drones

MDPI

Article

Intelligent Scheduling Technology of Swarm Intelligence Algorithm for Drone Path Planning

Zhipeng Meng [1], Dongze Li [1], Yong Zhang [2,*] and Haoquan Yan [3]

[1] National Innovation Institute of Defense Technology, Academy of Military Sciences, Beijing 100071, China; niidtmzp@163.com (Z.M.); dongzeli1010@126.com (D.L.)
[2] Research Institute of Pilotless Aircraft, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[3] College of Automation, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; yanzigu@nuaa.edu.cn
* Correspondence: yongzhang@nuaa.edu.cn

Abstract: Different kinds of swarm intelligence algorithm obtain superior performances in solving complex optimization problems and have been widely used in path planning of drones. Due to their own characteristics, the optimization results may vary greatly in different dynamic environments. In this paper, a scheduling technology for swarm intelligence algorithms based on deep Q-learning is proposed to intelligently select algorithms to realize 3D path planning. It builds a unique path point database and two basic principles are proposed to guide model training. Path planning and network learning are separated by the proposed separation principle and the optimal selection principle ensures convergence of the model. Aiming at the problem of reward sparsity, the comprehensive cost of each path point in the whole track sequence is regarded as a dynamic reward. Through the investigation of dynamic environment conditions such as different distances and threats, the effectiveness of the proposed method is validated.

Keywords: algorithm scheduling; deep reinforcement learning; path planning; swarm intelligence algorithm

Citation: Meng, Z.; Li, D.; Zhang, Y.; Yan, H. Intelligent Scheduling Technology of Swarm Intelligence Algorithm for Drone Path Planning. *Drones* **2024**, *8*, 120. https://doi.org/10.3390/drones8040120

Academic Editor: Francesco Nex

Received: 3 February 2024
Revised: 8 March 2024
Accepted: 12 March 2024
Published: 26 March 2024

1. Introduction

Swarm intelligence (SI) represents the collective intelligent behavior of a class of decentralized self-organizing systems, and the macroscopic SI is shaped through the interaction of microscopic individuals [1,2]. SI algorithms obtain the possible solutions through stochastic searches in the feasible domain and have all solutions converge to a target optimal solution through iterative updates. The essence of SI algorithms involves a type of parallel computing with individual information communication. The number of individuals determines the scale of parallel computing, and information exchange between individuals determines the efficiency of parallel operation. The increasing complexity of optimization problems leads to the continuous development and improvement of optimization algorithms. On the one hand, new structural algorithms are constantly proposed. The purpose of SI algorithm development is to study the swarm intelligence behavior of natural organisms and simulate more efficient information communication methods [3,4]. On the other hand, the existing algorithms are constantly improved. They can be improved by introducing chaos theory [5] quantum theory [6,7], Lévy flight [8,9], and other theories, and they can also be improved by the fusion of SI algorithms [10].

Path planning is an important basis for drones to execute various tasks. It has the characteristics of high dimensionality and multi-constraints. Current methods for path planning can roughly be divided into two categories: Traditional algorithm and intelligent optimization algorithm. Typical traditional algorithms include the Voronoi diagram method [11] and signpost diagram method based on graph theory, fast random tree method

(RRT) [12] based on random sampling, and A* algorithm [13] based on heuristic information. They have a fast construction speed and a high route safety but have different drawbacks, such as being unable to adapt to a dynamic environment and being unable to be applied to three-dimensional scenes.

The SI algorithm is the typical representative of intelligent algorithms. Due to its fast planning speed, simple principle, and freedom from spatial dimensions, it has numerous applications in the estimation and setting of control parameters [14–17] fault diagnosis, task allocation, and flight path planning [18–20]. However, the no free lunch (NFL) [21] theorem logically proves that there is no suitable SI algorithm for solving all optimization problems. A particular SI algorithm may perform well on one kind, while poorly on another. NFL has created the research upsurge of SI algorithms, and various new algorithms and improvements emerge endlessly. But these studies on single algorithms cannot break the limit of NFL. So, this paper demonstrates the differences of SI algorithms by building deep Q-learning networks. Through the reasonable selection of multiple algorithms, the advantages of each SI algorithm are maximized.

Combined with deep learning nets and reinforcement learning (RL) [22], deep reinforcement learning (DRL) takes advantage of the feature representation capabilities of deep neural networks to fit RL functions [23], including state, action, and value, to improve the performance of RL models. Among different DRL methods, the deep Q-network (DQN) has achieved remarkable success. Mnih et al. proposed Deep Q-Net [24], the first DRL framework. Its neural network fits the table of the action–value pairs in the Q-learning algorithm into a function, which addresses the disadvantage of only being applicable to a finite state space. The DQN has two neural networks, the eval network and target network, which can improve the learning efficiency through different updating methods. The presence of experience pools can weaken the correlation between samples.

Since the proposal of DQN, various improved algorithms have been proposed. The dual deep Q-network (DDQN) [25] eliminates the problem of overestimation by decoupling the selection of the target Q-action and the calculation of the target Q-value. Its improvement is reflected in the structural optimization. Prioritizing experience, the Q-network [26] gives the priority of the samples and stores the value of this priority in the experience replay buffer pool. Its improvement is reflected in the experience pool optimization. The dual deep Q-network [27] illustrates the improvement of network structure. It replaces the last fully connected layer with two networks, the price function network and the dominance function network. The final output of the Q-network is obtained by a linear combination of the output of the price function network and the output of the dominance function network. Except for the DQN, policy-based RL algorithms which include the actor–critic framework (A3C), deep deterministic policy gradient (DDPG) [28,29], and proximal policy gradient optimization (PPO) algorithm [30] are also growing. The problem of continuous control, which Q-learning is unable to solve, is solved. Regarding data processing, Markov chain Monte Carlo (MCMC) [31] and approximate Bayesian computation (ABC) [32] are widely used.

In terms of path planning, with different threat changes and planning dimensions, a single SI algorithm struggles to determine the optimal value every time. For this reason, analysis of performance differentiation of SI algorithms and intelligent scheduling technology are increasingly important.

In this paper, a novel scheduling technology about swarm intelligence algorithms based on deep Q-learning (DQN-SISA) is proposed to intelligently select the optimal algorithm to realize 3D path planning when a drone faces different dynamic environments. Firstly, this study creates a library of SI algorithms including four typical algorithms, ant colony optimization (ACO) [33], particle swarm optimization (PSO) [34], grey wolf optimizer (GWO) [35], whale optimization algorithm (WOA) [36] and chaos particle swarm optimization (CPSO) [37]. Based on the establishment of the library, theories of algorithms are compared and analyzed from two aspects of structure and parameters. Secondly, path points are calculated by the SI algorithms as the state that is constructed to establish

a Markov process for path planning. The DQN is suitable for discrete action spaces and is selected to learn the performance differences of the SI algorithms with different path distances, numbers of threats, and other dynamic environments. Two principles are proposed for model training. A separation principle is proposed to separate the solution processes of the SI algorithms from the DQN training process by establishing a path point database such that the point data are reusable and the training time cost of the DQN is greatly reduced. An optimal extraction principle is proposed to solve the influence of the randomness of the SI algorithms on model convergence.

For the hyperparameters of the model, an adaptive greedy strategy is used to select the behavior to then improve the early exploration ability and late convergence ability of the model.

2. SI Algorithm Library for Path Planning

2.1. Theory of Selected Algorithms

Most SI algorithms follow considerably similar processes. The definition of a population is the first step, which is responsible for defining objects including population size, iteration time, and characteristic parameters such as the speed and position of the algorithm. The iterative loop is the most important phase, which typically includes the necessary boundary processing, fitness calculation and comparison, and information updating methods of the SI algorithm. The last step involves determining whether the loop terminates. If the maximum number of iterations is reached, the loop breaks, otherwise, the loop iterates. The criteria for ending the loop can be a pre-set number of iterations, minimum error labeling, or minimum accuracy requirements. The overall flow chart is shown in Figure 1.

