search Algorithm for Multi-UAV Cooperative Path Planning. *IEEE Trans. Intell. Veh.* **2024**, 1–19. [CrossRef]
21. Mellinger, D.; Kumar, V. Minimum snap trajectory generation and control for quadrotors. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2520–2525. [CrossRef]