Theory and Practice of Science and Technology
2025, VOL. 6, NO. 4, 89-97
DOI: 10.47297/taposatWSP2633-456915.20250604

# Research on UAV Path Planning Evolutionary Algorithm Based on Reinforcement Learning

Xu Chen

Foshan University, Foshan, Guangdong China 528000

**ABSTRACT**
This paper presents an investigation into the integration of optimization algorithms with learning algorithms to enhance the performance of unmanned aerial vehicles (UAVs) in path planning. Specifically, an evolutionary algorithm is employed in conjunction with the reinforcement learning DQN model to improve its efficacy in UAV path planning. The evolutionary algorithm component focuses on population initialization methods and comparative analysis of several GWO algorithms. Subsequently, the DQN model training incorporates guidance from the evolutionary algorithm, demonstrating superior performance in simulated environments compared to the original training algorithm. Finally, a discussion and analysis of optimization algorithms and reinforcement learning algorithms in the path planning domain are provided, along with several research recommendations for future DQN model training.

**KEYWORDS**
UAV; path planning, reinforcement learning; evolutionary algorithm

## 1 Introduce

As a highly intelligent device, unmanned aerial vehicles (UAVs) exhibit significant flexibility and promising development prospects. In recent years, the application of UAVs in civilian domains has garnered increasing attention. Within the realm of UAV applications, numerous technical challenges exist, with path planning undoubtedly being a core concern. The path planning problem necessitates determining the optimal flight trajectory for a UAV to accomplish its mission in the shortest timeframe or with minimal energy expenditure. This establishes UAV path planning as a complex optimization problem, characterized by domain discontinuity, multiple constraints, and pronounced non-convexity, thereby complicating the identification of local optima. Consequently, this problem necessitates the application of sophisticated optimization algorithms, typically heuristic or evolutionary algorithms, to navigate the intricate, non-convex solution space in search of optimal or near-optimal solutions.

In recent years, evolutionary algorithms have been widely applied to path planning problems. These algorithms, which model natural phenomena computationally, exhibit robust engineering adaptability and practical applicability. For instance, Lee et al. employed an improved genetic algorithm (GA-RPP) for robot path planning [1]. Furthermore, algorithms such as simulated annealing, bat algorithm, and grey wolf optimization have also been successfully implemented in path planning applications [2-4].

While evolutionary algorithm-based path planning solutions demonstrate commendable performance and significant potential, their application in unknown environments is often unpredictable or computationally intensive, thereby failing to meet real-time requirements[5]. Evolutionary algorithms, as global path planning algorithms, employ a static framework, pre-planning UAV trajectories based on known flight conditions and mission waypoints[7]. Currently, deep reinforcement learning (DRL) algorithms are frequently utilized in path planning, leveraging their model-free nature and robust self-learning capabilities[6]. Through iterative interaction with the environment, these models autonomously address path planning challenges, enhancing the autonomy and intelligence of UAVs in this domain. Numerous studies have integrated DRL algorithms into UAV path planning. He Lei et al. explored the interpretability of autonomous UAV path planning using DRL[12]; Lan Bo et al. combined low-rank adaptation with reinforcement learning, achieving favorable training outcomes[13]. However, most studies acknowledge that DRL algorithms necessitate extensive sample learning and prolonged training durations. This parallels the initial, undirected learning phase in humans, necessitating guidance to optimize training. Zhang Ye et al.'s review of DRL for path planning also highlights eight fundamental enhancement methods to improve DRL algorithm integration[14].

Imitation learning, a significant subfield within reinforcement learning, leverages supplementary information, specifically expert demonstrations, to expedite policy optimization, thereby offering a viable solution to the sample inefficiency inherent in reinforcement learning [8]. Behavioral cloning (BC), a concept introduced by Bain and Sammut in 1999, constitutes a core methodology. BC trains an agent by assigning actions from expert demonstration data as labels for each state, effectively framing the process as a classification or regression task. In this domain, Ross and Bagnell's AggreVaTe (aggregate values to imitate) algorithm focuses on the future expected cost of policy selection, rather than solely on the discrepancy between the agent's policy and the expert policy [9].

This study investigates the feasibility of integrating evolutionary algorithms and reinforcement learning for UAV path planning to enhance the accuracy and intelligence of path selection in complex environments. We propose a path planning methodology that leverages behavioral cloning to bridge reinforcement learning and evolutionary algorithms. This approach aims to accommodate multi-objective, multi-constraint flight missions within dynamic environments, thereby optimizing path planning for reduced execution time and increased flexibility.