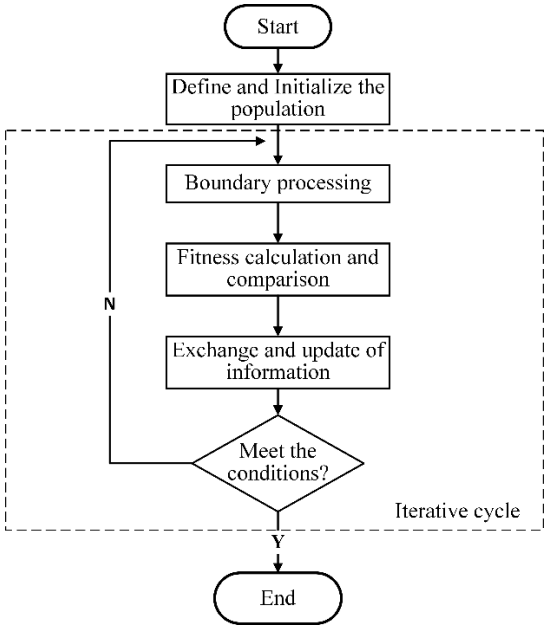

Figure 1. General flow chart of SI algorithm.

Mimicking the swarm intelligence behavior of ants in determining the closest route to target food, the ACO algorithm developed by Dorigo is the earliest SI algorithm with practical significance.

The pheromone updating formula of ACO is expressed as

$$\begin{aligned} \tau_{ij}(t+1) &= (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \\ \Delta\tau_{ij}(t) &= \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \end{aligned} \tag{1}$$

where $\tau_{ij}(t)$ denotes pheromone intensity at time t and ρ refers to the pheromone volatilization factor.

Inspired by the foraging behavior of birds, the PSO algorithm proposed by Eberhart is one of the most widely used SI algorithms.

The updating formula of position and velocity information of the standard PSO is expressed as

$$\begin{aligned} v &= \omega \cdot v + c_1 r_1 (pBest - x) + c_2 r_2 (gBest - x) \\ x &= x + v \end{aligned} \tag{2}$$

Here, w is the inertia weight, c_1 and c_2 are the learning rates, $pBest$ the historical optimal position of the corresponding particle, and $gBest$ the global optimal position in this iteration. As a coefficient, w determines the influence of the particle velocity of the previous iteration for this iteration.

Classical algorithms, including ACO and PSO, have a strong robustness and global search ability, but also a slow convergence speed and poor accuracy. From the perspective of model structure, the slow convergence speed is due to the single structure. There is no hierarchical stratification of information among individuals. Although the equivalence of information can enhance the search ability of the algorithm in the early stage, once the local optimum is found in the later stage, it is easy to break the balance between individuals and affect the adjacent individuals, so that the algorithm falls into the local optimum. Although the accuracy is low, due to a less random mechanism, it has a shorter computing time in the low-dimension route planning optimization problem.

Newer algorithms, such as GWO and WOA, whose superiority is attributed to the more diverse and efficient algorithm structure design, have fast convergence speeds and high accuracies and have been widely used in flight-path planning.

The GWO search and update formula of position information during its predation period is expressed as

$$\begin{aligned} &D = \left| C X_p(t) - X(t) \right| \\ &X(t+1) = X_p(t) - A \cdot D \\ &D_\alpha = \left| C_1 \cdot X_\alpha(t) - X(t) \right|, D_\beta = \left| C_2 \cdot X_\beta(t) - X(t) \right|, \\ &D_\xi = \left| C_3 \cdot X_\xi(t) - X(t) \right| \\ &X_1 = X_\alpha - A_1 \cdot D_\alpha, X_2 = X_\beta - A_2 \cdot D_\beta, \\ &X_3 = X_\xi - A_3 \cdot D_\xi \\ &X(t+1) = \frac{X_1 + X_2 + X_3}{3} \end{aligned} \tag{3}$$

Here, X_α, X_β, and X_δ denote the position vectors of wolves α, β, and δ in the current population; X is the location of individual grey wolves; D_α, D_β, and D_δ denote the distances between the current object wolf and the three candidate wolves; A and C are coefficient vectors; $X_n(t)$ refers to the position vector of the prey; D is the distance between the current grey wolf and the location of the prey.

Concerning WOA, the location information updating law for the two periods of encircling prey and forced attacks is expressed as:

$$\begin{aligned} &D = \left| \vec{C} \vec{X}^*(n) - \vec{X}(n) \right| \\ &\vec{X}(n+1) = \vec{X}^*(n) - \vec{A} \cdot \vec{D} \\ &\vec{X}(n+1) = \vec{d} e^{bl} \cos(2\pi l) + \vec{X}^*(n) \end{aligned} \tag{4}$$

Here, n denotes the number of iterations, $\vec{A}$ and $\vec{C}$ the updated coefficient vectors, $\vec{X}^*(n)$ the optimal position of the whale at iteration n, $\vec{X}(n)$ the position vector of the current whale. The first two formulas show the shrinkable encircling prey stage, and the final formula is the spiral position update.

$\left|\vec{d}\right| = \left|\vec{C}\vec{X}^{*}(n) - \vec{X}(n)\right|$ represents the distance between the whale and its prey. The role of random coefficient A is similar to that in GWO. When $|A| > 1$, the whale herd performs a decentralized search, and the whale position is updated randomly. The random updating formula is as follows:

$$\begin{aligned} K &= \left|C \times \vec{X}_{rand} - \vec{X}\right| \\ \vec{X}(n+1) &= \vec{X}_{rand} - \vec{A} \cdot \vec{K} \end{aligned} \tag{5}$$

WOA is one SI algorithm with many random search mechanisms, and the selection of the information updating formula is random. Many random search mechanisms can effectively improve the global search ability in the early stage of the algorithm, but simultaneously affect the stability of the algorithm performance and computation time in the late stages.

These four algorithms cover the early SI algorithms that have been widely used and new SI algorithms proposed in recent years. From the aspect of algorithm performance, they cover different performance requirements, such as strong robustness, strong convergence, and high accuracy. Most importantly, they are widely used in the field of drone cluster control. In summary, the four algorithms were selected as candidate algorithms and incorporated into the algorithm scheduling library for the DQN to study algorithm scheduling technology.

2.2. Modeling Environments and Threats

2.2.1. Digital Maps with Terrain Threats

An appropriate planning space must be established in accordance with the flight environment and mission requirements for terrain models. The former is expressed as digital elevation maps (DEMs) adopting the form of a group of ordered numerical arrays to represent the ground elevation. A mountain background is taken as a task environment, and a DEM is established using a random function to simulate peaks and other threat obstacles. The mountain model function was proposed in Ref. [31]. The former is expressed as:

$$\begin{aligned} z(x,y) = \sin(y+a) + b \cdot \sin(x) + c \cdot \cos(d \cdot \sqrt{x^2+y^2}) \\ + e \cdot \cos(y) + f \sin(f \cdot \sqrt{x^2+y^2}) + g \cdot \cos(y) \end{aligned} \tag{6}$$

where (x, y) refer to the point coordinates on a horizontal projection plane; z refers to the height coordinate that corresponds to (x, y); a, b, c, d, e, f are the coefficients. Changing the coefficients can obtain different landforms.

