aerospace

MDPI

Review

Review of Deep Reinforcement Learning Approaches for Conflict Resolution in Air Traffic Control

Zhuang Wang [1,*], Weijun Pan [1], Hui Li [2], Xuan Wang [1] and Qinghai Zuo [1]

[1] College of Air Traffic Management, Civil Aviation Flight University of China, Guanghan 618307, China; wjpan@cafuc.edu.cn (W.P.); atcxuan@cafuc.edu.cn (X.W.); zuoqh@cafuc.edu.cn (Q.Z.)
[2] College of Computer Science, Sichuan University, Chengdu 610065, China; lihuib@scu.edu.cn
* Correspondence: wangzhuang@cafuc.edu.cn

Abstract: Deep reinforcement learning (DRL) has been widely adopted recently for its ability to solve decision-making problems that were previously out of reach due to a combination of nonlinear and high dimensionality. In the last few years, it has spread in the field of air traffic control (ATC), particularly in conflict resolution. In this work, we conduct a detailed review of existing DRL applications for conflict resolution problems. This survey offered a comprehensive review based on segments as (1) fundamentals of conflict resolution, (2) development of DRL, and (3) various applications of DRL in conflict resolution classified according to environment, model, algorithm, and evaluating indicator. Finally, an open discussion is provided that potentially raises a range of future research directions in conflict resolution using DRL. The objective of this review is to present a guidance point for future research in a more meaningful direction.

Keywords: air traffic control; conflict resolution; deep reinforcement learning

Citation: Wang, Z.; Pan, W.; Li, H.; Wang, X.; Zuo, Q. Review of Deep Reinforcement Learning Approaches for Conflict Resolution in Air Traffic Control. *Aerospace* **2022**, *9*, 294. https://doi.org/10.3390/aerospace9060294

Academic Editor: Joost Ellerbroek

Received: 28 April 2022
Accepted: 25 May 2022
Published: 28 May 2022

1. Introduction

In the past few years, with the rapid development of civil aviation, air traffic flow continues to increase. According to Federal Aviation Administration (FAA) forecast, from 2020 to 2040, the number of aircraft handled by en-route centers will be increased at an average rate of 1.5% per year [1]. As a result, the workload of air traffic controllers (ATCos) is constantly increasing. Conflict resolution automation can reduce the amount of time to resolve potential conflicts, thus reducing the workload of ATCos. There are four common methods of conflict resolution: the geometric algorithm, the optimal control theory, the traditional intelligence approach, and the deep reinforcement learning (DRL) approach. Although many high-performing approaches have emerged in recent years, a review [2] published in 2000 is still often cited as an overview of conflict resolution. Jenie proposed a taxonomy of conflict resolution approaches for operating unmanned aerial vehicles (UAVs) in integrated airspace [3]. Ribeiro considered more than 100 conflict resolution methods and provided a more comprehensive analysis in 2020 [4]. However, as an advanced approach, DRL has not been analyzed by these reviews. This review aims to present an overview of DRL approaches for conflict resolution, which may be considered as a supplement to current reviews.

DRL is a type of artificial intelligence, which combines reinforcement learning (RL) [5] and deep learning (DL) [6]. It allows agents to learn directly from the environment through trial and error without a perfect knowledge of the environment in advance. It has the advantages of high decision-making efficiency and independent of model or data. An agent trained by DRL can automatically determine an adequate behavior within a specific context trying to maximize its performance using few computational times. DRL has been utilized in many fields and obtained great achievements of human-level or superhuman performance [7–9]. The theory of DRL is very suitable for solving sequential decision-making problems such as conflict resolution in air traffic control (ATC).

The existing research on using DRL to solve conflict resolution problems has established a variety of models and adopted different algorithms. The applied scenarios include both en-route and free flight scenarios. Mainstream DRL algorithms, such as Deep Q-Network (DQN) [10], Deep Deterministic Policy Gradient (DDPG) [11] and Proximal Policy Optimization (PPO) [12], have been adopted, and multi-agent algorithms have also been used. In this review, we classify methods according to the following four characteristics: environment, model, algorithm, and evaluating indicator. The model is analyzed from three aspects: state space, action space, and reward function. Each DRL algorithm is analyzed in detail, and its suitable application is described. We also listed some open issues and put forward future research suggestions. The objective of this article is to make a detailed analysis of the current research on using DRL to solve conflict resolution problems, hoping to inspire readers and promote the further application of DRL in this field.

This review is organized as follows. Section 2 introduces the problem of conflict resolution in air traffic control, as well as the deep reinforcement learning algorithm involved in the approaches summarized in this review. Section 3 describes the key contents of each approach, including environment, model, algorithm and evaluating indicator. Section 4 discusses problems and challenges in current approaches, and put forward future directions. Section 5 concludes the review.

2. Background

In this section, the problem of conflict resolution is introduced. Deep reinforcement learning and the algorithms used in the research involved in this review are also presented.

2.1. Conflict Resolution Process

A conflict occurs when two or more aircraft lose the minimum separation, of which the horizontal separation is 5 NM and the vertical separation is 1000 ft [13]. In this way, a cylindrical aircraft safety area is formed, as shown in Figure 1. A safe distance must be maintained between aircraft during flight. When the horizontal distance and vertical distance are less than the minimum separation at the same time, it is determined as a conflict.

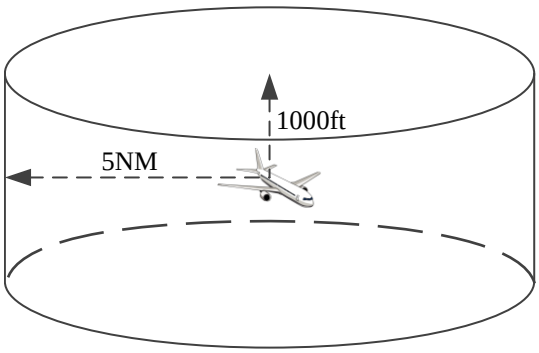
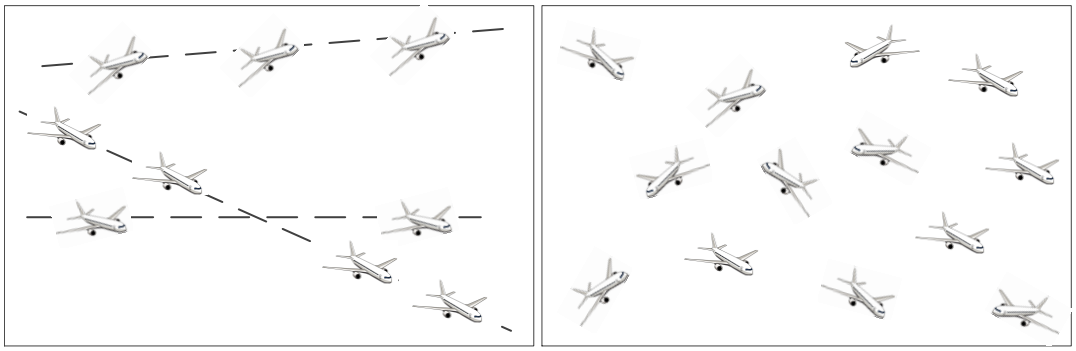


Figure 1. Safety cylinder around aircraft.

The flight of aircraft in airspace includes two scenarios: en-route and free flight, as shown in Figure 2. The en-route system consists of a mixture of direct and organized tracks, and fixed airways, as shown in Figure 2a. The continuous growth of air traffic demand has brought serious traffic congestion, which can be effectively solved by free flight. Free flight means that the aircraft can freely plan its route between the defined entry point and exit point without referring to the fixed route, as shown in Figure 2b. The DRL approaches have been used in both scenarios to solve the conflict resolution problem.



(**a**) En-route scenario (**b**) Free flight scenario

Figure 2. Aircraft flight scenario.

ATCos usually adopt three kinds of resolution methods: altitude adjustment, heading adjustment, and speed adjustment.

1. Altitude adjustment. This is the most effective and frequently used method. The controller needs to abide by the flight level (FL) for altitude adjustment. Below FL 41, there is a flight level every 1000 ft, and above FL41, there is a flight level every 2000 ft.
2. Heading adjustment. This method can be used for conflict resolution intuitively, but it will change the aircraft route. There are two approaches for heading adjustment. One is heading angle change, which is to control an aircraft to turn left or right by an angle. The other is the offset method, which controls an aircraft to fly a certain distance to the left or right to maintain the lateral separation.
3. Speed adjustment. Aircraft usually fly at the cruising altitude according to the economic speed, and the speed change cannot be represented intuitively. Therefore, speed adjustment is not recommended and should be expressed in multiples of 0.01 Mach or 10 kt.

In addition to the above basic adjustments, ATCos will also send some composite instructions, such as hovering and flying to the next waypoint. These instructions are a combination of three basic adjustments.

In the process of conflict resolution, ATCos need to adhere to high-level principles and employ different strategies. Safety is the highest priority, and a detected conflict should be resolved without bringing new conflicts. In order to reduce the workload of ATCos and pilots, it is necessary to adopt a key action to solve the problem. The resolution that requires the least amount of sector disruption and with the least amount of monitoring should be selected. The altitude adjustment is preferred, and the speed solution is considered last since the speed envelope is small at cruising altitude.

2.2. Deep Reinforcement Learning

Reinforcement learning [5] is a method that an agent continues to interact with the environment to maximize the long-term reward. The most important difference between RL and other types of learning is that it uses training information to evaluate the actions it takes, rather than using the correct actions for guidance. RL can be treated as Markov Decision Process (MDP). An MDP is defined as a tuple $<S, A, T, R, \gamma>$, where S is the set of states of the environment, A is the set of actions that the agent can use to interact with the environment, T is the transition function that defines the probability from one state to another, and R is the immediate reward function and γ is the discount factor. At each training time t, the agent receives a state S_t in a state space S, and generates an action A_t from an action space A, following a policy $\pi : S \times A \rightarrow \mathbb{R}$. Then, the agent receives a scalar reward R_t, and transitions to the next state S_{t+1} according to the environment dynamics, as shown in Figure 3. The goal of an agent is to learn a policy π which defines the action that should be used in a state to maximize the future cumulative reward.

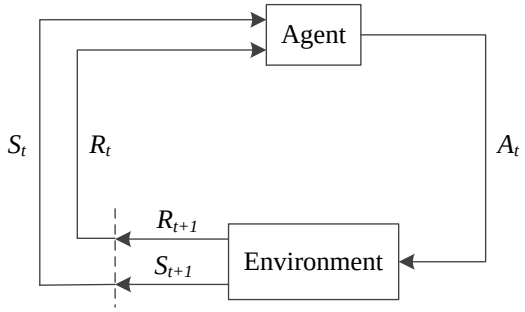


Figure 3. Reinforcement learning.

The state an agent received is a high-dimensional infinite vector or raw pixels. Therefore, the function approximate method should be used to combine features of state and learned weights. DRL is an effective method to solve this problem, including value-based algorithm and actor-critic algorithm. The algorithms used in solving conflict resolution problem include value-based algorithms such as DQN and its variants, and actor-critic algorithms including DDPG and PPO.