## 2 Problem formulation

In the context of UAV path planning, the objective is to identify a flight trajectory that minimizes distance, threat exposure, and energy consumption. This problem can be formulated as a Markov Decision Process (MDP), a stochastic control process designed to model and solve decision-making problems under uncertainty, where sequential decisions influence outcomes[16]. This study simulates UAV flight paths within a modeled urban environment, incorporating randomly generated obstacles represented as cuboids. The UAV's positional data is defined using three-dimensional coordinates. Furthermore, stochastic wind conditions are simulated within the urban environment to emulate realistic flight scenarios.

### 2.1 State-action space representation

To facilitate enhanced environmental exploration and accelerate training iteration, the state space is defined as follows:

$$State = \{Grid, Target, Infouav\} \tag{1}$$

The grid represents the surrounding environmental information of the UAV, the target denotes the target point's positional information, and $Info_{uav}$ represents the UAV's state information, encompassing the UAV's position, energy consumption, crash probability, and distance to the target. Regarding the UAV's action space design, this study defines the UAV's actions within a 27-dimensional action space, with three actions—backward, forward, and maintain—available in each of the three-dimensional space (x, y, z), represented by the numbers 0-26.

### 2.2 Reward function design

The reward function serves to evaluate the value of an action within a given state for a DQN. In this study, successful arrival at the target location is designated as the terminal action for the UAV. Prior to termination, the UAV continuously explores the environment.

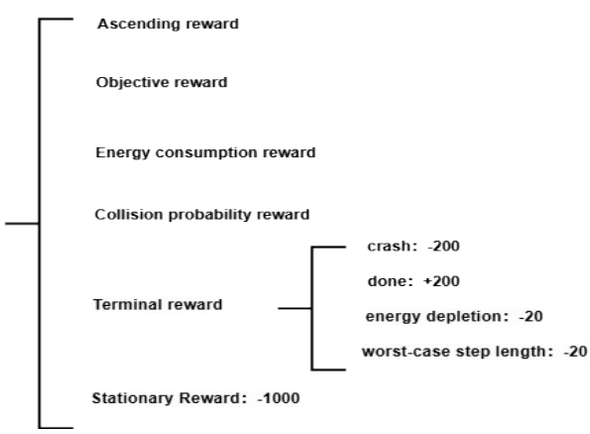


Figure 1 Reward function

## 3 Methodology

In behavioral cloning, an agent is trained using an expert dataset to replicate the expert's policy via a classifier or regressor. This study frames the path planning training of a DQN agent as imitation learning from expert demonstration data, where paths generated by evolutionary algorithms serve as the training data. Metaheuristic algorithms are often suitable for global optimization problems[10], and evolutionary algorithms are particularly effective for path planning in static environments, demonstrating robust performance. During the agent's training with expert demonstration data, environmental exploration is incorporated, and the resulting experiences are stored in an experience replay buffer, with parameters updated through periodic random sampling. The agent learning model constructed in this study is illustrated

in the figure.

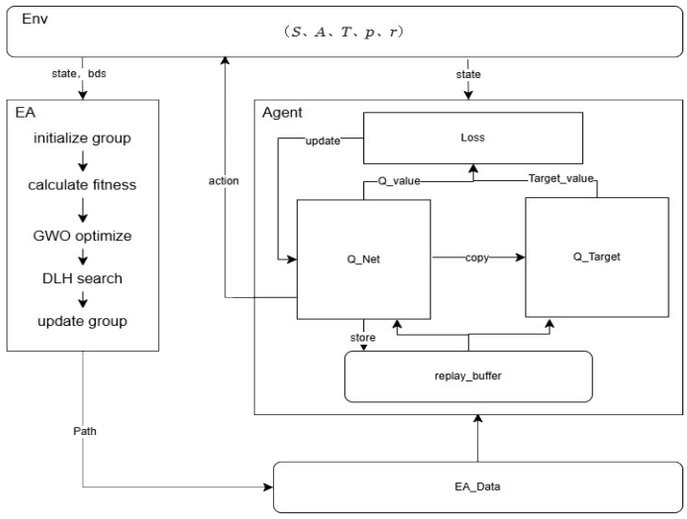


Figure 2 Learn the model framework

### 3.1 Evolutionary algorithms

In the domain of UAV path planning, prevalent algorithmic approaches encompass traditional methods, bio-inspired algorithms, and deep learning techniques. This study employs a bio-inspired algorithm for path planning, wherein the agent assumes an expert role within the constructed learning model. Evolutionary algorithms exhibit robust global search capabilities and extensive exploration of the solution space, enabling rapid identification of high-quality solutions and accelerating the convergence of agent training.