The threat cost $T_{terrain}$ of a certain peak k is expressed as

$$T_{terrain} = \begin{cases} R_t(h) + d_{\min} - d & , d < (R_t(h) + d_{\min}) \text{ and } h < H(k) \\ 0 & , d > (R_t(h) + d_{\min}) \text{ or } h > H(k) \end{cases} \tag{7}$$

where d is the distance from the drone to the symmetrical axis of the peak, $d_{\min}$ the minimum distance allowed on the terrain, $H(k)$ the height of peak k, R_t the maximum extension radius. The calculation formula of $R_t(h)$ is expressed as

$$R_t(h) = (H(k) - h) / \tan\theta \tag{8}$$

where θ the slope of the terrain.

The relationship between the variables in the formula of the mountain threat is shown in Figure 2.

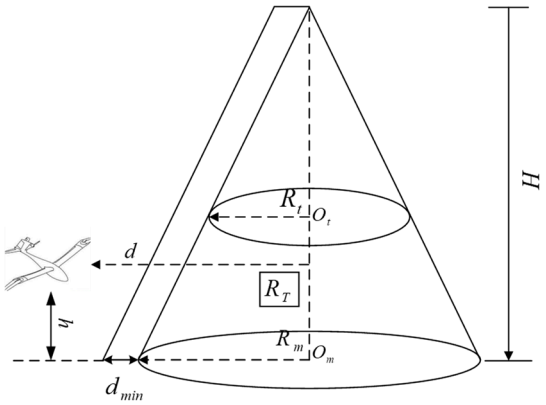

Figure 2. Schematic Diagram of Mountain Threat.

2.2.2. Radar Threat

Radar is the most common threat in battlefield scenarios that transmits directional electromagnetic waves to search and analyze objects in a space, obtaining information about the direction, altitude, and speed of objects. The signal strength of the radar is the same in all directions and satisfies the signal-to-noise ratio (SNR) formula as follows:

$$S/N = \frac{P_t G_r G_t P \lambda^2 \sigma}{(4\pi)^3 K_s L_m B_n T_s d_R^4} \tag{9}$$

Here, S/N denotes the signal-to-noise ratio of the radar, P_t the transmitter power, G_r the gain of the receiving antenna, G_t the transmit antenna gain, σ the effective radar scattering area of the detected target, λ the operating wavelength, K_s the Boltzmann constant, L_m the loss factor, B_n the bandwidth, T_s the thermodynamic temperature, and d_R denotes the distance between the drone and radar source. All parameters except d_R can be regarded as constants. Thus, the SNR formula is simplified as follows:

$$S/N = K_a \cdot \frac{\sigma}{d_R^4} \qquad (K_a = \frac{P_t G_r G_t P \lambda^2}{(4\pi)^3 K_s L_m B_n T_s}) \tag{10}$$

3. Algorithm Scheduling Technology Based on DQN

3.1. Framework of DQN-SISA

In this section, the training flow of the DQN-SISA model is designed and it is shown in Figure 3.

As shown in Figure 3, in an episode, the agent first greedily selects an action based on the current state S. When the random number is larger than the default greedy value, the model chooses behavior A according to the maximum Q-value. When the random value is less than the greedy value, the model randomly selects the four algorithms. Based on the selected behavior A, the scheduled SI algorithm completes the path planning and outputs immediate reward R, the next point, and provides the next state. Subsequently, if the experience replay buffer is not full, the experience replay buffer should store the experience data in the form $(state, action, reward, next\_reward)$, and the next path point is used to determine the final state. The final state is determined as follows: If the agent does not reach the final target point, the program transfers to the next state according to the relative position obtained from the next path point and then repeats the previous operation. The episode ends if the agent reaches the final target point. When the experience replay buffer reaches its maximum capacity, the agent randomly selects experiences consistent with the batch sample size for learning. At this stage, the parameters of the evaluation network are updated by a gradient descent method.

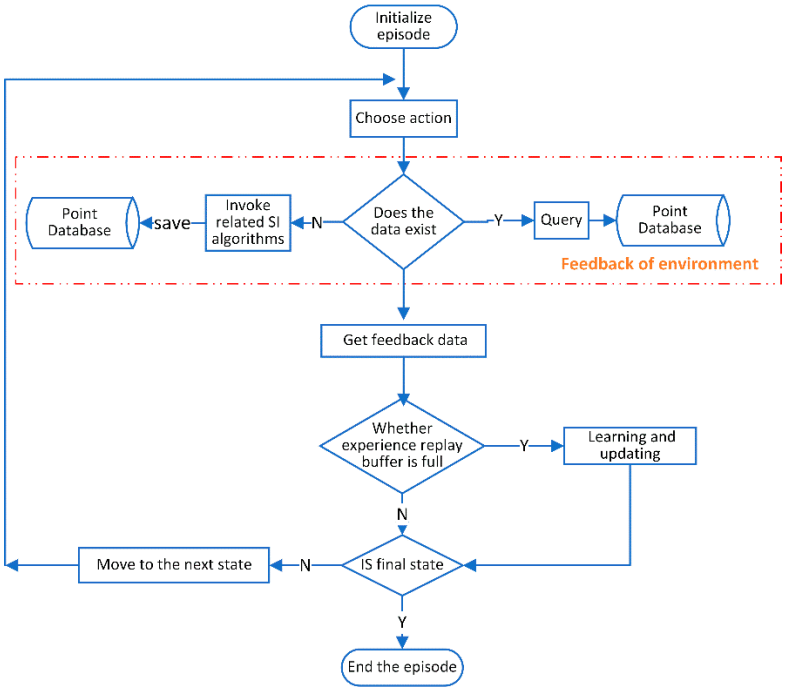

Figure 3. Training flow of DQN-SISA model.

Applying the DQN model directly to algorithm scheduling is evidently impractical, and the following problems must be considered. The randomness of the optimization algorithm makes forming a fixed mapping relationship between action and state difficult, which affects the learning efficiency of the DQN model and even the convergence ability. The computation times of the SI algorithms also multiply the cost of training the model by tens or thousands of episodes. Therefore, this study proposes two basic principles and builds a node database to solve the above problems and the details are presented in the following sections.

3.1.1. Network Structure of DQN for Path Planning

Deep reinforcement Q-learning is a type of method based on a Markov decision process (MDP) to solve temporal decision problems. Obviously, path planning satisfies the Markov property. When a drone interacts with a threat environment, the point state value at the next moment is only related to the current state and action, irrespective of the previous state and action. The MDP can be represented by a quintuple $< S, A, R, T, \gamma >$, where S (state) represents an environment consisting of a finite set of known states, and A (action) represents a finite set of actions that can be taken in the behavior space. The reward function R defines the immediate reward for a state transition, and γ is the reward discount factor. State transition function T represents the probability distribution function that maps a state–action pair to a possible successor state, known as the policy π. At time t, the policy function of current state s executes behavior a as follows:

$$\pi(a|s) = P(a_t = a|s_t = s) \tag{11}$$

In Q-learning, the Q-value determines the expected reward obtained by state s after taking action a under policy π, and its formula $Q^{\pi}(s, a)$ is expressed as:

$$Q^{\pi}(s,a) = E_{\pi}[G_t|s_t = s, a_t = a] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k R_t + k + 1|s_t = s, a_t = a] \tag{12}$$

The expected reward Gt is related to current and past rewards. The current action provides a delay reward $Rt + 1$, and the subsequent reward is multiplied by discount factor

γ, whose number of powers is divided by the interval. In practice, the time-difference control algorithm of the offline strategy of the Q-function is as follows:

$$Q^{\pi}(s,a) = Q^{\pi}(s,a) + \alpha(R + \gamma \max Q^{\pi}(s',a) - Q^{\pi}(s,a)) \quad (13)$$

The current Q-value of Equation (11) is updated depending on the maximum Q-value of the next state. However, the actual selection of an action–state transition follows a greedy strategy with random selection. The difference between two policies is an important basis for classifying Q-learning as off-policy.