The action-value function $Q(s, a)$ can be used, which is defined as the value of taking action a in state s under a policy π:

$$Q(s,a) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a \right\} \tag{1}$$

The Bellman optimality equations is:

$$Q^*(s,a) = \mathbb{E} \left\{ R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a \right\} \tag{2}$$

Any policy that is greedy with respect to the optimal evaluation function $Q^*(s, a)$ is an optimal policy. Actually, $Q^*(s, a)$ can be obtained through iterations using temporal-difference learning, and its updating formula is defined as [14]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)) \tag{3}$$

The action-value function in DQN can be estimated with function approximation, $\widehat{Q}(S_t, A_t; \theta) \approx Q_\pi(S_t, A_t)$, where θ are the weights of a neural network. This leads to a sequence of loss functions $L_i(\theta_i)$ that changes at each iteration i:

$$L_i(\theta_i) = \mathbb{E}_{(S_t, A_t, R_t, S_{t+1})} \left[\left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_i^-) - Q(S_t, A_t; \theta_i) \right)^2 \right] \tag{4}$$

where θ_i is the parameter for online network and θ_i^- is the parameter for target network. The parameters are updated as follows:

$$\theta_{t+1} = \theta_t + \alpha(y_t^Q - Q(S_t, A_t; \theta_t)) \nabla_{\theta_t} Q(S_t, A_t; \theta_t) \tag{5}$$

where

$$y_t^Q = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-) \tag{6}$$

Double DQN [15] is proposed to tackle the over-estimate problem in DQN. The greedy policy is evaluated according to the online network, while its value is evaluated by the target network. This can be achieved with a minor change to the DQN algorithm, replacing y_t^Q with:

$$y_t^{D-DQN} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t^-) \tag{7}$$

Dueling DQN [16] is another variant of DQN, which combines the state value function $V(s)$ and the advantage function $A(s,a)$ to estimate action-value function $Q(s,a)$. One stream of fully-connected layers is made to output a scalar $V(s;\theta,\beta)$, and the other stream output an $|A|$-dimensional vector $A(s,a;\theta,\alpha)$.The action-value function are then given by:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \max_{a'} A(s,a';\theta,\alpha)) \tag{8}$$

The max operator can be replaced with average as below for better stability:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha)) \tag{9}$$

The dueling architecture with its separate advantage stream is robust and converge faster than DQN and double DQN.

Value-based algorithms first optimize the value function and then derive the optimal strategy, while policy-based algorithms directly optimize the objective function. An actor-critic algorithm learns both a policy and a value function. The critic uses an approximation architecture and simulation to learn a value function, which is then used to update the actor's policy parameters in a direction of performance improvement.

DDPG is an actor-critic, model-free algorithm in continuous action spaces, by extending DQN and DPG [17]. The critic network $Q(S_t, A_t; \theta^Q)$ is updated by minimizing the loss:

$$L = \frac{1}{N}\sum_t (y_t - Q(S_t, A_t; \theta^Q))^2 \tag{10}$$

where:

$$y_t = R_{t+1} + \gamma Q'(S_{t+1}, \mu'(S_{t+1}; \theta^{\mu'}); \theta^{Q'}) \tag{11}$$

The actor policy $\mu(S_t; \theta^\mu)$ is updated using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_t \nabla_{A_t} Q(S_t, A_t; \theta^Q) \nabla_{\theta^\mu} \mu(S_t; \theta^\mu) \tag{12}$$

The deterministic policy gradient is the expected gradient of the action value function and can be estimated more efficiently than the stochastic policy gradient.

PPO is proposed to benefit the stability and reliability from Trust Region Policy Optimization (TRPO) [18], with the goal of simpler implementation, better generalization, and better empirical sample complexity. Parameters for value function and policy can be shared in a neural network, and advantage function can be estimated to reduce the variance of policy parameters estimation. The objective of PPO is defined as follows, which is maximized each iteration:

$$L_t^{C+VF+P}(\theta) = \hat{\mathbb{E}}_t[L_t^C(\theta) - c_1 L_t^{VF}(\theta) + c_2 B[\pi_\theta](S_t)] \tag{13}$$

where $\hat{E}_t[...]$ indicates the empirical average over a finite batch of samples, c_1, c_2 are coefficients, $B[\pi_\theta](S_t)$ denotes an entropy bonus, and $L_t^{VF}(\theta)$ is a squared-error loss $(V_\theta(S_t) - V_t^{targ})^2$. $L_t^C(\theta)$ is the clipped surrogate objective, which is defined as:

$$L_t^C(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{a}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{a}_t)] \tag{14}$$

where ϵ is a hyperparameter, $r_t(\theta)$ denotes the probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(A_t \mid S_t)}{\pi_{\theta_{old}}(A_t \mid S_t)} \tag{15}$$

where $\pi_\theta(A_t \mid S_t)$ and $\pi_{\theta_{old}}(A_t \mid S_t)$ denote the probability to generate action A_t under situation S_t under new strategy and old strategy.

The multi-agent DRL approach is also used to solve the problem of conflict resolution. Instead of considering one agent's interaction with the environment, a set of agents that share the same environment are concerned [19,20]. A multi-agent setting is denoted with tuple $<N, S, A, T, R, O, \gamma>$, in which N is the number of agents, S is the state space, $A = \{A_1, \ldots, A_N\}$ is the set of actions for all agents, T is the transition probability, R is the reward function, $O = \{O_1, \ldots, O_N\}$ is the set of observations for all agents, and γ is the discount factor. Similar to the single-agent RL, every agent needs to learn the optimal value or policy. However, because the policy of each agent changes with the progress of training, the environment becomes non-stationary from the perspective of individual agent, which makes it difficult for an agent to converge to a good policy. Therefore, non-stationary must be solved when using multi-agent algorithm for conflict resolution.

3. Application

3.1. Overview

Since AlphaGo using DRL made great achievements in Go in 2016 [21], scholars in various fields are trying to use DRL to solve their problems. Researches on conflict resolution using DRL have been published continuously since 2018. The objective of using DRL is to generate conflict resolution decisions with safety and efficiency, to reduce the workload of ATCos and pilots. In this section, the DRL algorithms used in conflict resolution are summarized, as shown in Table 1.

Table 1. Deep reinforcement learning algorithms in conflict resolution.

Article	Algorithm	Authors	Year	Achievement
[22]	Heuristic Dyna-Q algorithm with Value Approximation	Yang et al.	2014	Similar to the principle of DRL
[23]	Strategic Conformal Automation	Regtuit et al.	2018	Training based on ATCos data
[24]	DDPG with Model Voltage Potential	Ribeiro et al.	2020	Combination with geometric method
[25]	Deep Q-Learning from Demonstrations	Hermans	2021	Explainability of the automation is contributed
[26]	Hierarchical DRL Framework	Brittain et al.	2018	Hierarchical agents are used
[27]	Deep Distributed Multi-Agent RL Framework	Brittain et al.	2019	Multi-agent RL is adopted
[28]	Deep Distributed Multi-Agent Variable	Brittain et al.	2021	LSTM network is used
[29]	Deep Distributed Multi-Agent Variable-Attention	Brittain et al.	2020	Attention network is added
[30]	Dropout and Data Augmentation Safety Module	Brittain et al.	2021	Can be used in uncertain environment
[31]	Single-Step DDPG	Pham et al.	2019	Continuous action space and non-fixed step
[32]	AI Agent with Interactive Conflict Solver	Tran et al.	2019	A conflict solver is developed
[33]	K-Control Actor-Critic	Wang et al.	2019	Decision-making is improved based on ATC process
[34]	Physics Informed DRL	Zhao et al.	2021	Prior physics understanding and model are integrated
[35]	Independent Deep Q Network	Sui et al.	2021	The DRL model closest to ATC process
[36]	DDPG	Wen et al.	2019	The efficiency of DRL method is proved
[37]	Corrected Collision Avoidance	Li et al.	2019	Correction network is added
[38]	Graph-Based DRL Framework	Mollinga et al.	2020	Graph neural network is added
[39]	Message Passing Neural Networks-Based Framework	Dalmau et al.	2020	Message passing neural network is added
[40]	Deep Ensemble Multi-Agent RL Architecture	Ghosh et al.	2021	Deep ensemble architecture is adopted

Yang used RL for UAV conflict resolution in 2014 [22]. Although DRL is not adopted, the linear value function approximation used is similar to the principle of DRL. Some universities and researchers continue to carry out research since 2018. The Delft University of Technology has been engaged in the research of conflict resolution, and DRL is one of its important directions [23–25]. It has put forward many representative achievements and ideas, including Strategic Conventional Automation, DDPG with Model Voltage Potential (MVP), and Deep Q-Learning from Demonstrations (DQfD). Brittain and Wei proposed a variety of continuously improved algorithms, including the hierarchical method, Deep Distributed Multi-Agent RL Framework (D2MA) algorithm, variant algorithms of Deep Distributed Multi-Agent Variable (D2MAV) and Deep Distributed Multi-Agent Variable-Attention (D2MAV-A), and the algorithm combining dropout and data augmentation [26–30]. Pham

and Tran used DDPG algorithm to study conflict resolution in a free flight scenario and developed solution software [31,32].

The achievements obtained in some studies are state of the art. Wang proposed the K-Control Actor-Critic algorithm to guide an intruder to make a fixed number of maneuvers for conflict resolution in the free flight scenario. It is verified that DRL has superior efficiency compared with traditional approaches [33]. Zhao proposed a physics informed DRL method which integrates prior physics understanding and model to facilitate the optimal policy searching and to present human-explainable results for display and decision-making [34]. Sui used multi-agent DRL to solve the multi-aircraft conflict problem in a three-dimensional en-route scenario, and analyzed its performance. Its conflict resolution model is closest to the current air traffic control process [35].

Other researchers have also made good achievements, which can give readers a variety of inspiration. Wen [36] used the DDPG algorithm for conflict resolution, whose performance is close to that of traditional methods, while its calculation time is greatly reduced. Li [37] proposed a corrected collision avoidance system, which can operate more efficiently than traditional methods in dense airspace while maintaining high levels of safety. Mollinga [38] presented a model that guides an arbitrary number of aircraft across three-dimensional, unstructured airspace while avoiding conflicts. Dalmau [39] presented a tool based on multi-agent RL to support ATCos in complex traffic scenarios. Ghosh [40] developed a novel deep ensemble multi-agent RL framework that efficiently learns to arbitrate between the decisions of the local kernel-based RL model and the wider-reaching DRL model.

Unlike supervised learning agent, DRL agents improve their abilities through continuous interaction with the environment. The conflict resolution training process is shown in Figure 4. Through training in the environment, an agent has the ability of conflict resolution and then can be validated in special air traffic control simulation software or realistic situations.

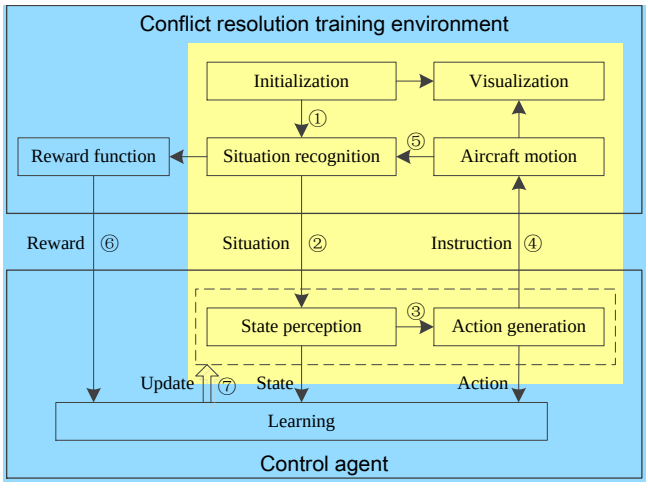


Figure 4. Conflict resolution agent training process.

Aircraft control in conflict resolution using DRL is an episodic task and a training episode is composed of training steps. In each training step, first, the information recognized by ATCos is sent to the control agent as a tuple of state, see Steps 1 and 2 in Figure 4. Then, the agent generates an action through its neural networks and sends it to the environment, as shown in Steps 3 and 4. Next, the environment calculates the next state and a scalar reward, then sends them to the agent, as shown in Steps 5 and 6. Since agent training is an iterative process, all the steps except Step 1 are executed repeatedly in one episode. Last, for the update of neural networks, the tuple of the current state, action, reward, and the next state in each step is used, see Step 7.

By analyzing these papers, this section discusses the key contents of using DRL to solve conflicts, including the training environment used, the state, action and reward function of the model, the advantages and applicable scenarios of each algorithm, as well as the evaluating indicator.