Evolutionary algorithms (EA) iteratively refine a population of candidate solutions through processes of crossover, mutation, and selection, ultimately converging towards an optimal solution for a given problem. The fitness function, specific to the problem domain, quantifies the quality of each solution throughout the iterative process. Key elements in EA implementation include the representation of population members, selection strategies, crossover and mutation operators, and fitness evaluation[10]. In this study, a complete path from the starting point to the destination represents an individual within the population, thus constituting a potential solution to the path planning problem. The genes of an individual are defined by a sequence of waypoints along the path. Crossover and mutation operations are implemented by manipulating these waypoints, thereby facilitating the exploration of the solution space. The representation of solutions for the path planning problem is defined as follows:

$$X=\left[Start,point1,point2,\cdots\cdots,pointN,End\right] \tag{2}$$

In this study, the Logistic map method is employed for population initialization, facilitating a broader distribution across the search space and thereby enhancing the algorithm's exploratory capabilities [11]. Furthermore, the initialization of the path points across three dimensions involves an initial proportional division based on the variation in the starting point's x-coordinate to determine the x-coordinates of N path points. The calculation of the x-coordinate is defined by the following equation:

$$Xi,N=\left[Xstart,Xstart+(i+1)*\frac{Xend-Xstart}{N},\cdots\cdots,Xend\right] \tag{3}$$

Subsequently, random initialization is applied to the y and z coordinates, thereby reducing the dimensionality of the problem and enhancing algorithmic efficiency. This approach facilitates directional guidance for the initially generated paths, directing path points towards the terminal point, which significantly improves the quality of the initial solution. Finally, initialization range constraints are established based on the coordinates of the starting point and environmental obstacle information, thereby accelerating the convergence of the algorithm.

$$\mathrm{Ymin}=\min\left\{\mathrm{BD}_y-\mathrm{w}\right\}-\mathrm{d}\left(\mathrm{Ymin}\geq 0\right) \tag{4}$$

$$Ymax=max\left\{BD_y+w\right\}+d\left(Ymax\geq 0\right) \tag{5}$$

The path planning algorithm employed in this research is based on the Grey Wolf Optimizer (GWO), a metaheuristic algorithm introduced by Mirjalili et al. in 2014 [15]. GWO mimics the social hierarchy and hunting behavior of grey wolves in nature. The positions of individual wolves within the pack are updated based on the positions of the α, β, and δ wolves to converge towards superior hunting locations. In the context of UAV path planning, the optimal solution is defined as the hunting target, with the three wolves closest to the target designated as α, β, and δ. The GWO algorithm utilizes two

primary parameters, A and C, to guide the search process. When parameter A exceeds 1 or is less than -1, the wolves engage in exploration, deviating from the hunting target to identify potentially superior solutions. Parameter C, as emphasized by Mirjalili et al., is maintained as a stochastic variable to ensure exploration throughout the algorithm's execution, particularly when the population stagnates during iterations.

It is noteworthy that the GWO webpage (GWO) references an enhanced algorithm, I-GWO, predicated on the Grey Wolf Optimizer. This algorithm, developed by Mohammad H. Nadimi-Shahraki et al., is an engineering optimization technique with limited application in UAV path planning. I-GWO incorporates a dimension learning-based hunting strategy (DLH), which augments the algorithm's exploitation capabilities, thereby outperforming GWO in complex environments.

While both GWO and I-GWO studies acknowledge the hierarchical structure of the wolf pack, comprising alpha, beta, and delta wolves, their roles in guiding the population within practical algorithmic implementations are often treated as equivalent. This research builds upon this by implementing differential guidance ratios based on fitness values. The alpha wolf is assigned the highest proportion, while the delta wolf receives the lowest. Furthermore, to enhance computational efficiency, a comparison is conducted with the delta wolf during each movement phase; superior solutions are adopted, thereby enabling more timely guidance of the population by superior solutions.