Figure 4 shows the network structure of the DQN model for path planning. The DQN uses a convolutional neural network (CNN) instead of a table to store the Q-values and expands the storage range through function fitting to broaden Q-learning to continuous state spaces. Moreover, two networks are used for interactive learning. The eval network updates network parameters with each action selection and state transition when the target network does not train in real time. After a certain number of training iterations, the target network is updated by copying the parameters of the training network. The eval network is trained by calculating the mean square deviation loss function, and the parameters of the Q-network are updated through gradient backpropagation of the neural network. In addition, the DQN creates the experience replay buffer to accumulate experience data and reduces the correlation between agent actions using the experience data and random sampling. This method helps reduce the overfitting of neural networks and facilitates more extensive learning.

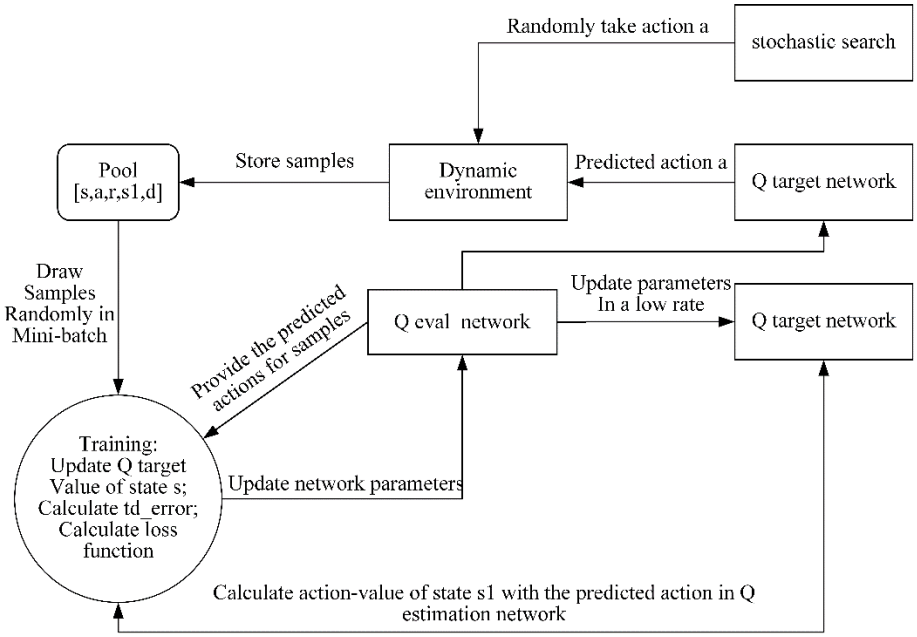

Figure 4. Network structure of DQN model.

3.1.2. Database of Equidistant Path Points

Similar to the experience replay buffer in the Figure 3, a path point database stores a series of path points obtained by the SI algorithms in the form (*now_point*, *next_point*, *reward*) that includes the current location of the drone *now_point*, calculated next point *next_point*, and *reward*. The path points obtained from one network training can be directly used for the next network training under the condition that the starting point and the target point are unchanged.

For this database, the coverage of the database can be improved by expanding the voyage distance of the initial point, intermediate target, and final target points such that the model using the database is more general. In the actual training process, the DQN model only requires action A and *now_po*int to index the next path point and correspond-

ing reward from the database and finally complete the data exchange and environment interaction. If no interactive data are in database, the SI algorithm corresponding to A is directly invoked to execute real-time path planning, and the data are stored in the database after the solution. Thus, the database continues to expand during training.

The SI algorithm obtains the number of nodes by using the method of dividing coordinates with fixed step. The method of equal division of X-axis coordinates is shown in Figure 5.

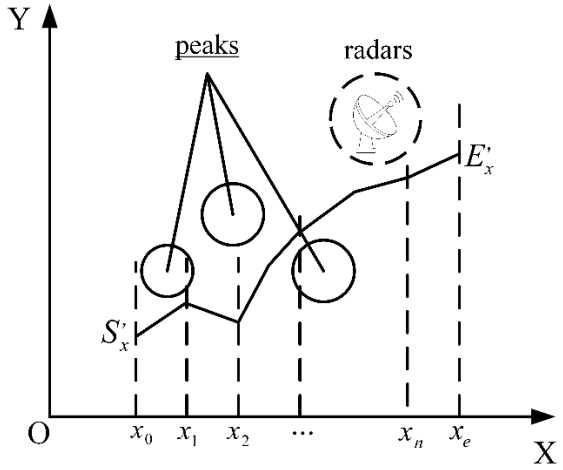

Figure 5. Equal division of X-axis coordinates.

Using the X-axis coordinate or Y-axis coordinate alone will cause the risk of algorithm model failure. If only X coordinates are used, the model will fail when the X-axis coordinates of the starting point and target point of the UAV are equal. Moreover, when the change of X coordinates is much smaller than that of Y coordinates, the calculated planned flight track points are too few, resulting in the flight track having no flight value. In order to prevent the above problems, this paper adopts the joint coordinate method. The maximum value of the X coordinate and Y-axis coordinate under a fixed step size is selected as the number of nodes to be planned.

The number of points changes with the distance from the target point. At the same distance, the change of step size will also cause the change of planning dimension. Each time the drone reaches a node, it will call the algorithm to execute the path planning from the current position to the final target point until it reaches the final target point. Arriving at each node is accompanied by path planning with different dimensions and threats. In this manner, the DQN-SISA model can indirectly learn the performance of the SI algorithms in different environments such that the algorithm model has general applicability.

3.1.3. Separation and Optimal Selection Principles

Satisfying the requirements of real-time performance and stability while processing long distances and complex flight-path planning is difficult. To satisfy these real-time requirements, the convergence times of SI algorithms must be limited, which makes it difficult for algorithms to obtain stable convergence solutions. Therefore, determining the mapping relationship between the four algorithms and path points is difficult, which leads to the difficulty of model convergence.

The optimal selection principle is proposed to solve the randomness problem. It means when database data are indexed by action a and now_point, if the path point database has multiple matching data, the model follows the principle of optimal selection and uses a sorting comparison to select the next point with the best reward. The optimal selection principle involves forming a unique mapping relationship between action A and the next state to reduce the influence of the randomness of the SI algorithms on the convergence of the DQN model. Note that the optimal selection principle plays a role in optimizing the

DQN training results, and the specific influence is shown in the following comparison of simulation results.

The point database established in the above section corresponds to the proposal of the separation principle. By establishing a path point database, the two processes of SI algorithm path-planning and DQN model training are separated such that the solution results of the SI algorithms can be reused, and the time sunk cost of model training can be significantly reduced.

3.2. Markov Process for DQN-SISA

3.2.1. State and Action Spaces

(1) State space

The SI algorithm exists as a kind of algorithm which must be abstracted through the state space to establish a real Markov process that reflects the performance difference of SI algorithms. DQN-SISA selects the node coordinates of path planning as the state space. The starting and target points form corresponding relations with the initial and final states, and the state transition of the algorithm is transformed into a typical position state transition. Finally, a complete Markov chain is formed. In order to find the path from the starting point to the target point in any environment, the drone cannot directly use the absolute coordinate information. Directly using the original coordinate information reduces the fitting degree of the model. Solidifying the information of the target point in the model is easy, and reaching the new target point in the new environment will be difficult.