3.2. Environment

DRL agents need to be trained in an environment so that they can have decision-making ability. There are available environments in areas where DRL methods are well applied, such as Arcade Learning Environment (ALE) [41] for the video game, MuJoCo [42] for robotic control, and the integration framework OpenAI Gym [43]. For conflict resolution, a DRL environment is needed to provide an aircraft maneuver model and flight scenario, to support agents to continuously learn in the environment and realize the decision-making. However, there is no general conflict resolution environment, which is the main limitation of establishing unified research in the community.

This section classifies and summarizes environments used in the existing methods, as shown in Table 2, including the following factors:

1. Geometric dimension. The current environment is either operating altitude, speed and heading in three-dimensional space, or operating speed or heading in a two-dimensional horizontal plane.
2. Operation mode. En-route is implemented in most airspace at present, and free flight, which has been implemented in some airspace in Europe, is the trend of operation mode in the future. Therefore, the DRL environment should be clear about whether to solve the problem of conflict resolution in en-route mode or free flight mode.
3. Aircraft number. It is necessary to specify the number of aircraft for conflict resolution, 2 or more generally, and multiple aircraft also include a fixed number and arbitrary numbers.
4. Platform. DRL environment can be developed based on commercial or open-source ATC software, or independently.

Table 2. Environment.

Article	Dimension	Operation Mode	Aircraft Number	Platform
[22]	2D	Free Flight	Multiple, fixed	Independent development
[23]	2D	Free Flight	2	Aircraft Data
[24]	2D	En-route	Multiple, arbitrary	BlueSky
[25]	2D	En-route	2	BlueSky
[26]	2D	En-route	Multiple, fixed	NASA Sector 33
[27]	2D	En-route	Multiple, arbitrary	BlueSky
[28]	2D	En-route	Multiple, arbitrary	BlueSky
[29]	2D	En-route	Multiple, arbitrary	BlueSky
[30]	2D	En-route	Multiple, arbitrary	BlueSky
[31]	2D	Free Flight	2	Independent development
[32]	2D	Free Flight	Multiple, fixed	Independent development
[33]	2D	Free Flight	Multiple, fixed	Independent development
[34]	2D	En-route	Multiple, arbitrary	PyGame
[35]	3D	En-route	Multiple, arbitrary	ATOSS
[36]	2D	Free Flight	Multiple, fixed	Independent development
[37]	2D	Free Flight	Multiple, fixed	Independent development
[38]	3D	En-route	Multiple, fixed	CSU Stanislaus
[39]	2D	Free Flight	Multiple, arbitrary	Independent development
[40]	2D	En-route	Multiple, arbitrary	ELSA-ABM

When developing or selecting an environment, the spatial scope must be determined first. The currently used environment is mainly two-dimensional (2D), including a few three-dimensional (3D). 2D environment refers to resolving conflicts in a horizontal plane by adjusting heading, speed, or both. In [22,23,31–33,36,37], the speed is fixed and only

the heading can be adjusted. In [26–30,40], only the speed can be adjusted. In [24,25,34,39], both speed and heading can be adjusted. The altitude can be adjusted in a 3D environment. In [35,38], the altitude, speed, and heading can be adjusted. Although there are more maneuvers in a 3D environment, the complexity of the 3D environment will make the training of agents difficulty. According to the requirement, if the research is to be carried out in complete airspace, a 3D environment is needed. If there is no need to adjust the altitude, such as keeping the track unchanged or cruising at a fixed flight level, the 2D environment can be used.

Almost all civil aviation airspace operations are in en-route mode. Although free flight has been implemented in a little airspace in Europe, it is still under planning and will be widely implemented in the future. The conflict resolution of en-route mode needs the support of route information, so all the environments used in the research of en-route mode are developed based on the air traffic management (ATM) platform [24–30,35,38,40]. The fictitious route is used in an en-route mode in [34], so ATM software is not used. Instead, the environment is developed based on the DRL framework PyGame [44] by adding ATC content. For free flight, the complex environment makes it more difficult for pilots and ATCos to operate. Therefore, all environments are 2D and are independently developed without the support of commercial or open-source platforms [22,23,31–33,36,37,39].

Conflict resolution in the scenario of two aircraft is generally the feasibility verification of DRL method, which is not as good as geometry or optimization method and has no performance advantage. Although the conflict resolution process is to resolve the conflict between a pair of aircraft, in the scenario of multiple aircraft, additional conflicts may be generated when solving a pair of conflicts, so the problem will become complex. The advantage of using scenario with a fixed number of aircraft is that it has low demands for the algorithm, while the disadvantage is that the trained agent can only be used in the scenario with a certain number of aircraft. For example, in [33], the conflict resolution of 2–5 aircraft scenarios is carried out. For scenarios with different numbers of aircraft, multiple agents need to be trained pertinently. For the scenario of arbitrary number of aircraft, such as [24], there is a limitation of 20 aircraft, and any number of aircraft is supported within the range of 20. In [34], the number of aircraft is not limited, and the algorithm can be flexibly applied to all kinds of airspace. However, it has high requirements for the design of DRL model and training algorithm.

There are three roadmaps for the development of environment: independent development, adding conflict resolution function based on DRL platform, and encapsulating DRL interface based on ATM software. The advantage of independent development is that it can effectively focus on the problem which is need to be solved, while its disadvantage is that it takes more development time, lacks persuasion, and is difficult to establish a research baseline. The advantage of adding conflict resolution function in DRL framework is that it conforms to DRL specification and is easier to implement agent training, such as [34]. However, the essence of this method is still independent development. Supported by professional data and scenarios, it is a recommended method to expand based on ATM platform.

At present, the open-source ATM platforms mainly include CSU Stanislaus [45], ELSA-ABM [46] and BlueSky [47,48]. CSU Stanislaus is an open-source software developed in Python for ATC. However, its latest version was released in 2015 and is no longer updated. ELSA-ABM is an agent-based open ATM simulator and focuses on investigating traffic complexity metrics using scenarios that consist of flight plan inputs. However, it still lacks a simple and intuitive user interface for input and output and requires knowledge of the simulation setup. BlueSky is now a full-blown, user-friendly air traffic simulator that can run at higher update rates than expected for a high number of aircraft. The user interface is shown in Figure 5. To sum up, the authors recommend that the independent development environment be used to carry out the research in the free flight mode, and the environment based on the BlueSky platform be used to carry out the research in the en-route mode.

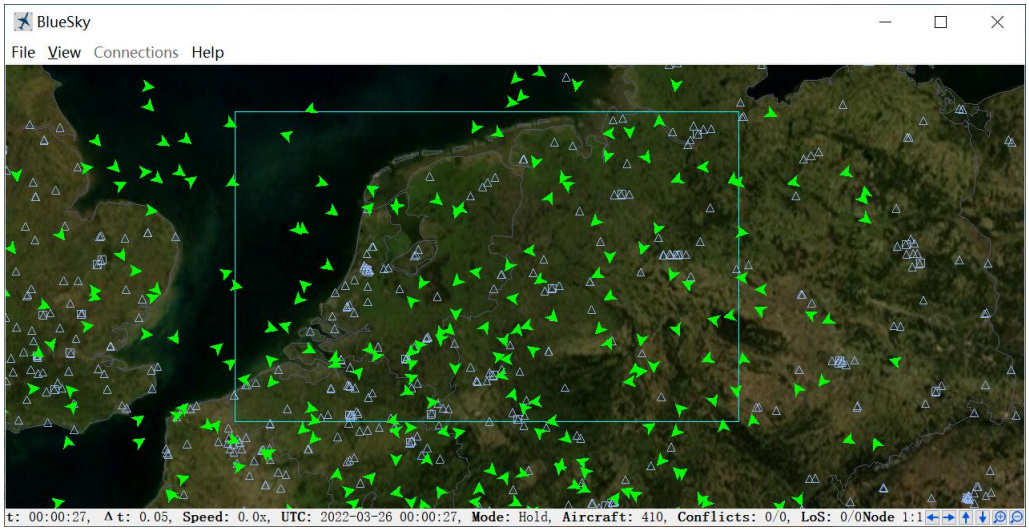


Figure 5. BlueSky user interface.

3.3. Model

Model is the key component of research, which needs to be established according to the conflict problem to be solved. Establishing DRL model of conflict resolution is actually establishing the MDP model, including action space, state space and reward function.

3.3.1. Action Space

In the DRL model, the reasonable action space is consistent with the instructions issued by ATCos. However, this action space will increase the complexity of the DRL model and the difficulty of agent training, so the model needs to be simplified. This section summarizes the action space of each model according to the adjustment content, control mode, action dimension, and decision-making times, as shown in Table 3. The adjustment includes altitude, speed, heading, and route operation. The control mode includes continuous mode and mode discrete. The dimension of discrete action space is the number of actions. Decision times refer to the number of instructions issued by ATCos in the conflict resolution phase, including one instruction and multiple instructions given in fixed or non-fixed steps.

Table 3. Action space.

Article	Adjustment	Control Mode	Action Dimension	Decision Times
[22]	Heading	Discrete	Multi	Fixed step
[23]	Heading	Discrete	3	1
[24]	Heading, speed	Continuous	2	Non-fixed step
[25]	Heading	Discrete	6	Fixed step
[26]	Route, speed	Discrete	Multi	1, fixed step
[27]	Speed	Discrete	3	Fixed step
[28]	Speed	Discrete	3	Fixed step
[29]	Speed	Discrete	3	Fixed step
[30]	Speed	Discrete	3	Fixed step
[31]	Heading	Continuous	2	Non-fixed step
[32]	Heading	Continuous	2	Non-fixed step
[33]	Heading	Continuous	4	Non-fixed step
[34]	Heading, speed	Both	3 for Dis., 2 for Con.	Fixed step
[35]	Altitude, heading, speed	Discrete	14	Fixed step
[36]	Heading	Continuous	1	Non-fixed step
[37]	Heading	Discrete	6	Fixed step
[38]	Altitude, heading, speed	Discrete	Multi	Fixed step
[39]	Heading, speed, route	Discrete	Multi	Fixed step
[40]	Speed	Discrete	3	Fixed step

In [35,38], 3D environments are used. Their action spaces include altitude adjustment, heading adjustment, and speed adjustment. Both are discrete, but the actions are different. the action space in [35] is expressed as $A = \{A_{Spd}, A_{Alt}, A_{Heading}, null\}$. Speed adjustment includes acceleration and deceleration that are integer multiples of 10 kts with a maximum range of 30 kts, which is expressed as $A_{Spd} = \{+10, +20, +30, -10, -20, -30\}$. Altitude adjustment includes ascent and descent in integer multiples of 600 m with a maximum range of 1200 m, which is expressed as $A_{Alt} = \{+1200, +600, -600, -1200\}$. Heading adjustment includes an offset and direct flight to the next way-point, where the offset is 6 NM, which is expressed as $A_{Heading} = \{DirectToNext, +6, -6\}$. This action space conforms to ICAO specifications and is the closest to reality in all research.

Unlike a 3D environment, for a 2D environment, only heading or speed can be adjusted. In [24,34,39], both heading and speed can be adjusted. The action space in [24] is continuous, and common values were assumed for turn rate (max: 15°/s) and acceleration/braking (1 kts/s). The action space in [39] is modeled as a set of $3 + n_\chi + n_v$ discrete actions, where n_χ and n_v are the numbers of possible heading and speed changes, respectively, and 3 actions are maintaining the current heading and speed, pointing towards the exit-point, and returning to the optimal speed. Several categories of action space are studied in [34], including discrete heading action space, continuous heading action space, speed action space, and simultaneous heading change and speed control action space. In addition, in ATC, ATCos can also issue instructions for route selection, such as [26,39].

The control mode of action space is mainly discrete. The action space dimension of [35] is 14, while the action space of other methods is smaller than 7. For continuous action space, only the heading can be adjusted in [31–33,36], while both heading and speed can be adjusted in [24,34].