### 3.2 Deep reinforcement learning algorithms

In a reinforcement learning (RL) system, five primary components are typically identified: the agent, the environment, the policy, the reward function, and the value function. The objective of RL algorithms is to maximize cumulative reward, with the value function serving to evaluate the long-term value of state-action pairs. This aligns with the objectives of Markov Decision Processes (MDPs), positioning RL as a leading approach for solving MDP problems in numerous studies.

However, reinforcement learning necessitates substantial memory allocation for reward storage. For instance, the prevalent Q-learning algorithm employs look-up tables to map states and actions to values, resulting in significant data volume and limitations in continuous state-action spaces. In the deep learning model (DQN) proposed by Volodymyr Mnih et al., deep convolutional neural networks are utilized to approximate the Q-value function, mirroring human processing of high-dimensional sensory inputs—a fusion of reinforcement learning and hierarchical processing systems. The DQN network can process high-dimensional sensory inputs and actions, thereby learning concepts such as object categories [17].

The instability observed in reinforcement learning algorithms when addressing non-linear problems is primarily attributed to correlations within the observation sequence and between action values and target values. To mitigate these issues, the DQN network employs two key strategies: experience replay and periodic target value updates. These methods are designed to disrupt correlations, thereby enhancing the model's stability in handling non-linear problems.

### 3.3 DQN network incorporating imitation learning

The objective of imitation learning is to acquire expert strategies through observation of expert demonstrations. Inspired by the imitative capabilities observed in humans and other animals, such as infants learning tasks by mimicking parental behavior, this class of algorithms is developed. Traditional imitation learning algorithms are primarily categorized into three types: behavioral cloning, inverse reinforcement learning, and adversarial imitation learning. In this study, the concept of behavioral cloning is integrated into the training of the DQN network, with the aim of accelerating algorithm convergence, enhancing the model's ultimate performance, and mitigating the issue of low sample efficiency in training.

In behavioral cloning, expert data can be represented as <state, action>, where the action is a label assigned to each state. The model replicates the expert policy by learning from this data. We denote the expert action as ai, the environmental state as si, and the agent policy as Π. Behavioral cloning can then be formulated as:

$$\min \sum_{i=1}^{N} \left( a_i - \Pi\left( s_i \right) \right)^2 \tag{6}$$

In this study, the paths generated by the evolutionary algorithm were employed as the expert dataset. Experimental results demonstrated that the generated paths exhibited favorable characteristics in terms of length, obstacle avoidance, and curvature. Consequently, these paths are deemed suitable for knowledge demonstration as expert demonstrations. Furthermore, this approach streamlines the acquisition of expert datasets and reduces training costs.

The algorithmic process of this study integrates the concept of aggregated data, as proposed by Ross and Bagnell. Initially, the agent's policy is initialized for each training episode. A decaying exponential function, converging towards a minimal value, determines the probability $\epsilon$, which governs the utilization of expert policy during training. The probability $\epsilon$ is mathematically represented as:

$$\epsilon = \left(1 - \alpha\right)^{(iter - 1)} \tag{7}$$

Here, 'iter' denotes the current iteration count, and 'alpha' represents a hyperparameter influencing the decay rate of

During each training episode, a sequence of 'Step' actions is executed. A random variable 't', uniformly distributed over the interval (0, Step), is defined. Prior to time step 't', actions are selected according to the agent's policy. Subsequently, for the remaining steps, the expert policy is employed, and the resulting expert data is recorded. Expert data is collected in each iteration and aggregated into a comprehensive expert dataset, 'Data'. At the conclusion of each training iteration, 'Data' is utilized to train and select the optimal policy. This methodology facilitates the incorporation of historical training information, thereby enhancing the model's stability.

Furthermore, this study incorporated capacity constraints within the expert dataset, Data, to prevent overfitting to early, redundant data, thereby enhancing training efficiency and reducing computational costs. To maintain decorrelation of the observation sequences within the DQN network, we employed random sampling from the expert dataset, which, in turn, improved the utilization of expert data. Finally, modifications were made to the loss function calculation for model updates, which can be represented as:

$$\mathrm{BC\_Loss} = \left(a - \Pi(s)\right)^2 \tag{8}$$

$$Loss = \mathrm{E}(s,a,r,s') \sim \mathrm{U}(D)\left[\left(r + \gamma \max a' Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\right)^2\right] + w*BC_{Loss} \tag{9}$$

We incorporated a behavioral cloning loss term into the loss calculation of the DQN network. Here, 'w' denotes the weight assigned to the BC loss, which ensures that the model replicates the expert policy while simultaneously engaging in autonomous exploration of the environment.