This study uses relative coordinates as P_{state} and its formula is expressed as

$$\begin{aligned} P_{uav} &= [x_{uav},\ y_{uav},\ z_{uav}] \\ P_t &= [x_t,\ y_t,\ z_t] \\ P_{state} &= [x_{uav} - x_t,\ y_{uav} - y_t,\ z_{uav} - z_t] \end{aligned} \tag{14}$$

where P_{uav} is the original drone coordinate; P_t is the target coordinate.

The drone has a pre-defined sensing range. With the change of position coordinates, the number of threats and planning dimensions will also change dynamically, thus introducing dynamic environment input for the whole model.

(2) Action space

DQN-SISA action space is discrete, and the code 0–3 is used to replace the four SI algorithms. The action space is defined as a finite set $\{0,\ 1,\ 2,\ 3\}$.

3.2.2. Episode Reward and Q-Function

DQN-SISA adopts the weighted sum of the path cost and running time of the SI algorithm as the immediate reward R, and the episode reward $R_{episode}$ for an episode is the sum of all immediate R rewards for a whole training episode.

Step size can increase or decrease the number of points of different path lengths, which can satisfy the performance selection between high-precision solutions and quick path planning calculations. The real-time reward R formula is denoted as

$$R_{episode} = \sum_{t}^{N} R_t = -\sum_{t}^{N} (\alpha_1 J_{\cos t} + \beta_1 J_{time}) \tag{15}$$

where N refers to the number of points, α and β are the weight coefficients, and the specific value depends on the performance requirements. If the threat environment is complex and the flight path is long, the safe arrival of the drone should be guaranteed first, and the threat and flight-path costs should be reduced such that the size of α can be appropriately increased. Conversely, for short distances and high real-time requirements, the J_{cost} of each point is the path cost is calculated by the algorithm from the current point to the final goal point. A state transfer and real-time reward are obtained when each point is reached.

A fixed step is a partition to isolate the state of each path point, which solves the problem of sparse rewards such that the step obtained by each point has a guiding effect on the final target point. Since the SI algorithm requires solving the minimum J_{cost} as the optimization direction, it is negatively related to the pursuit of the maximum reward by the DQN when interacting with the environment. So, a negative sign has been added to the formula to represent this.

The path cost J_{cost} comprehensively considers the constraints of the drone and the environment. The constraints of the drone include length range *path*, climb angle *angle*, and altitude *heigth*. The threat environmental constraints include: Terrain threats $T_{terrain}$ and radar threats T_{radar}. In conclusion, the path planning cost J_{cost} formula is as follows:

$$J_{cost} = w_1 J_{path} + w_2 J_{angle} + w_3 J_{height} + w_4 J_{threat} \tag{16}$$

where w_1, w_2, w_3, w_4 are the weight coefficients of each cost, and J_{threat} is the total cost of the threat environment constraint, including $T_{mountain}$ and T_{radar}.

In Section 2, the formula of the radar SNR showed that the detection threat of radar is inversely proportional to the fourth power of the distance. Thus, the radar threat can be simplified with T_{radar}:

$$T_{radar} = \frac{K}{d_R^4} \tag{17}$$

Here, K is a constant parameter. By combining the mountain threats $T_{mountain}$, we can obtain a new calculation formula $J_{threat} = T_{mountain} + T_{radar}$.

According to the defined real-time reward R, the $Q_\pi(s_t, a_t)$ of the DQN-SISA model is rewritten as follows:

$$\begin{aligned} Q_\pi(s_t, a_t) &= R_t + \gamma R_t + 1 + \ldots = \sum_{\tau=0}^{N} \gamma^\tau R_t + 1 + \tau \\ &= \sum_{\tau=i}^{Nt} \gamma^\tau [\alpha_1 (w_1 J_{path} + w_2 J_{\cos t} + w_3 J_{height} + w_4 J_{threat}) + \beta_1 J_{time}] \end{aligned} \tag{18}$$

The formula of $Q_\pi(s_t, a_t)$ can be understood in Figure 6.

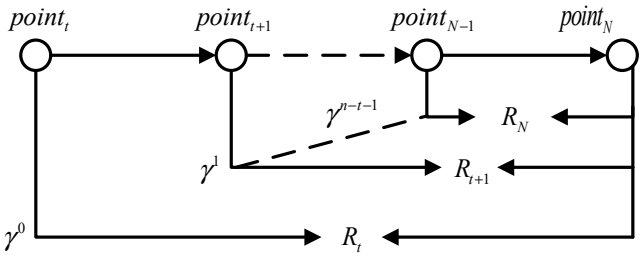

Figure 6. $Q(t)$ of path planning.

DQN−SISA does not offer an additional reward if the drone reaches a target point, which is convenient for the simulations and comparisons with the solution results of a single SI algorithm.

3.3. Use of the Point Database and Model

In practice, DQN-SISA exists as a black box. During model training, the fixed step is used as a tool to obtain position information, whereas during the process of the DQN-SISA model, the current position is obtained by the on-board sensor as input. The model no longer outputs the next path point of an artificially constructed position state transition but outputs a complete point sequence. If a dynamic change is not evident in the environment, the sequence is directly executed. Since the real-time reward is directly related to the path cost and is selected according to the maximum Q-value, the sequence planned at one time remains relatively optimal. When the drone is in a dynamic environment, an airborne

sensor is used to update the environmental data and the flight control system selects fixed time or distance step to call the module to output a sequence of new path points for the new dynamic environment.

4. Simulation Validation

4.1. Parameter Selection

The simulation experiment platform was on a Lenovo notebook AIR14 computer with a running memory of 16 G, AMD Ryzen 4800U CPU, and an AMD integrated graphics card of 512 MB. The DEM had a range of 100 km × 100 km × 300 m, and parameters of the original digital terrain model were set to $a = 0.1$, $b = 0.01$, $c = 1$, $d = 0.1$, $e = 0.2$, $f = 0.4$, and $g = 0.02$. The parameters of each SI algorithm were set according those shown in Table 1.

Table 1. SI algorithm parameter settings.

Algorithm	Population	Iterations	Characteristic Parameters
ACO	30	90	transition probability p0 = 0.2, pheromone volatilization factor rou = 0.8, original pheromone tau0 = 0.3;
PSO	30	90	learning rates c1 = c2 = 2, inertia weight w = 0.6, velocity v = [−1,1];
GWO	30	90	A = 2 − 2t/90
WOA	30	90	a1 = 2 − 2t/90, a2 = −1 − t/90

In addition to the parameters of the SI algorithms, the hyperparameters of the DQN were set according those shown in Table 2.

Table 2. Hyperparameters of the DQN algorithm.

Replay Memory	Episodes	Greedy	Update Interval	Learning Rate	Batch Size
6×10^4	30,000	0.8–1.0	400	0.001	128

4.2. Model Validation

In order to simulate a real threat environment, a starting point, intermediate, and final target point were randomly initialized, and 2 radar threats were randomly selected from the map data for path planning. In addition, the simulation results of the four algorithms need to be calculated separately to facilitate data comparison.

DQN-SISA designed in this section consisted of four network layers, with 128 × 128 neurons in the input layer, 128 × 128 neurons in the first fully connected layer, 128 × 64 neurons in the second fully connected layer, and 64 × 4 neurons in the output layer. The activation function chooses the rectified linear activation function (ReLU), and the adaptive formula of the greedy strategy is expressed as

$$Greedy = 0.8 + (episode - 10000)/10000 \times 0.1 \quad (19)$$

With a starting point at (8, 2, 60), intermediate point at (20, 35, 65), and final target point at (40, 80, 75), Figure 6 shows the model reward $R_{episode}$ convergence curve.

In Figure 7, after 100 hundred episodes, which means 10,000 training episodes, the experience replay buffer has sufficient experience data to perform network learning, and the reward rapidly increases and converges. Rewards can converge quickly and level off as the greedy behavior increases. The optimal convergence solution is −322.5979.