In ATC, control instructions are usually issued at non-fixed time intervals. However, most DRL algorithms run according to fixed time steps. Some methods consider the characteristics of actual ATC and adopt non-fixed step size. Two control times were used, one time to resolve the conflict and one time to return to the way-point or fly to the exit-point [31,32,36]. In [33], a K-Control algorithm is proposed, where K represents the number of control times. Since the non-fixed step control requires the improvement of the DRL framework, most methods use fixed-step. However, it does not mean that it is not good to use a fixed step. A track composed of a series of points is formed by using a fixed step, which can be smoothed by reward shaping [49], to provide auxiliary decision-making for ATCos.

In most RL problems, the action space is chosen based on intuition [50]. Since ATCos gives discrete commands in actual ATC, it is recommended to adopt the control mode of [35] combined with [39]. The action space is expressed as:

$$A = \{A_{Spd}, A_{Alt}, A_{Heading}, A_{Spcl}\} \tag{16}$$

where A_{Spcl} represents a special action set, including maintaining the current position, pointing towards the next way-point, returning to the optimal speed, and so on. In the training process, the action space needs to be adjusted according to the phased results to achieve better performance.

3.3.2. State Space

ATCos rely on radar to obtain situation information and communicate with pilots through radio. Due to recent use of ADS-B, ATC interaction is transitioning to digital. No matter what kind of monitoring, the situation information of conflict resolution using DRL needs to be expressed as state space. This section summarizes the state space of each model according to the state scope, data structure, information displayed, whether pre-processing is required, and support variable number of aircraft or not, as shown in Table 4. State scope includes all aircraft, N-nearest aircraft and sub-airspace. Data structure includes a data matrix and raw pixels. The displayed information includes all aircraft information, self aircraft information and relative information. Pre-processing refers to expressing the

monitoring information as a state directly or after processing. The variable number of aircraft means that the state space can only be used in the scenario of a fixed number of aircraft or any number of aircraft.

Table 4. State space.

Article	Scope	Structure	Information	Pre-Processing	Aircraft Number
[22]	All aircraft	Data matrix	All information	No	Fixed
[23]	All aircraft	Data matrix	Relative information	Yes	Fixed
[24]	N-nearest	Data matrix	Relative information	Yes	Arbitrary
[25]	All aircraft	Raw pixels	All information	Yes	Fixed
[26]	All aircraft	Both	All information	No	Fixed
[27]	N-nearest	Data matrix	Self & relative information	Yes	Arbitrary
[28]	N-nearest	Data matrix	Self & relative information	Yes	Arbitrary
[29]	All aircraft	Data matrix	All & relative information	Yes	Arbitrary
[30]	N-nearest	Data matrix	Self & relative information	Yes	Arbitrary
[31]	All aircraft	Data matrix	Relative information	Yes	Fixed
[32]	All aircraft	Data matrix	Relative information	Yes	Fixed
[33]	All aircraft	Data matrix	All information	No	Fixed
[34]	All aircraft	Raw pixels	All & relative information	Yes	Arbitrary
[35]	Sub-airspace	Data matrix	All information	Yes	Arbitrary
[36]	All aircraft	Data matrix	All information	No	Fixed
[37]	All aircraft	Data matrix	All information	No	Fixed
[38]	All aircraft	Data matrix	All & relative information	Yes	Fixed
[39]	N-nearest	Data matrix	Self & relative information	Yes	Arbitrary
[40]	N-nearest	Data matrix	Self & relative information	Yes	Arbitrary

The state scope includes all aircraft in the airspace in most studies, which is simple in design but poor in expandability. Changes in the number of aircraft will lead to changes in the dimension of state space, which requires retraining a new agent. In [24,27,28,30,39] and [40], the state scope contains the N-nearest aircraft of one aircraft. This method has the advantage that the number of aircraft in the airspace is variable on the premise of unchanged state space dimension, which is more adaptable than the state scope including all aircraft. In [35], the airspace is divided into several parts, and the information of all sub-airspace is regarded as a state. Although the number of aircraft in this method is not limited, the dimension of state space is large and information in most sub-airspace is invalid.

The matrix structure data is adopted by most studies, which is easy to implement, and only needs to list the aircraft information or relative relationship to be included. The combination of the raw pixels and a matrix is adopted in [26]. The parent agent selects the route through the raw pixels, and the child agent outputs the control decision according to the matrix information of the selected route. The solution space diagram (SSD) is used as state data input in [25,34]. SSD was first proposed as an alternative metric to predict workload for ATCos [51], and further extended as a visual aid for possible conflict detection and resolution tasks on navigational display [52].

The SSD as a method of expressing conflict resolution state is illustrated in Figure 6. The circle around an aircraft represents the protection zone. The relative velocity vector can be constructed and a triangular area can be identified for which the aircraft would be in conflict in relative space. Conflict will occur if the relative velocity vector is within the conflict cone. The relative velocity can be translated to the absolute velocity and the conflict cone is also translated by adding the corresponding intruder's velocity vector to

the coordinates of each point. Therefore, it indicates a potential conflict if the endpoint of the absolute velocity is within the conflict cone area. Although SSD contains all relevant information about conflict resolution, it does not contain enough information for ATCos to identify the most effective solution. Therefore, in the use of SSD, necessary extensions can be made from the stacking, color, look-ahead time, and other directions.

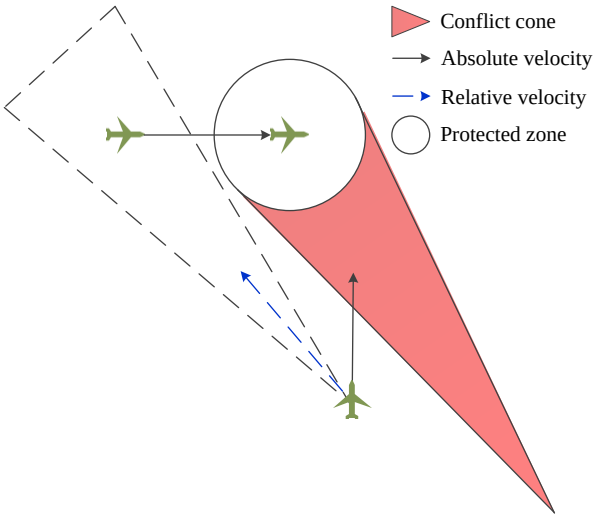


Figure 6. SSD for conflict detection.

If the state scope includes all aircraft and all information is displayed directly, this is the simplest state space, and pre-processing is usually not required, such as in [22,26,33,36,37]. There are two methods to display the relative information: one is to display the relationship between aircraft at the current time, and the other is to display the predicted conflict between two aircraft. By predicting the future trajectory of the aircraft, in [24,31,32], the predicted time and position of conflict are used as state input. Other state spaces using relative information take the relative distance and relative heading between aircraft as the input. The use of relative information must pre-process the original data, which can improve the training efficiency but will increase the workload in the early stage. For a multi-agent DRL algorithm, each aircraft information combined with the relative information of N-nearest aircraft can be used as state input, such as [27,28,30,40].

Using N-nearest or SSD methods, the number of aircraft can be changed with a fixed dimension of state space. Using a multi-agent DRL algorithm, although the dimension of state space of each agent is fixed, the number of aircraft can be changed by changing the number of agents. Taking the sub-airspace as the state input can also solve the conflict resolution problem of a variable number of aircraft. Besides, for other types of state space, the algorithm needs to be improved to be used in the scenario with a variable number of aircraft. For example, in [28], a Long Short Term Memory (LSTM) [53] network is used to encode all of the aircraft's information into a fixed-length vector, which allows the agent to have access to all aircraft's information without defining a max number of agents to consider.

In the 2D horizontal plane, if the number of aircraft is fixed, the matrix expressing all aircraft information or relative information can be used as the state input. If the number of aircraft is variable, SSD is recommended as the state input. For 3D airspace, it is recommended to use a multi-agent algorithm or state space combined with a data matrix and SSD.

3.3.3. Reward Function

Unlike supervised learning, DRL takes the reward function as the updated label of the neural network. Therefore, the reward function has an important impact on the training speed, convergence, and performance of agents. For conflict resolution, the objective of the

reward function is to improve flight safety, improve efficiency and meet the requirements of ATC scenarios. The purpose of conflict resolution is to prevent a collision, so safety is the primary consideration. On the premise of ensuring safety, it is also very important to improve the efficiency, rationality, and economy of agent decision-making. In addition, ensuring legitimacy and consistency with ATCos decisions are also constraints on the design of reward function.

According to the requirements, the reward function is designed and shaped, usually including minimum separation maintenance design, additional cost penalty, and other penalties, as shown in Table 5. There are two approaches to set the reward to maintain the minimum separation. One is to punish the conflict only in the termination step. The other is to provide guided shaping at each step to maintain a safe distance. The penalty of additional cost refers to the penalty of altitude change, heading angle change, speed change, additional fuel cost, and delay caused by ATC instructions. Other penalties include deviation from preset route, decision inconsistency, invalid action, and illegal instruction.

Table 5. Reward function.

Article	Minimum Separation Maintenance	Additional Cost Penalty	Other Penalties
[22]	Termination step	/	Deviation from preset route
[23]	Termination step	/	Decision inconsistency, invalid action
[24]	Termination step	Heading angle, speed	/
[25]	Termination step	Heading angle	Deviation from preset route, invalid action
[26]	Termination step	Speed	/
[27]	Every step	/	Invalid action
[28]	Every step	/	/
[29]	Every step	/	/
[30]	Every step	/	Invalid action
[31]	Termination step	/	Deviation from preset route, illegal instruction
[32]	Termination step	/	Deviation from preset route, illegal instruction
[33]	Termination step	Heading angle	/
[34]	Termination step	/	Deviation from preset route
[35]	Every step	Heading angle, speed, altitude	Illegal instruction
[36]	Termination step	Heading angle	/
[37]	Every step	Heading angle	/
[38]	Every step	/	/
[39]	Every step	Heading angle, speed, delay	Deviation from preset route
[40]	Every step	Delay, fuel	/

Safety is the reward to be considered in every study, and it has the highest priority, which can not be influenced by any other reward. Most studies take whether the distance between aircraft is less than the safety separation as the standard, and reward or punish it at the termination step. Some researchers divide different separation regions according to the distance and give an agent different penalties in different regions. In [27–30], the distances of 3 NM and 10 NM are adopted, and the reward function is defined as:

$$R_t = \begin{cases} -1, \text{ if } d < 3 \\ -\alpha + \beta \times d, \text{ if } d < 10 \text{ and } d > 3 \\ 0, \text{ otherwise} \end{cases} \tag{17}$$

where d is the distance from the own-ship to the closest aircraft in nautical miles, and α and β are small, positive constants to penalize agents as they approach the loss of separation distance. The conflict radius and alert radius are defined in [39], and the conflict radius and congestion radius are defined in [40]. The reward function in the following form is constructed:

$$R_t = \alpha I(o_1, d_1) + \beta I(o_2, d_2) \tag{18}$$

where the indicator function $I(x)$ is 1 if the condition x is satisfied and 0 otherwise. The value is set to 1 if the relative distance from the nearest aircraft is less than the separation radius according to the current observation, and otherwise, it is set to 0.

In [35], four situations are divided, and the rewards under different situations are given by:

$$R_t = \begin{cases} p_1, \text{ if conflict resolution is successful} \\ p_2, \text{ if conflict resolution is failure and there are new conflict} \\ p_3, \text{ if conflict resolution is failure and there are no new conflict} \\ p_4, \text{ time is out and the resolution is not complete} \end{cases} \tag{19}$$

where p_1 is the maximum reward, and the shorter the resolution time, the higher the reward value. The characters p_2 and p_3 are negative rewards and $|p_2| \geq |p_3|$. Usually, $p_4 > 0$ and is a relatively small reward.