## 4 Simulation experiment

The simulation experiments were conducted utilizing a model architecture implemented within the PyTorch framework. The programming language employed was Python 3.8, and the experiments were executed on an Nvidia RTX 4060 GPU.

### 4.1 Experimental setup

The experimental setup for this study was conducted within a simulated urban environment, defined by a (100, 100, 22) dimensional space. The environment incorporated randomly generated cuboid obstacles. During the training phase, the agent was exposed to environments of level 8 for 5000 cycles. Level 8 environments were characterized by the stochastic generation of 8 to 12 obstacles. A representative example of the simulated environment is illustrated in the accompanying figure.

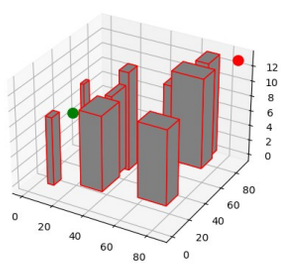


Figure 3 level8

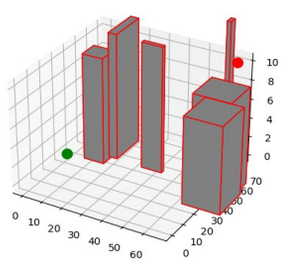


Figure 4 level5

Prior to the implementation of DQN experiments, this study initiated comparative analyses of diverse path planning algorithms within a static environment. Fifty iterations were conducted for each algorithm within a simulation environment of level 8 complexity. The performance of each algorithm was evaluated by comparing the mean fitness values, standard deviations, worst fitness values, and best fitness values across the populations.

During the training phase, the agent underwent 5,000 training cycles, with each cycle comprising 100 timesteps, representing the number of actions taken by the drone. A terminal state was defined upon the drone reaching the target within these 100 steps, concluding the episode. The DQN network parameters were consistent across both training algorithms. This study involved separate training of the DQN network and the DQN network guided by an evolutionary algorithm. We recorded and compared the loss curves and the changes in average scores during training to analyze the convergence and overall performance of the algorithms.

The hyperparameters utilized in the experiments conducted for this study are detailed as follows:

Parameters used

Table1 Parameters used in this article

| Parameters | Value |
|---|---|
| Learning rate | 0.0004 |
| BC_weight | 0.5 |
| Batch size | 128 |
| Gamma | 0.99 |
| Replay memory size | 10000 |
| EA replay memory size | 10000 |
| Population size | 50 |

## 4.2 Experimental results and analysis

### 4.2.1 Comparative experiments on path planning algorithms

In this experiment, obstacles were positioned within a defined area of (100, 100, 22) units. Comparative analyses were conducted utilizing the Grey Wolf Optimizer (GWO), Improved Grey Wolf Optimizer (IGWO), and a novel optimization algorithm incorporating weighted parameters. All simulations were executed on a single computational platform. The results of the path optimization algorithms, under a simulated environment with eleven obstacles, are presented graphically. The results indicate that all three algorithms successfully generated feasible flight paths. However, both the GWO and IGWO algorithms converged to local optima, whereas the proposed algorithm demonstrated superior performance, yielding the shortest path. Specifically, the objective function values were 71.04 for GWO, 67.58 for IGWO, and 38.78 for the proposed algorithm.

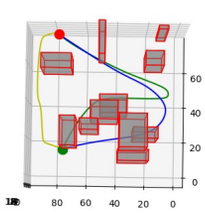


Figure 5 Top-down view of the path planning results

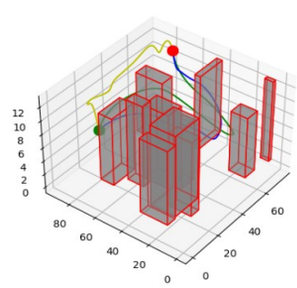


Figure 6 A frontal view of the path planning outcomes

Table 2 Path Comparison Experiment

| Arithmetic | BEST | WORST | AVE | STD |
|---|---|---|---|---|
| GWO | 71.04 | 91.17 | 87.14 | 6.29 |
| IGWO | 67.58 | 112.66 | 92.28 | 3.26 |
| Used in this article | 38.78 | 109.86 | 72.21 | 4.29 |

### 4.2.2 Comparative experiments on deep reinforcement learning algorithms

The results of two training methodologies for the DQN network are presented graphically. Initially, the DQN training incorporating an evolutionary algorithm did not demonstrate a performance advantage. However, its learning curve exhibited a smoother, more consistent ascent, ultimately achieving an average score exceeding 5000 points in the later stages of training. Conversely, the DQN training without the evolutionary algorithm displayed a more volatile score trajectory, culminating in an average score of approximately 2000 points at the conclusion of training.