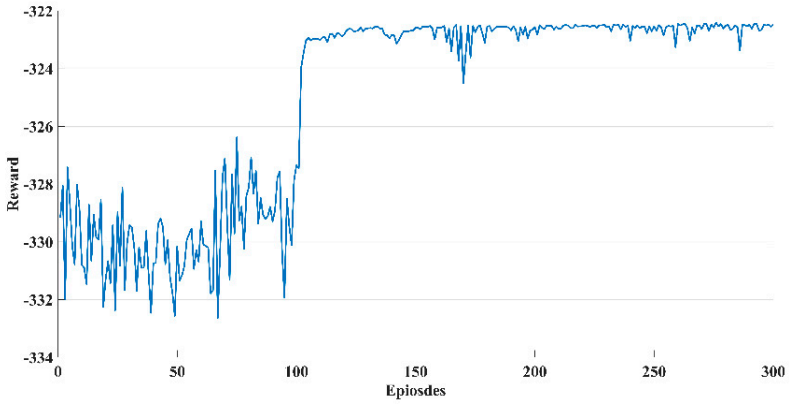

Figure 7. Reward convergence curve of DQN−SISA.

For data comparison, ACO, PSO, WOA, and GWO were used separately in sequence to obtain the solving reward of one episode. The rewards $R_{episode}$ obtained were −585.0233, −405.8917, −402.6372, and −409.554, by testing 100 times and taking the average value. Obviously, DQN-SISA achieves a better reward than any single algorithm, with a reward improvement rate of nearly 20% compared with the single optimal solution re−ward of −402.6372. The effectiveness of the algorithm model is verified. The three−dimensional path planning and contour map of each SI algorithm are displayed in Figure 8, and the result of DQN-SISA is shown separately in Figure 9.

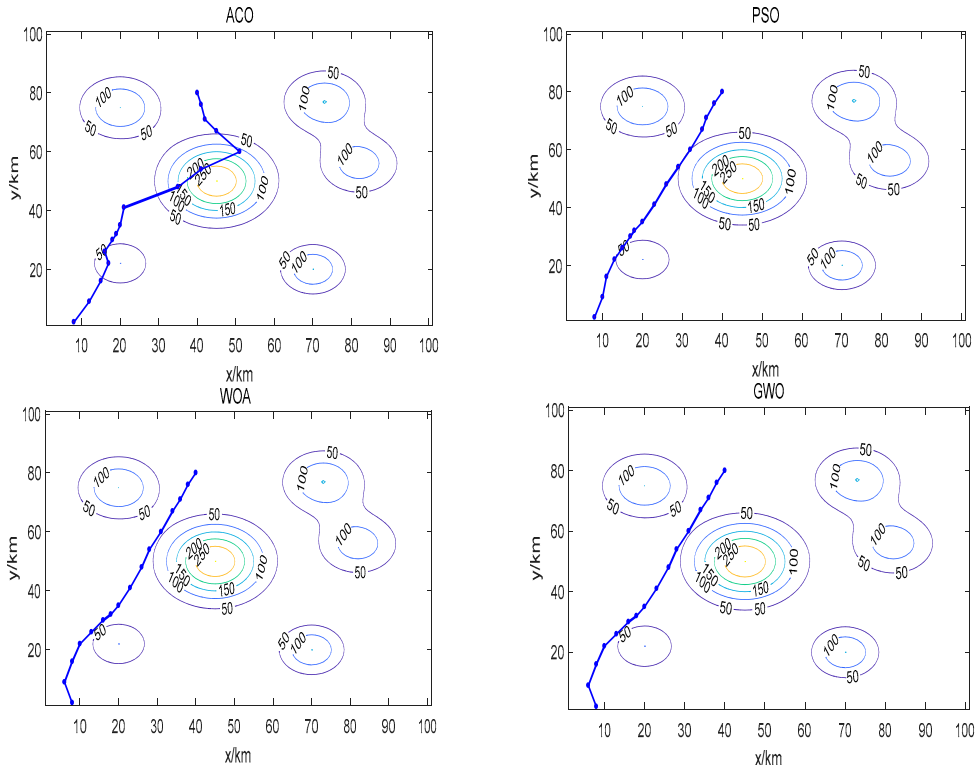

Figure 8. Paths of all SI algorithms.

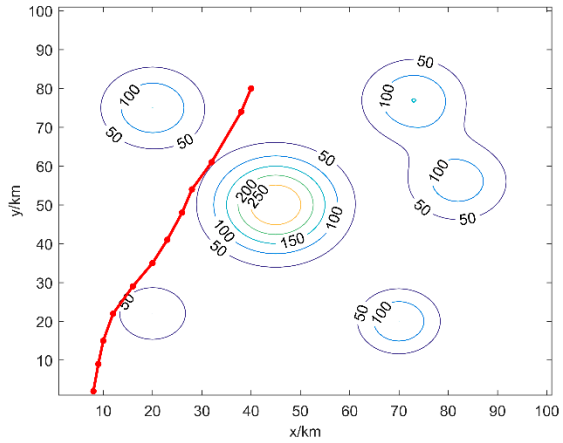

Figure 9. Path planned by DQN-SISA.

Each path point settlement incurs a significant time cost. The path shown in Figure 9 has fewer path points, indicating that DQN-SISA can maximize the efficiency of saving unnecessary computational costs. In addition, this path is smoother than the others.

4.3. Dynamic Environment Verification

This section defines two terms for data analysis: Optimal rate and suboptimal rate. The optimal rate represents the success rate of DQN-SISA in obtaining episode rewards that are better than the four single SI algorithms, and the suboptimal rate is the success rate of model rewards that are second only to the optimal rewards of the four SI algorithms.

Starting or target points are changed and 50 simulations are performed to verify whether DQN-SISA can adapt to the change of planning dimensions. When the node step size is unchanged, different distances mean that the number of nodes to be planned is different, that is, the planning dimensions are different. Table 3 summarizes the comparison data between the model and single SI algorithm.

Table 3. Change in the position of the intermediate target point and the starting point.

Test Description	Starting and Target Points	Optimal Rate	Suboptimal Rate
Initial start and target points	starting point: (8, 2, 60) target point: (20, 35, 65), (40, 80, 75)	100%	100%
Change intermediate target point	starting point: (8, 2, 60) target point: (35, 75, 65), (40, 80, 75)	98%	100%
Change starting point	starting point: (15, 5, 60) target point: (20, 35, 65), (40, 80, 75)	98%	100%

Table 3 shows the test results with the path point database. Since the database contains the final target points during DQN-SISA model training, the final target points are unchanged.

The results show that the optimal rate of DQN-SISA is 100%, and the optimal rate of DQN-SISA remains over 95%, with the suboptimal rate near 100%. The existence of a path point database saves solution time. In this case, the time DQN-SISA requires to complete the path planning of all path points from the original starting point to the target point is 2.47 s. This involves 11 points in the state space; thus, the time required using DQN-SISA technology is 0.224 s. Note that the points in the state space are selected from the path points of flight-path planning, and the number of flight-path planning points in this study is three times that of the former.

Variation in the number of threats is also an important indication of a dynamic environment. Table 4 shows the results of changing the number of radars without the path

point database. Therefore, the dynamic environment in Table 4 shows the real simulation results without using the database.

Table 4. Changing the number of radar threats.