After an action is output by the agent, the pilot adjusts altitude, heading angle, and speed of the controlled aircraft, which will incur additional costs. This will also have three adverse effects: increased pilot workload, increased fuel consumption, and delay. These factors should be considered in the process of reward shaping, and the additional cost should be minimized on the premise of safety. This is closely related to the action space. Some studies give the reward value through the direct impact on the action, while others make further calculations. In [33,36,37,51], the additional heading angle is punished. In [26], the additional speed deviation is punished. In [24,39], both additional heading angle and speed deviation are punished. In [35], the additional heading angle, speed deviation, and altitude adjustment are all punished. In addition to the adjustment penalty of heading angle and speed, in [39], the delay penalty is carried out by using the deviation between adjusted speed and optimal speed. Delay and fuel cost penalties are used in [40], and these reward for an agent i at time t is:

$$R_t^i = \gamma \times max(0, \hat{d}_t^i - d_t^i) + \delta F(v_t^i - v_O^i) \tag{20}$$

The character $\hat{d}_t^i$ denotes the expected distance to travel by aircraft i at time t according to the given schedule and d_t^i denotes the actual distance traveled, and then $max(0, \hat{d}_t^i - d_t^i)$ characterizes the amount of delay. The term $F(v_t^i - v_O^i)$ represents the quadratic fuel cost function depending on the deviation of current speed from the optimal speed. γ and δ are weights for delay and fuel cost penalties.

In [24], the operating data of ATCos is taken as the environment, and the deviation from the decision made by ATCos is added as punishment in its reward function. The reward is given by computing the absolute error between the agent's states and the reference states and is multiplied by a weight:

$$R_{Reference} = max(\alpha \times |States_{agent} - States_{reference}|) \tag{21}$$

The closer to the decision made by ATCos, the smaller the penalty. The conflict-free route is pre-designed in [22], and the agent will be punished for the deviation between the real trajectory and the expected route caused by its action. For the scenario of preset waypoints for aircraft, the deviation from the preset waypoints during flight can be punished, such as in [25,39]. Penalties for changes in heading angle and speed are both designed in [34]. The reward function to minimize disruption is defined as:

$$R_h = 0.001 cos(\theta) \tag{22}$$

where θ is the angle between the action direction and the intended direction. This term rewards the action following the intention and penalizes the action of deviation. The reward function to moderate speed change is defined as:

$$R_s = 0.001e^{-(\frac{v-v_0}{v_u-v_l})^2} \tag{23}$$

where v is the current speed, v_o is the original speed, v_l and v_u are the lower bound and upper bound of the speed, respectively.

An intuitive method of punishing the deviation of the aircraft from the original trajectory is used in [31,32], as shown in Figure 7. $Aircraft\_1$ makes a heading change α at point A, and continues in the new heading AC, then towards the original endpoint B at return point C. The deviation area is calculated:

$$S_{ABC} = l_{AC} \times l_{AB} \times sin(\alpha)/2 \tag{24}$$

The maximum deviation area that could happen is the area that takes the sector border as its circumcircle:

$$S_{max} = (3\sqrt{3}r^2)/4 \tag{25}$$

where r is the radius of the sector. Employing this, the punishment for a large deviation is computed as:

$$R = -S_{ABC}/S_{max} \tag{26}$$

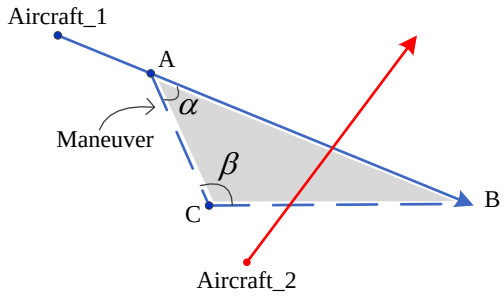


Figure 7. Deviation of the resolution from original path.

The illegal instruction refers to the instruction beyond the scope of aircraft performance or in violation of control habits, which needs to be punished. In [31,32], as shown in Figure 7, if the return point C is falls outside the circular sector or the return angle $\beta < 120°$, the agent will receive a negative reward value. In [35], if the action falls within the range of infeasible solutions, such as the climbing action followed by the descending action, a negative reward is given.

In order to reduce the workload of ATCos and pilots, control instructions should not be given frequently in actual ATC process. In [23,25,28,30], to take into account the preference of limiting the maneuver of ATCos and pilots, any actions that require a maneuver are penalized as well. The reward function that penalizes invalid action is given:

$$R_{action} = \begin{cases} 0, \text{ if no ATCos and pilots maneuvers} \\ a, \text{ otherwise} \end{cases} \tag{27}$$

where a is a negative value.

Reward shaping is important and necessary when using DRL for conflict resolution. The safety function must be defined, and the efficiency function should be designed according to the actual requirements. The reward function is not designed as meticulous as possible. Too detailed reward function may cause the opposite result. Therefore, in order to improve the performance of agents, the phased results in the training process should be analyzed, and the reward function should be adjusted according to this.

3.4. Algorithm

The steps of algorithm design are as follows: firstly, according to the model, choose whether to use a single agent or multi-agent algorithm. Secondly, select the appropriate algorithm classification and basic algorithm. Finally, improve the selected algorithm according to the problem to be solved. This section classifies and summarizes algorithms used in the existing methods, as shown in Table 6, including the following factors:

1. Agent number. The selection of a single agent or multi-agent mainly depends on the requirements for solving problems. The single agent has two advantages. One is in line with the actual ATC process, that is, one ATCo controlled all aircraft. The other is relatively simple in technology and easy to implement. However, its disadvantage is that it is difficult to deal with the changing number of aircraft. Multi-agent is opposite to a single agent, and communication between agents is needed.
2. Basic algorithm. The choice of the basic algorithm is closely related to the type of action space. DQN algorithm can only deal with discrete action space and is adopted in [25,26,35,37,38]. DDPG can only deal with continuous action space and is adopted in [24,31,32,36]. The standard actor-critic architecture can handle both continuous and discrete action spaces, which is improved in [33,39]. PPO is one of the best performing algorithms at present, which is used in [27–30,34].
3. Improved algorithm. Based on the basic algorithm, there are two improved approaches. One is to improve the structure of the agent according to the characteristics of ATC, including the improvement of neural network structure or agent structure. The other is the improvement of the DRL training method to improve the accuracy of selecting actions according to the situation.

Table 6. Algorithm.

Article	Agent Number	Basic Algorithm	Improved Algorithm
[22]	Single	Dyna-Q	Training method improvement
[23]	Single	Q-learning	/
[24]	Single	DDPG	Training method improvement
[25]	Single	DQN/Double DQN	Training method improvement
[26]	Single	DQN	Agent structure improvement
[27]	Multi	PPO	Agent structure improvement
[28]	Multi	PPO	Agent structure improvement
[29]	Multi	PPO	Agent structure improvement
[30]	Multi	PPO	Agent structure improvement
[31]	Single	DDPG	/
[32]	Single	DDPG	/
[33]	Single	Actor-critic	Training method improvement
[34]	Single/Multi	PPO	Training method improvement
[35]	Multi	DQN	Training method improvement
[36]	Single	DDPG	/
[37]	Single	DQN	Agent structure improvement
[38]	Single	DQN	Agent structure improvement
[39]	Multi	Actor-critic	Agent structure improvement
[40]	Multi	Model-based	Agent structure improvement

3.4.1. Agent Structure Improvement

In the aspect of agent structure improvement, a hierarchical structure including a parent agent and a child agent is proposed in [26], as shown in Figure 8. The parent agent will take an action to change the route and then the child agent will control the actions of changing speeds. The parent agent takes the raw pixels as the input and outputs the

selected path. Figure 9 shows an example of a state of a parent agent, whose state is represented as follows:

$$S_P = (p_1, p_2, \cdots, p_{m \times m}) \tag{28}$$

where p_k represents the intensity tuple of pixel k. Its action space can be defined as follows:

$$A_P = (C_1, C_2, \cdots, C_j), \forall j \tag{29}$$

where C_j is the combination of route that the aircraft will take.

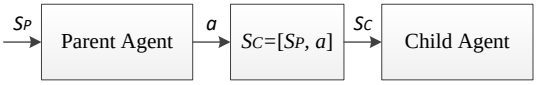


Figure 8. Progression from parent agent to child agent. Reprinted with permission from [26]. 2018, Peng.

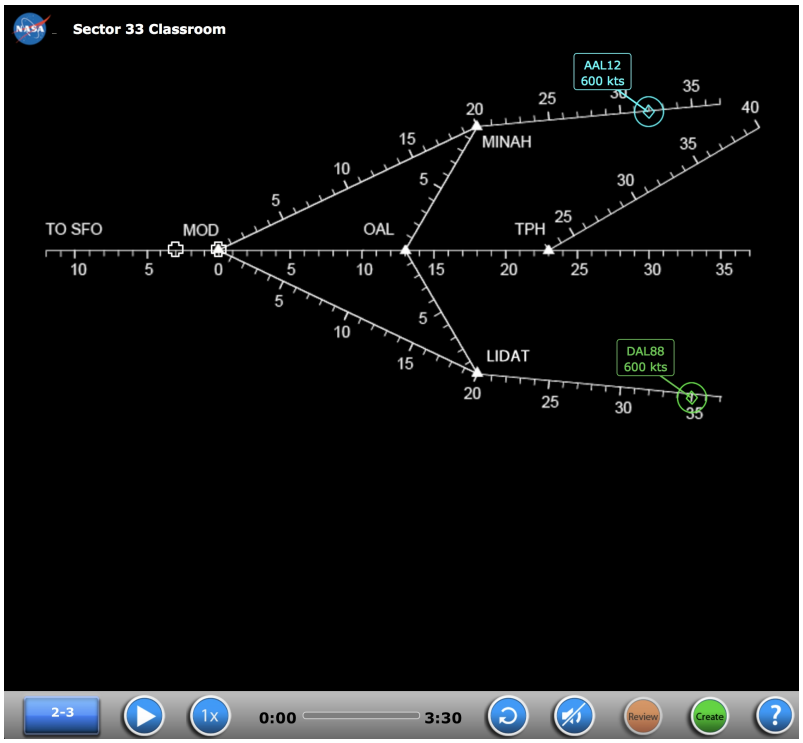


Figure 9. Example of a state of parent agent. Reprinted with permission from [26]. 2018, Peng.

For the child agent, the state is defined as:

$$S_C = (x_1, x_2, y_1, y_2, v_1, v_2, C_j) \tag{30}$$

where the subscript represents a specific aircraft. The action space can be defined as follows:

$$A_C = (U_1, U_2, \cdots, U_k), \forall k \tag{31}$$

where U is all of the possible combinations of speeds for the aircraft and k is a unique speed combination.

In [27], a fully connected (FC) neural network and N-nearest multi-agent algorithm are used, which can be used in dynamic scenarios with a varying number of aircraft. The basic agent structure is shown in Figure 10a. Based on this structure, an LSTM [53] is used instead of FC to encode information into fixed-length vectors to flexibly handle a variable number of agents [28], as shown in Figure 10b. Unlike LSTM, which may lose past information by propagating hidden vectors forward through time, attention networks [54] have access to

the global set of hidden vectors. The attention network is adopted in [29] to encode and control the importance of information into a fixed-length vector, as shown in Figure 10c. With this framework, the neural network's policy can be implemented in all aircraft at the beginning of each episode, and each aircraft then follows this policy until termination.