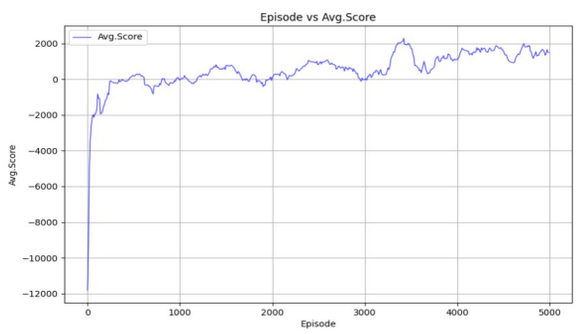


Figure 7 the average scores during DQN training

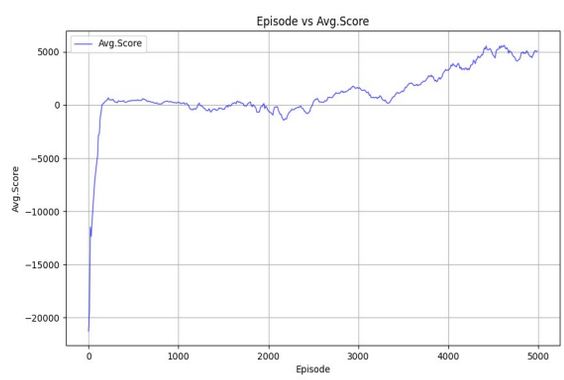


Figure 8 the average scores during DQN training incorporating an evolutionary algorithm

The loss curves for the two training methodologies are presented below. A comparative analysis of the two plots reveals that the DQN training, incorporating evolutionary algorithms, initially exhibits higher loss values, yet demonstrates a smoother decline. Both methodologies converge within 20 epochs, with minimal disparity in the convergence values during the later stages of algorithm training. This observation suggests that the current methodological approach may not fully facilitate the DQN model's comprehension of the underlying patterns within the data, primarily accelerating the DQN training process.

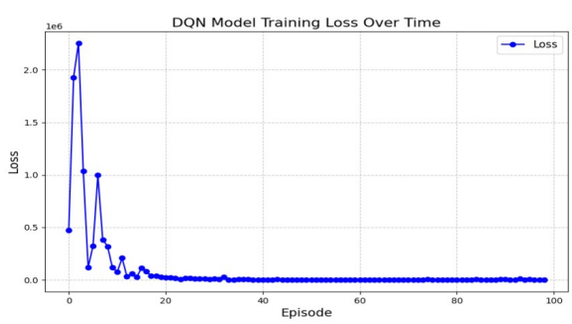


Figure 9 the loss curve during DQN training

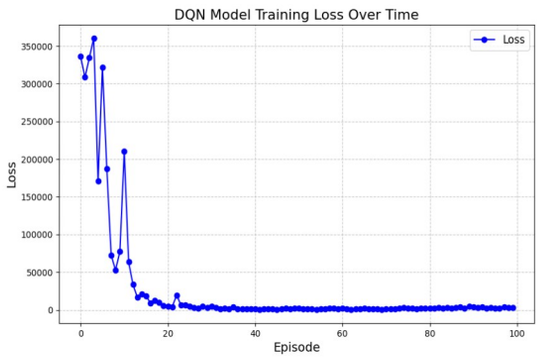


Figure 10 the loss curve during DQN training incorporating an evolutionary algorithm

In this investigation, we employed paths generated by an evolutionary algorithm to guide the training of the DQN. The average, maximum, and minimum return values associated with the paths produced by the evolutionary algorithm are presented in the following table.