Test Description	Positions of Radar Threats	Optimal Rate	Suboptimal Rate
Two radar threats	(40, 10, 5), (10, 60, 15)	56%	80%
Three radar threats	(40, 10, 5), (10, 60, 15), (60, 60, 30)	52%	80%
One radar threat	(10, 60, 15)	64%	96%

The optimal and suboptimal rates are significantly reduced, but the reward gap of the optimal solution is less than 1%, which still proves DQN-SISA has optimization ability. Furthermore, DQN-SISA still maintains a suboptimal rate of more than 80%. This method effectively avoids the situation where a single algorithm will fail and verifies the success of redundancy design.

4.4. Comparative Performance Study

The optimality principle not only ensures the convergence of the model but also exaggerates the actual results that creates a new problem. For the sake of technical rigor, a separate discussion of the optimal selection principle is still necessary. The experimental results in this section are obtained without the database. In other words, there is no optimality principle.

Firstly, it is necessary to discuss the advantages and disadvantages of the algorithm in different dynamic environments. The optimal rates of each algorithm at different distances are summarized in Figure 10. The optimal rates of each algorithm at different numbers of radars are summarized in Figure 11. It is worth noting that the test distance refers to the Euclidean distance, and all test results are the average values obtained after 50 text times.

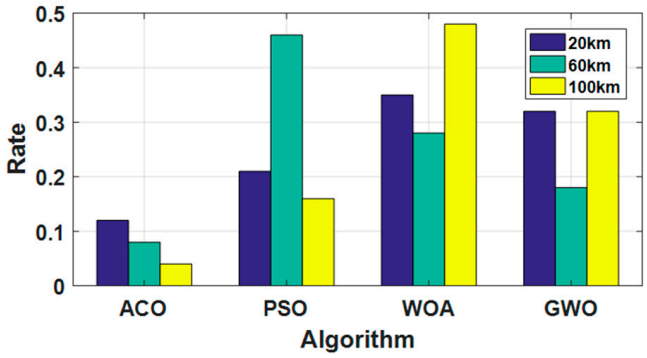

Figure 10. Optimal rates of each algorithm at different distances.

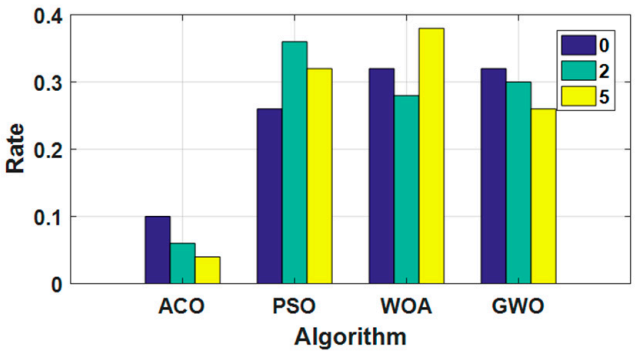

Figure 11. Optimal rates of each algorithm at different numbers of radars.

Compared with each other, the variation of the number of tested radars has little influence on the optimal rate. In the case of a short distance or low risk, the advantages and disadvantages of algorithms are not obvious.

Table 5 shows specific data about Figure 10 and the new DQN-SISA technology. Table 6 shows specific data about Figure 11 and DQN-SISA.

Table 5. Text data about different distances.

Distance/km	ACO	PSO	WOA	GWO	DQN-SISA
20	−66.9389	−57.0966	−56.9566	−56.9566	−56.8932
40	−223.0057	−196.9844	−197.853	−198.8895	−196.9767
60	−585.0233	−402.6372	−405.8917	−409.554	−400.1349
80	−856.9973	−637.824	−605.5098	−631.9259	−618.375
100	−1331.3168	−975.9081	−932.2784	−948.8938	−913.3382
120	−1688.0116	−1236.6424	−1181.642	1213.7527	−1058.5061

Table 6. Text data about different numbers of radars.

Number of Radars	ACO	PSO	WOA	GWO	DQN-SISA
0	−527.8471	−387.6395	−389.5866	−393.403	−372.5513
1	−543.8884	−398.9181	−399.4087	−404.5372	−395.4596
2	−552.2305	−409.0755	−406.0639	−409.2116	−400.9123
3	−592.1842	−411.3604	−409.2053	−414.0574	−375.7854
4	−623.1587	−414.0406	−412.8543	−418.0748	−406.1839
5	−632.8925	−419.6303	−418.0416	−422.0225	−413.3023

The optimal rates of the four algorithms for each of the above dynamic environments are summarized in a single figure, and the proportion by which DQN-SISA outperforms all SI algorithms is added. The final result is shown in Figure 12.

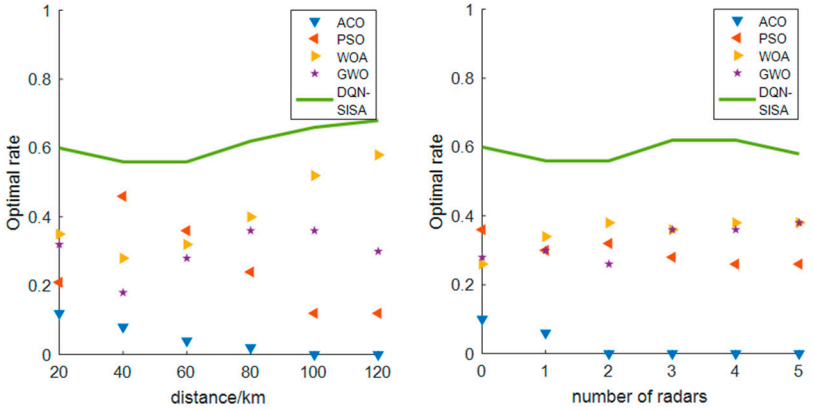

Figure 12. Comprehensive comparison results.

As shown in Figure 12, DQN-SISA still has high optimal and suboptimal rates for actual route planning without data. Especially, with the increase in the Euclidean distance of the planned path, the proportion of DQNs obtaining the optimal solution also increases steadily.

It is worth noting that under the condition that all algorithms are not invalid, the ACO results are always the worst. It cannot be regarded as a mistake to select ACO as a candidate algorithm. The disadvantage of ACO in accuracy and convergence makes it produce a wider range of path point sequences, which plays an important role in improving the coverage of the path point database. In this manner, ACO has a "catfish effect".

Considering the actual environment, we synthesize these experiments and change the number and positions of radar threats and the starting and target point positions to create a comprehensive dynamic environment. The number of radar threats is limited to a range of [0–5], and the starting and target points are limited to the range of the model training map. After 100 simulations, we obtain the optimal solutions of each SI algorithm and DQN-SISA. Figure 13 shows the percentages of optimal solutions. The dynamic simulation evidently verifies the feasibility of DQN-SISA.

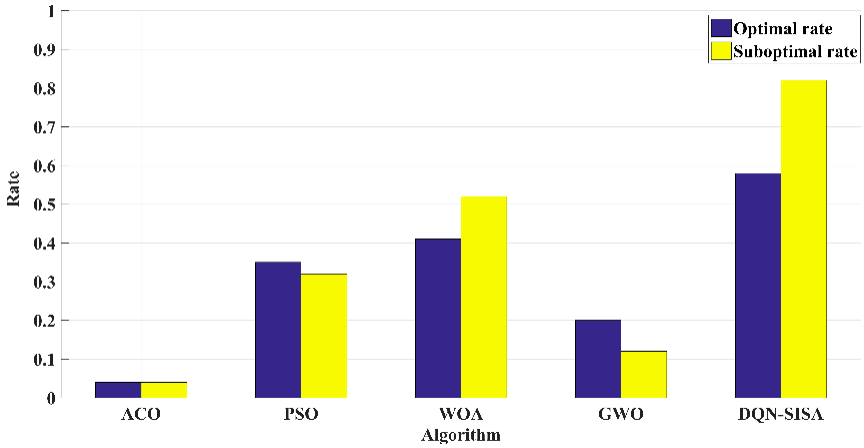

Figure 13. Text results for comprehensive dynamic scene.