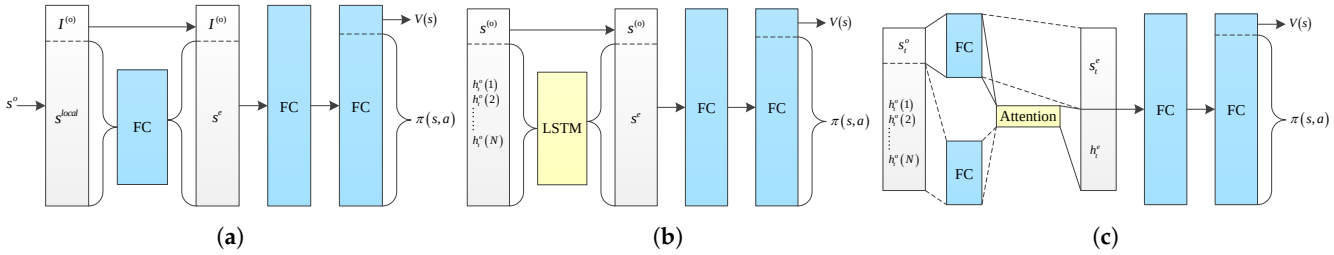


Figure 10. Attention network architecture evolutionary process. (**a**) Fully connected structure. Reprint with permission from [27]. 2019, IEEE; (**b**) LSTM structure. Reprint with permission from [28]. 2021, Peng; (**c**) Attention network architecture. Reprint with permission from [29]. 2020, Peng.

In [38], the graph convolutional neural network (GCN) [55] is applied, and a feature matrix $X \in \mathbb{R}^{N \times D}$ and three adjacency matrices $A \in \mathbb{R}^{N \times N}$ are taken as inputs, where N is the number of aircraft and D is the input dimension, as shown in Figure 11. Implementing three parallel graph-based layers with different adjacency matrices allows the aircraft to understand its surrounding on multiple levels. The first adjacency matrix takes global information into account, allowing each aircraft to always have a view of its surrounding. The second adjacency matrix only takes aircraft into account inside the detection area. The third adjacency matrix only takes aircraft into account inside the penalty area, providing more information if aircraft are closer.

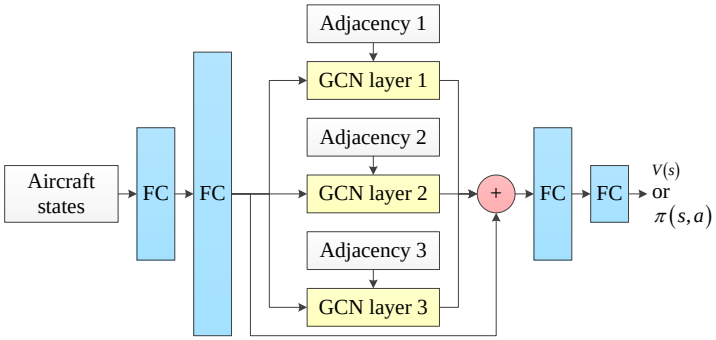


Figure 11. Neural network architecture with graph convolutional layers. Reprint from [38].

Message Passing Neural Networks (MPNN) is a general framework for supervised learning on graphs [56]. A recommendation architecture for ATC that combines multi-agent RL and MPNN is proposed in [39]. Flights correspond to the nodes of a graph, and edges include features about their relative positions. Aircraft are modeled as the agents of a cooperative multi-agent system. The interactions between agents are represented in the form of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node in $\mathcal{V}$ corresponds to one agent in N. The edges of the graph indicate which agents can communicate, and they include some features relative to the two agents that they connect. At every time step, each agent observes its state and encodes it into a hidden state. Then, agents start the message passing phase composed of several communication rounds. After the message passing phase, the probability distribution over all possible actions for each agent is generated. Finally, each agent samples an action from the distribution and acts accordingly.

A novel deep ensemble multi-agent reinforcement learning architecture is proposed in [40], as shown in Figure 12. A kernel model [57] is designed to employ agent-centric local observation information and a DRL model is adopted with extended richer state

information. The kernel model has the advantage of its theoretical convergence and asymptotic bounds for the model learned from experience training samples while it is only feasible if the dimension of the state space remains small. A trained DRL agent may have pathology due to non-linear function approximation and can be brittle in certain state regions. Therefore, the ensemble architecture is developed that leverages the pre-trained policies from the kernel and multi-agent DRL to efficiently arbitrate the complex boundary between the effectiveness of these two models and to obtain the final ensemble policy.

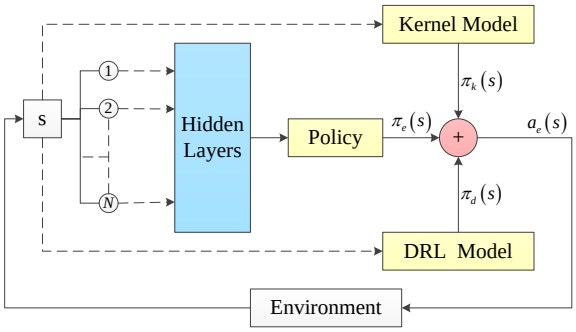


Figure 12. Architecture of the deep ensemble multi-agent RL. Reprint from [40].

An approach that adds corrections learned [58] through DRL for improving collision avoidance in dense airspace is proposed in [37]. The formulation of policy correction can be derived from multi-fidelity optimization, as shown in Figure 13. A surrogate model combines a simpler low-fidelity model f_{lo} and an additive parametric correction term δ can be used to approximate a high-fidelity model f_{hi} as $f_{hi} \approx f_{lo} + \delta$. Then, a parameterized correction term to approximate $Q^*(s,a)$ is added by:

$$Q^*(s,a) = (1 - w_c)Q^*_{lo}(s,a) + w_c\delta(s,a;\theta) \tag{32}$$

where $\delta(s,a;\theta)$ is the correction term parameterized by θ, and w_c is the weight placed on the correction.

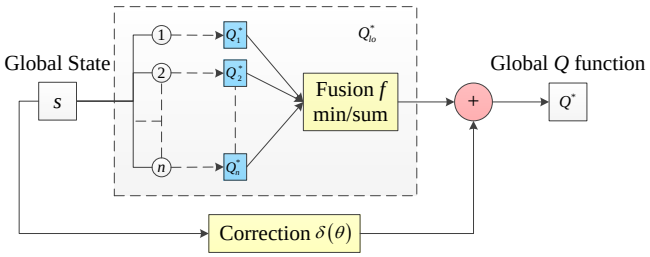


Figure 13. Utility decomposition with correction. Reprint from [37].

A dropout and data augmentation (DODA) safety module is proposed to improve the safety of the DRL model in unseen environments with uncertainties in [30], as shown in Figure 14. The safety module incorporates a model safety sub-module to quantify the DRL model uncertainty based on Monte Carlo (MC)-dropout (DO) and a state safety sub-module based on execution-time data augmentation (DA) to handle the state uncertainty. The state safety sub-module generates m disturbed states $\hat{s}_j (j = 1, 2, \cdots, m)$ and the model safety sub-module calculates the action distribution p_j for each disturbed state. For each disturbed state $\hat{s}_j$, the model safety sub-module samples n forward passes with MC-dropout and generates an action distribution p_j. Then m disturbed states $\hat{s}_j$ and the corresponding action distributions p_j are used in the state safety sub-module. The majority vote is implemented in the majority-based method to select the final action a^* while the minimal entropy is used in the entropy-based method to select the final action a^*.

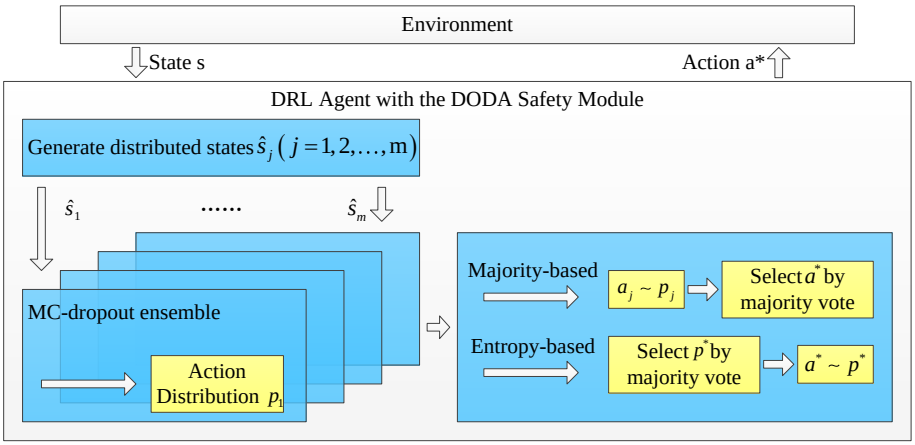


Figure 14. Structure of the DODA safety module. Reprint with permission from [30]. 2021, Peng.

3.4.2. Training Method Improvement

Another way is to improve from the perspective of the training algorithm without changing the structure of the agent. A hybrid solution between Model Voltage Potential (MVP) [59] approach and DDPG is created in [24], aimed at improving conflict resolution performance at high densities. The geometric resolution of the MVP model is displayed in Figure 15. In the MVP model, the calculated positions at the closest point of approach (CPA) repel each other. The repelling force is converted to a displacement of the predicted position at CPA, in a way that the minimum distance will be equal to the required minimum separation between aircraft. Such displacement results in a new advised heading and speed. The MVP model is used to pre-train the critic network of DDPG to make the model run within a safer operating range.

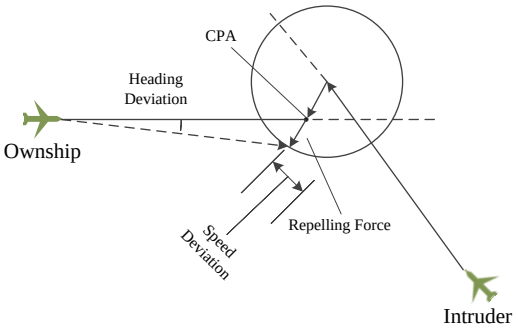


Figure 15. MVP resolution advisory based on geometry at CPA. Reprint from [24].

Inspired by pre-training, Deep Q-Learning from Demonstrations (DQfD) [60] is used in [25], as shown in Algorithm 1. A pre-training phase is employed in this algorithm to learn a policy that imitates the demonstration data, which is obtained from human manual performing. Results show that the DQfD agent achieves a more optimal policy with regard to minimizing the flight path and the number of resolution instructions. Although the DRL algorithms used in this method are DQN and double DQN, the pre-training phase can be added to any other DRL algorithms.

Algorithm 1 Deep Q-Learning from Demonstrations [60] used in [25].

Require:
- $\mathbb{D}^{replay}$: initialized with demonstration data set;
- θ: weights for initial behavior network;
- θ': weights for target network;
- τ: frequency at which to update target net;
- k: number of pre-training gradient updates;
- α: learning rate;
- $N_{trainingepochs}$: number of epochs for training.

for step $t \in \{1, 2, \cdots, k\}$ {pre-training phase} **do**
  sample a mini-batch of n transitions from $\mathbb{D}^{replay}$ with prioritization
  calculate loss $L(Q)$ using target network
  perform a gradient descent step to update θ
  if t **mod** $\tau = 0$ **then**
    $\theta' \leftarrow \theta$ {update target network}
  end if
  $s' \leftarrow s$
end for
for step $t \in \{1, 2, \cdots, N_{trainingepochs}\}$ {normal training phase} **do**
  sample action from behavior policy $a \sim \pi^{\epsilon Q_\theta}$
  play action a and observe (s', r)
  store (s, a, r, s') into $\mathbb{D}^{replay}$, overwriting oldest self-generated transition if over capacity occurs
  sample a mini-batch of n transitions from $\mathbb{D}^{replay}$ with prioritization
  calculate loss $L(Q)$ using target network
  perform a gradient descent step to update θ
  if t **mod** $\tau = 0$ **then**
    $\theta' \leftarrow \theta$ {update target network}
  end if
  $s' \leftarrow s$
end for

A heuristic search algorithm is adopted in [22] for saving both CPU time and memory. For m aircraft, each aircraft has n actions, and the search space is n^m. It is supposed that each aircraft only conflicts with two other aircraft around it and the whole space can be decoupled into several sub-spaces, as shown in Figure 16. There are 5 aircraft and the original search space is n^5. The whole aircraft can be divided into five subsets by cutting the chain, and then the action space could be reduced to $4n^2$.