Table 3 the values of the paths

| Avg Reward | Best Reward | Worst Reward |
|---|---|---|
| 2182.04 | 5859.52 | 48.07 |

## 5 Conclusion and Future Outlook

According to the No Free Lunch (NFL) theorem, as proposed by David Wolpert and William Macready, no single optimization algorithm universally outperforms all others across all problem domains, and all algorithms exhibit a dependence on prior knowledge to achieve superior performance on specific problems. This study presents an initial integration of evolutionary algorithms and deep reinforcement learning within the context of unmanned aerial vehicle (UAV) path planning. The experimental results indicate that the incorporation of paths generated by evolutionary algorithms enhances the performance of Deep Q-Networks (DQN) training. However, an analysis of the DQN training loss and the reward values associated with the paths generated by the evolutionary algorithm reveals that further investigation is warranted to improve the DQN's ability to discern data patterns and to optimize the evolutionary algorithm's search for optimal solutions. Ultimately, this research suggests that incorporating expert demonstrations to guide the training of DQN models can improve the final path planning performance. Future work will explore the integration of alternative optimization algorithms and refinements to the DQN model's learning framework.

## References

[1] Lee, J.; Kang, B.Y.; Kim, D.W. (2013) Fast genetic algorithm for robot path planning. Electron. Lett. Commun.,49:1449–1451.

[2] Guo, J.; Gao, Y.; Cui, G. (2015) The path planning for mobile robot based on bat algorithm. Int. J. Autom. Control. Commun., 9: 50–60.

[3] Tsai, P.W.; Dao, T.K.; others. (2016) Robot path planning optimization based on multiobjective grey wolf optimizer. In: International Conference on Genetic and Evolutionary Computing. Berlin/Heidelberg, Germany. pp. 166–173.

[4] Turker T, Yilmaz G, Sahingoz O K. (2016) GPU-Accelerated Flight Route Planning for Multi-UAV Systems Using Simulated Annealing [J]. Artificial Intelligence Method ology Systems Applications. pp. 279–288.

[5] Xing Lijing, Li Min, Zeng Xiangguang, Zhang Ping, Peng Bei. (2024) AUV path planning based on behavior cloning and improved DQN in partially unknown environments[J]. Journal of System Simulation.1-13. https://doi.org/10.16182/j.issn1004731x.joss.24-0678.

[6] Ding Jie, Wang Di. (2025) Urban low-altitude UAV path planning based on reinforcement learning[J/OL]. Journal of Transportation Engineering and Information.Commun.,1-22.

[7] Xiao Heng, Zhang Hui, Zhou Wen, et al. (2024) Research on UAV path planning algorithm based on improved virtual spring model[J/OL]. Progress in Aeronautical Engineering.Commun.,1-8.

[8] Zhang Chao, Bai Wensong, Du Xin, Liu Weijie, Zhou Chenhao, & Qian Hui. (2023). A survey on imitation learning: Traditional and new developments. Journal of Image and Graphics. Commun., 28(06), 1585-1607.

[9] Ross, S., & Bagnell, J. A. (2014). Reinforcement and imitation learning via interactive no-regret learning. arXiv preprint.

[10] Agrawal, S., Patle, B. K., & Sanap, S. (2024). A systematic review on metaheuristic approaches for autonomous path planning of unmanned aerial vehicles. Drone Systems and Applications. Commun.,12, 1-28.

[11] Jia, C., He, L.*, Liu, D., & Fu, S. (2024). Path planning and engineering problems of 3D UAV based on adaptive coati optimization algorithm. Scientific Reports. Commun., 14(1), 30717.

[12] He, L., Aouf, N., & Song, B. (2021). Explainable deep reinforcement learning for UAV autonomous path planning. Aerospace Science and

Technology. Commun.,118, 107052.
[13] Bo, L., Zhang, T., Zhang, H., Hong, J., Liu, M., Zhang, C., & Liu, B. (2024). 3D UAV path planning in unknown environment: A transfer reinforcement learning method based on low-rank adaptation. Advanced Engineering Informatics. Commun.,62(D), 102920.
[14] Zhang, Y., Zhao, W., Wang, J. Y., & Yuan, Y. (2024). Recent progress, challenges and future prospects of applied deep reinforcement learning: A practical perspective in path planning. Neurocomputing. Commun., 608, 128423.
[15] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. Advances in Engineering Software. Commun., 69: 46-61.
[16] Adawadkar, A. M. K., & Kulkarni, N. (2022). Cyber-security and reinforcement learning—A brief survey. Engineering Applications of Artificial Intelligence. Commun., 114, 105116.
[17] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015) Human-level control through deep reinforcement learning. Nature. Commun., 518: 529-533.