5. Conclusions

This study proposed an intelligent scheduling technology, DQN-SISA, based on SI algorithms for drone trajectory planning. The differential optimization performance of SI algorithms in different dynamic scenarios is analyzed. DQN-SISA enables drones to obtain relatively optimal solutions in multiple situations by establishing an SI algorithm library and using RL for intelligent scheduling. Simultaneously, the redundancy design of the four SI algorithms can reduce the probability of finding no solution, unlike when using one path planning algorithm, thereby significantly improving drone survivability. For DQN-SISA, a reusable path point database is established, and the SI algorithm solving and DQN model training processes are separate. Without influencing the model applicability, querying data can reduce the influence of the randomness of SI algorithms on the model convergence according to the optimal principle. The simulation results show that this technique can obtain better path point sequences than a single SI algorithm.

The DQN-SISA technology in this study still has shortcomings. The DQN model training depends on the point database of the algorithms, and the coverage of the database directly impacts the model's optimal rate. In addition to expanding the distance between the initial starting point and target point, improving the regional coverage of the node library such that DQN-SISA can obtain broader path planning solutions requires further study and discussion.

Author Contributions: Methodology, Z.M. and D.L.; software, Z.M., H.Y. and Y.Z.; original draft preparation, Z.M. and Y.Z.; writing—review and editing, D.L. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by project No. 2021ZD0140300.

Data Availability Statement: The data that support the findings of this study are available on request from the corresponding author, [Yong Zhang], upon reasonable request.

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

References

1. Mavrovouniotis, M.; Li, C.; Yang, S. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm Evol. Comput.* **2017**, *33*, 1–17. [CrossRef]
2. Tang, J.; Liu, G.; Pan, Q. A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1627–1643. [CrossRef]
3. Jain, M.; Singh, V.; Rani, A. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm Evol. Comput.* **2019**, *44*, 148–175. [CrossRef]
4. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control Eng.* **2020**, *8*, 22–34. [CrossRef]
5. Kaur, G.; Arora, S. Chaotic whale optimization algorithm. *J. Comput. Des. Eng.* **2018**, *5*, 275–284. [CrossRef]
6. Coelho, L.D.S. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Syst. Appl.* **2010**, *37*, 1676–1683. [CrossRef]
7. Decerle, J.; Grunder, O.; El Hassani, A.H.; Barakat, O. A hybrid memetic-ant colony optimization algorithm for the home health care problem with time window, synchronization and working time balancing. *Swarm Evol. Comput.* **2019**, *46*, 171–183. [CrossRef]
8. Liu, J.; Shi, J.; Hao, F.; Dai, M. A novel enhanced global exploration whale optimization algorithm based on Lévy flights and judgment mechanism for global continuous optimization problems. *Eng. Comput.* **2022**, *39*, 2433–2461. [CrossRef]
9. Heidari, A.A.; Pahlavani, P. An Efficient Modified Grey Wolf Optimizer with Lévy Flight for Optimization Tasks. *Appl. Soft Comput.* **2017**, *60*, 115–134.
10. Stephan, P.; Stephan, T.; Kannan, R.; Abraham, A. A hybrid artificial bee colony with whale optimization algorithm for improved breast cancer diagnosis. *Neural Comput. Appl.* **2021**, *33*, 13667–13691. [CrossRef]
11. Hu, J.; Wang, M.; Zhao, C.; Pan, Q.; Du, C. Formation control and collision avoidance for multi-UAV systems based on Voronoi partition. *Sci. China Technol. Sci.* **2020**, *63*, 65–72. [CrossRef]
12. Li, G.; Zhou, S.; Wu, Q. An Improved RRT Algorithm for UAV Path Planning. *Acta Electron. Sin.* **2017**, *45*, 1764–1769.
13. Moon, S.; Oh, E.; Shim, D.H. An Integral Framework of Task Assignment and Path Planning for Multiple Unmanned Aerial Vehicles in Dynamic Environments. *J. Intell. Robot. Syst.* **2013**, *70*, 303–313. [CrossRef]
14. Bhagat, S.K.; Saikia, L.C.; Raju, D.K.; Babu, N.R.; Ramoji, S.K.; Dekaraja, B.; Behra, M.K. Maiden Application of Hybrid Particle Swarm Optimization with Genetic Algorithm in AGC Studies Considering Optimized TIDN Controller. In *Modeling, Simulation and Optimization: Proceedings of CoMSO 2020*; Springer: Singapore, 2021; pp. 335–346.
15. Teng, Z.; Lv, J.; Guo, L. An improved hybrid grey wolf optimization algorithm. *Soft Comput.* **2019**, *23*, 6617–6631. [CrossRef]
16. Siddavaatam, P.; Sedaghat, R. Grey Wolf Optimizer Driven design space exploration: A novel framework for multi-objective trade-off in architectural synthesis. *Swarm Evol. Comput.* **2019**, *49*, 44–61.
17. Deng, L.; Jiang, J.; Fei, M.R. PID Parameters Tuning and Adaptation Based on Immunity Particle Swarm Optimization. *Process Autom. Instrum.* **2013**, *34*, 65–67, 71.
18. Jain, G.; Yadav, G.; Prakash, D.; Shukla, A.; Tiwari, R. MVO-based path planning scheme with coordination of UAVs in 3-D environment. *J. Comput. Sci.* **2019**, *37*, 101016. [CrossRef]
19. 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.
20. Gaidhane, P.J.; Nigam, M.J. A hybrid grey wolf optimizer and artificial bee colony algorithm for enhancing the performance of complex systems. *J. Comput. Sci.* **2018**, *27*, 284–302.
21. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82.
22. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
23. Botvinick, M.; Ritter, S.; Wang, J.X.; Kurth-Nelson, Z.; Blundell, C.; Hassabis, D. Reinforcement learning, fast and slow. *Trends Cogn. Sci.* **2019**, *23*, 408–422. [CrossRef] [PubMed]
24. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
25. Hasselt, H.V.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
26. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
27. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
28. Zhou, Q. A novel movies recommendation algorithm based on reinforcement learning with DDPG policy. *Int. J. Intell. Comput. Cybern.* **2020**, *13*, 67–79.
29. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust Region Policy Optimization. *Comput. Sci. February* **2015**, 1889–1897.
30. Proctor, P.; Teuscher, C.; Hecht, A.; Osiński, M. Proximal Policy Optimization for Radiation Source Search. *J. Nucl. Eng.* **2021**, *2*, 368–397. [CrossRef]
31. Nguyen, N.M.; Tran, M.-N.; Chandra, R. Sequential reversible jump MCMC for dynamic Bayesian neural networks. *Neurocomputing* **2024**, *564*, 126960.

32. Fernández, J.; Chiachío, J.; Barros, J.; Chiachío, M.; Kulkarni, C.S. Physics-guided recurrent neural network trained with approximate Bayesian computation: A case study on structural response prognostics. *Reliab. Eng. Syst. Saf.* **2024**, *243*, 109822. [CrossRef]
33. Colorni, A.; Dorigo, M.; Maniezzo, V. Distributed optimization by ant colonies. In Proceedings of the First European Conference on Artificial Life, Paris, France, 11–13 December 1991; Volume 142, pp. 134–142.
34. Kennedy, J.; Eberhart, R.C. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; IEEE: New York, NY, USA, 1995; pp. 1942–1948.
35. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61.
36. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
37. Cheng, Z.; Tang, Y.X.; Liu, Y.L. 3-D path planning for uav based on chaos particle swarm optimization. *Appl. Mech. Mater.* **2012**, *232*, 625–663. [CrossRef]