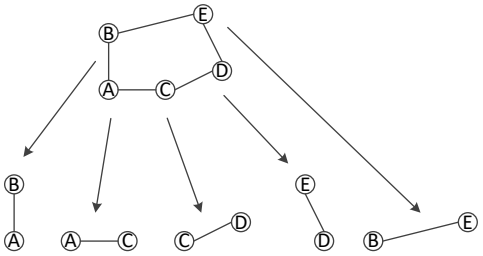


Figure 16. Decomposition of conflict connection. Reprint with permission from [22]. 2014, Jian.

Different from n-step RL, it is not necessary to control the aircraft in a fixed time step in the process of ATC. In [33], a K-Control Actor-Critic algorithm is proposed, where K represents the number of control times. On the premise of no conflict, the fewer control times, the better. The K-Control Actor-Critic algorithm is shown in Algorithm 2.

Algorithm 2 K-Control Actor-Critic. Reprint with permission from [33]. 2019, John Wiley and Sons.

Require: • ρ and φ are polar radius and angle, forming a two-dimensional polar coordinate, which can be used to describe a position in the sector;
• actor and critic neural networks.

if is training mode **then**
  for each episode **do**
    initialize random environment parameters
    for each step in range K **do**
      if is the Kth step **then**
        set the destination as the action
        run environment
        update parameters of critic network
      else
        obtain action by $\rho \sim N(\mu_\rho, \sigma_\rho), \varphi \sim N(\mu_\varphi, \sigma_\varphi)$
        update parameters of critic network
        update parameters of actor network
        if conflict **then**
          break
        end if
      end if
    end for
  end for
else
  for each testing episode or application situation **do**
    for each step in range K **do**
      if is the Kth step **then**
        set the destination as the action
        run environment
      else
        obtain action by $\rho = \mu_\rho, \varphi = \mu_\varphi$
        if conflict **then**
          break
        end if
      end if
    end for
  end for
end if

In [35], to solve the multi-aircraft flight conflict problem, an Independent Deep Q Network (IDQN) algorithm is proposed, as shown in Algorithm 3. IDQN is a simple learning framework that extends the DQN algorithm to solve a multi-agent problem, and each aircraft is controlled by one agent. For a scenario with n aircraft, n agents are required that correspond to them. Independence means that there is no coupling relationship and no communication behavior between agents. The neural networks of all agents share the same parameters, and data are sampled from the environment to update the network independently.

Algorithm 3 IDQN algorithm for the conflict resolution model [35].

initialize the experience replay pool of each agent with a capacity of D
initialize the action state value function Q_i and randomly generate the weights θ
initialize the target Q network with weight $\theta^- = \theta$
for each episode **do**
  randomly select the conflict scenario and initialize the state s_0
  for each step **do**
    each aircraft adopts an ϵ-greedy strategy to select instruction actions a_t^i from the action space to form joint actions u_0
    execute the joint instruction action u_0 according to the reward r_t received and the new state s_{t+1} of the aircraft
    save the conflict samples $(s_t, a_t^i, r_t, s_{t+1})$ into the experience playback pool D
    randomly select a conflict sample $(s_j, a_j^i, r_j, s_{j+1})$ from the experience pool D
    if the step $j+1$ is final **then**
      set $y_j = r_j$
    else
      set $y_j = r_j + \gamma \underset{a_i'}{max}\, Q(s_{j+1}, a_i'; \theta^-)$
      calculate the loss function $L_i(\theta_i) = E_{(s,a,r,s')}[(y_i^{DQN} - Q(s, a; \theta_i))^2]$
      update θ in $(y_j - Q(s_j, a_j; \theta))^2$ using gradient descent
      update the target Q network for each C step, $\theta^- = \theta$
    end if
  end for
end for

In [34], a meta control logic to let the aircraft return to its waypoint after deconflict is applied as shown in Algorithm 4. Once the endpoint of the intention velocity moves out of the conflict cone, the aircraft will choose the intention velocity to return to its original flight plan.

Algorithm 4 Meta controller [34].

Require: v: velocity of next step
while run **do**
  $obs. \leftarrow$ observe airspace
  $states \leftarrow SSDProcess(obs.)$
  velocity for resolution: $v_r \leftarrow Policy(states)$
  velocity for return: $v_i \leftarrow$ intention velocity
  conflict detection for v_i
  if conflict **then**
    $v \leftarrow v_r$
  else
    $v \leftarrow v_i$
  end if
end while

The objective of algorithm improvement is to better output decisions according to the situation in the process of conflict resolution. The performance of intelligent decision-making can be improved by both the improvement of agent structure and training algorithm. By improving the agent structure, the agent's understanding of perceived information and output actions can be enhanced, and then the decision-making performance can be improved. The improvement of the training algorithm is to reduce the scope of solution exploration in the training process through pre-training, action decomposition, and other methods, to improve the training efficiency.

3.5. Evaluating Indicator

The evaluation of agents or algorithms is a crucial process. The evaluating indicators include reinforcement learning indicators, conflict resolution safety and efficiency indicators, and other auxiliary indicators.

Reinforcement learning indicator refers to the indicators that all reinforcement learning methods need to use, including reward value and convergence speed. Although the reward functions designed by various methods are different, they all try to obtain a large convergent reward value according to their operation mechanism. The stability of reward can not be ignored. If the value of the reward changes constantly, the maximum value should not be taken as the indicator, but the fitting curve should be drawn, refer to [39], or the mean and standard deviation should be used. Since the reward is generally composed of multiple parts, each part of the reward can be evaluated separately to optimize the reward function more reasonably and improve the performance of agents. For the convergence speed, although the slow speed does not affect the performance of the final trained agent, the fast speed can reduce resource consumption and save time.

The reward curve is the main display method of the training process and results of reinforcement learning, which can intuitively display the performance and efficiency of agents or algorithms. The performance comparison of different algorithms can be intuitively displayed by drawing multiple reward curves in one graph. Through this, which method is efficient and which method has good performance can be seen [25,31]. In addition, the reward curve of the training process should not show the reward value obtained from a certain training, but the average value of multiple tests [33], or the curve with the maximum and minimum value [39].

The most important purpose of conflict resolution is to ensure aircraft safety, and its core indicator is how the conflict is resolved. There are many ways to evaluate the results of conflict resolution, such as flight success rate, conflict rate, number of conflicts, conflict distribution and so on. The display forms include: using curves to show the changes of the number of conflicts or the number of successful conflict resolution in the training process [24,27,31,33,35,39], showing the changes of the range of conflict rate [34], using tables to show the mean and standard deviation [27,37] or probability distribution [35] of the number of successful conflict resolution, or directly giving the success rate of conflict resolution [38].

The success rate of conflict resolution must be 100%. However, DRL is an approximation algorithm, which can not reach 100% in the published papers. To solve this problem, a method using multiple agents is proposed in [33]. Several agents are trained to obtain multiple solutions and reduce the conflict rate. One agent is used to generate a solution and determine whether the conflict is free. If the conflict still exists, the next agent is used until the conflict is completely resolved by the generated solution or all agents have been used.

Decision generation speed is an efficiency indicator and an advantage of DRL compared with traditional algorithms. Faster decision generation means detecting conflicts and generating instructions earlier, to reduce the workload of ATCos and pilots. In [33], DRL is compared with mixed-integer linear programming (MILP) [61]. The improved MILP takes 49 s to generate a solution, while DRL takes less than 0.2 s. However, these data are not obtained under the same calculation conditions. DRL data is obtained by a personal computer, while MILP data is directly cited from paper data. In [36], DRL and MILP are compared under the same conditions. The time for DRL to generate a solution is 0.098 s, while MILP takes 4.43 s. In [33], DRL is compared with genetic algorithm (GA), the average solving time of GA is 37.6 s, while that of DRL is 0.01 s. Through comparison, it can be found that DRL algorithms have great advantages in solving time, so they can improve decision efficiency.

In the evaluation of conflict resolution, the additional distance and time are also efficiency indicators. The distance increasing is caused by heading adjustment or trajectory change. In [39], the change of additional distance caused by ATCos operation is evaluated as the training goes on. The lengths of trajectory changes caused by different algorithms

are compared in [25,33]. In [37], the trajectories generated by algorithms are normalized based on the original trajectory. Heading adjustment is taken as an important indicator in [33,36]. The heading adjusted by different methods is compared and the smaller the angle adjustment, the better. In [40], according to the deviation between the adjustment speed and the optimal speed, the increase of fuel consumption is punished, and the action distribution under different fuel consumption scenarios is analyzed. Additional time consumption mainly refers to the delay caused by operation [35,38,40]. In [35], the statistics of the delay distribution show that most of the delays are distributed between −30 s and 30 s, and the solution generated by DRL does not bring too much delay. Safety and efficiency are a pair of contradictions. In [37], the trade-off between normalized distance and conflict rate is implemented. The results show that although the longer distance makes the conflict rate smaller, the efficiency becomes worse.

At present, conflict resolution is mainly based on the management of ATCos. The decisions generated by agents should be consistent with those of ATCos as much as possible. Some studies evaluate the similarity with human controllers. In [32], a heat map is used to display the similarity between the agent's solution and human solutions, which can intuitively show the difference between them. In [23,25], the resolution maneuvers provided by ATCos and agent are visualized in terms of the conflict angle and distance to the closest point of approach between the observed aircraft and the controlled aircraft. The current methods are mainly displayed qualitatively through charts, and there is no quantitative method for detailed comparison.

The habit of human controllers can be used for reference to quantitatively analyze the decision-making of agents, including the number of instructions and action distribution. In the actual ATC process, on the premise of ensuring safety, the fewer the number of instructions of ATCos and pilots, the better. In [39], the changes in agent instructions are counted, which shows that the number of agent instructions decreases significantly with the continuous training. In [38], the number of aircraft instructions is taken as an important indicator, and the number of instructions of different algorithms is compared. The less the number, the better the performance of the algorithm on this indicator. Action distribution can also be used to evaluate agents. In [29], the action distribution is used to evaluate the improved algorithm. The actions of the old algorithm are mainly distributed in acceleration and deceleration, while that of the new algorithm focuses on non-operation, which reduces the pilot's pressure. In [35], the actions performed in the process of conflict resolution are counted. The most used action is climbing 600m, which is consistent with that of ATCos. Finally, the number of aircraft also has a great impact on the performance of agents. In [33,34,37], the performance of agents in scenarios with a different number of aircraft is analyzed. With the increase in the number of aircraft, the convergence time of agent training and the possibility of conflict increase. Therefore, scenarios with a large number of aircraft have higher requirements for agent training and using.

4. Discussion

4.1. A Guideline for the Application of DRL to Solve Conflict Resolution Problems

It can be seen from the above that if DRL is used to solve the conflict resolution problem, the relationship between the actual conflict resolution problem and the DRL model should be established. Conflict resolution problem includes operation mode, geometric dimension, aircraft number, displayed information, instruction type, decision mode, and evaluating indicator. DRL model includes environment, state space, action space, reward function, and algorithm. The relationship between conflict resolution and the DRL model is shown in Figure 17. To realize the application of DRL in the field of conflict resolution, it needs to be designed according to the following steps.

1. Build a conflict resolution environment, which not only provides current or future conflict resolution scenarios, but also supports the training of agents.
2. Define the control object, whether it is one aircraft or multiple aircraft.

3. Define the input and output of the control process, that is, the state space and action space of the agent.
4. Construct the research goal in the form of a reward function, to make the agent meet the expected evaluation requirements.
5. Design appropriate agent structure and training algorithm.

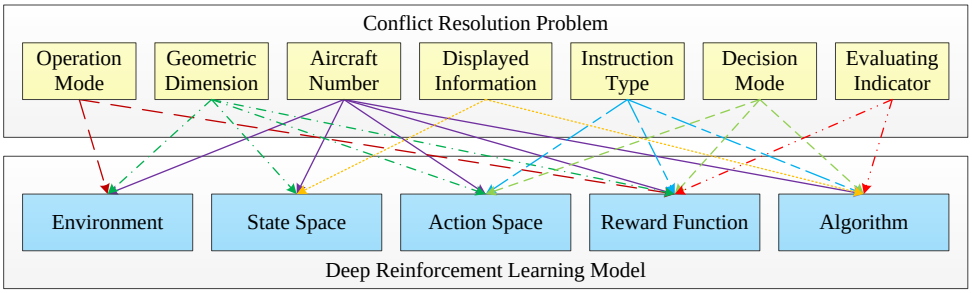


Figure 17. Relationship between conflict resolution and DRL model.

The operation mode affects the design of the environment and reward function. For en-route mode, the environment supports the aircraft to fly along the route, provides flight rules, and is developed based on the existing platform or independently developed. The rules of free flight mode are simple, but it lacks the support of an open platform and needs to be developed independently according to actual needs. The two modes have different requirements and evaluations on aircraft control instructions. Therefore, it is necessary to design relevant reward functions according to the selected mode.

Geometric dimensions should be considered in the design of the environment, state space, action space, and reward function. For the environment, if the altitude is not considered, the two-dimensional environment can be selected. On the contrary, the three-dimensional environment or three-dimensional electronic progress strip must be used. For the state space, the information needs to be expressed in a way consistent with the geometric dimension. If it is a scenario in three-dimensional space, altitude adjustment needs to be considered in the design of action space, which makes the action space more complex. In addition, different geometric dimensions have different requirements for the ATC process and different evaluations for conflict resolution. Therefore, the design of the reward function also needs to be improved.

Whether the number of aircraft is variable or not has an impact on all factors of the DRL model. The factor that is most affected by the number of aircraft is the state space. If the number of aircraft is variable, it is necessary to use the structured data of the specified maximum number of aircraft or the state space in raw pixel format. If the number of aircraft is variable, the environment needs to support aircraft interaction between airspace or support the insertion of new aircraft entities. If only one aircraft is controlled, it has no impact on the action space. If all aircraft are controlled like the actual ATC process, the variable number of aircraft also has a great impact on the action space, and the multi-agent algorithm can be adopted. The variable number of aircraft complicates the design of the reward function, which requires the sub-functions and their weights to be optimized and normalized effectively.

The design of state space and algorithm needs to refer to the information displayed. For structured data input, the more information displayed, the more complex the state space, and more pre-processing is needed. If the graphical display mode is adopted, the state space is the raw pixel with the same dimension. Due to the different display modes and amount of information, the algorithm has to be designed or improved according to the actual requirement.

Instruction type refers to whether the instruction is discrete or continuous, and whether to adjust altitude, heading, speed, or others. The instruction type directly affects the design of the action space. The selection of basic algorithms depends on the continuity or discreteness of instructions. For example, DQN can be used for discrete instructions

and DDPG can be selected for continuous instructions. The structure and policy of the agent need to refer to the adjusted parameters. The objective of reward function design is to optimize the instructions output by the agent, so the reward function also needs to be designed according to the instruction type.

Decision mode refers to whether the instruction is sent in a fixed step or a non-fixed step. If the decision of non-fixed step size is adopted, the architecture of the DRL algorithm needs to be changed, such as [33]. The action space composed of basic actions is recommended in fixed step decision-making, while the action space composed of combined actions is used in non-fixed step decision-making. If the aircraft is manned, it is recommended to use the method of non-fixed step size, or use fixed step decision-making and optimize the smoothness of action through reward shaping, For UAVs, the decision-making method of fixed-step size can directly be used.

The evaluating indicators directly guide the design of the reward function, which is the content that must be referred to in the design of sub-functions. The training trajectory of an agent can be guided to obtain a higher evaluation through reward shaping. It is usually difficult to obtain an agent that meets the evaluation requirements through the parameter adjustment of a neural network, so the algorithm needs to be improved, including the improvement of agent structure and optimization of the training algorithm.

4.2. Open Issues of DRL in Conflict Resolution

DRL-based conflict resolution approaches have the following open issues.

1. Need a unified DRL framework. Some researchers have simplified the models in different directions according to their actual demands, and then many types of DRL models have been established. Some studies have also carried out personalized designs of state space or action space, which are inconsistent with the actual ATC scene, so they can not provide enough reference. Therefore, a challenge is that a DRL framework should be provided, and research should be carried out based on this to solve their respective problems.
2. Lack of a baseline. Compared with the fields with excellent DRL performance such as video games and robot control, the current application of DRL in the field of conflict resolution lacks a baseline. Firstly, an environment accepted by the research community is necessary. However, the environment is developed by each researcher independently and there is no unified version support. Secondly, there is also a lack of unified ATC scenarios. All studies design scenarios according to their practical problems and needs. Finally, the open-source baseline algorithm is also lacking. Of course, with the support of the environment and scenarios, the algorithm will be developed gradually.
3. New conflicts caused by multiple aircraft. For two aircraft, adjusting one or both aircraft can solve the conflict and will not bring new conflict. However, when the number of aircraft increases and the conflict between two aircraft is solved, the conflict between the original non-conflict aircraft pair may be caused. The current DRL conflict resolution strategies, including adjusting one aircraft or all aircraft, may bring new conflicts. This problem is mainly solved through the design of the reward function, which can not be solved perfectly.
4. Incomplete/uncertain information. Actual maneuver of the aircraft is always deviated from the controller's command due to some noise. However, in current researches, it is assumed that the information is complete and unbiased, which is inconsistent with that of the actual situation. How to capture the errors that exist in the real world and have an impact on the conflict resolution process and reflect them in the model is also one of the important challenges.
5. Look-forward time. In actual ATC, ATCos make predictions according to the flight plan, flight route, and current aircraft position. However, for the DRL model, there is no unified standard for the length of look-forward time. Look-forward for a long time, on the one hand, wastes time and consumes resources, on the other hand, it is

easy to generate a false alarm. Look-forward for a short time is not enough to detect potential conflicts. Therefore, adding an appropriate prediction of the future time into the model, whether autonomous or auxiliary ATCos decision-making, can be of great help.

6. Meteorological conditions are not considered. Meteorological conditions have a great impact on ATC. In bad weather, the aircraft needs to change the flight plan or route temporarily, which will bring potential conflicts. However, the influence of weather is not considered in current research. Therefore, it is necessary to consider the impact of meteorological conditions affecting flight and establish a model in the environment.

4.3. Future Research Suggestions

For future research, the authors have the following research directions and suggestions.

1. Representation of state and action space. In view of the inconsistency of MDP model in various studies, the representation methods of state space and action space need to be studied. By comparing the performance of agents with different state space and action space in typical ATC scenarios, effective combination points of actual conflict resolution and DRL model should be established.
2. Optimization of reward function. The agent learns according to the feedback of the reward function, and its role in DRL is equivalent to the label in supervised learning. Reward function is the main way to reflect the evaluation indicators of conflict resolution, including safety and efficiency. The agent should be guided to establish a better track by optimizing the reward function, so as to improve the performance.
3. Utilization of prior knowledge. In the short term, agents cannot work independently, and it is difficult to generate instructions beyond human controllers. Therefore, the prior knowledge of human controllers should be fully utilized to make the instructions of agents comply with the operation of ATCos and provide efficient auxiliary decision-making.
4. Combination with traditional approaches. The advantages of DRL are model-free and efficient decision-making. At present, its decision-making performance is not as good as traditional algorithms, such as geometric algorithms and optimization algorithms. Therefore, traditional algorithms can be combined with DRL to solve the problems that DRL is not good at.
5. Pre-training. Due to exploration, reinforcement learning often takes more time in the early phase of training. Pre-training can be carried out by using ATC data or in scenarios that are similar and easy to train. Retraining based on the pre-trained agent can significantly improve the training efficiency.
6. Improvement of evaluating indicators. The research on the evaluating indicator architecture of agents is also an important direction. Measure the safety and efficiency of conflict resolution and the evaluating indicators of DRL agents to establish a unified and reasonable evaluation system. On the one hand, it can form a more scientific evaluation, which is conducive to improvement. On the other hand, it can also guide the design of reward function.
7. Altitude adjustment agent. In a 3D scenario, both state space and action space are complex. Altitude adjustment is the most commonly used operation of ATCos, and its mechanism is different from that of adjustment in the horizontal plane. In the future, two agents can be trained, one to adjust the altitude and the other to adjust in the horizontal plane, and the coordination mechanism can be studied to use two agents together in practice.
8. Aircraft selection agent. For the scenario of multiple aircraft, it is necessary to select a conflicting aircraft for adjustment. However, the existing conflict resolution agents do not have this ability. An aircraft selection agent can be first trained, and be used to select one or more aircraft to be adjusted. Then the conflict resolution agent is adopted to generate maneuver decisions.

9. Returning to intention. There are flight routes in en-route mode, and pre-set routes and exit point in free flight mode. The current conflict resolution agent is only responsible for resolution while returning to intention still adopts the traditional method. Therefore, the research on the approach of intelligent return to intention is also one of the important research directions.
10. Distinguish between manned and unmanned aircraft. With the wide application of UAVs, the research on conflict resolution of UAVs is a hotspot. Different from manned aircraft, multi UAVs are mainly distributed, and the actions generated by an agent do not need to be smoothed.

5. Conclusions

DRL has already been applied to several cases in the context of conflict resolution. In the present article, we reviewed the available contributions on the topic to provide readers with a comprehensive state of play of DRL in conflict resolution. Details are provided on the environment, model, algorithm, and evaluating indicators. This review also lists the open issues of using DRL to solve the problem of conflict resolution, and gives future research directions and suggestions.

As of now, the capabilities and explainability of DRL algorithms in conflict resolution in actual ATC scenarios remain to be explored. Also, their behavior and convergence speed in complex decision space is unknown. However, it makes no doubt that the upcoming years will see the mastering of these obstacles, supported by the steady progress made in the DRL field. It is hoped that this review can inspire scholars and promote DRL to achieve fruitful research results in the field of conflict resolution in the future.

Author Contributions: Conceptualization, W.P. and Z.W.; validation, H.L. and Z.W.; formal analysis, H.L.; investigation, Z.W. and Q.Z.; writing—original draft preparation, Z.W.; writing—review and editing, W.P., Z.W.; visualization, X.W.; supervision, W.P.; project administration, X.W.; funding acquisition, W.P. and X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (U1733203) and Sichuan Science and Technology Program (2021YFS0319) and Scientific Research Fund Project of Civil Aviation Flight University of China (J2022-053).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Federal Aviation Administration. *FAA Aerospace Forecast: Fiscal Years 2020–2040*; U.S. Department of Transportation: Washington, DC, USA, 2020.
2. Kuchar, J.; Yang, L. A review of conflict detection and resolution modeling methods. *IEEE Trans. Intell. Transp. Syst.* **2000**, *1*, 179–189. [CrossRef]
3. Jenie, Y.I.; Van Kampen, E.J.; Ellerbroek, J.; Hoekstra, J.M. Taxonomy of conflict detection and resolution approaches for unmanned aerial vehicle in an integrated airspace. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1–10. [CrossRef]
4. Ribeiro, M.; Ellerbroek, J.; Hoekstra, J. Review of conflict resolution methods for manned and unmanned aviation. *Aerospace* **2020**, *7*, 79. [CrossRef]
5. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: London, UK, 2018.
6. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
7. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **2018**, *362*, 1140–1144. [CrossRef]
8. Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; Hutter, M. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **2019**, *4*, 1–20. [CrossRef]
9. Degrave, J.; Felici, F.; Buchli, J.; Neunert, M.; Tracey, B.; Carpanese, F.; Ewalds, T.; Hafner, R.; Abdolmaleki, A.; de Las Casas, D. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **2022**, *602*, 414–419. [CrossRef]

10. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjel, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
11. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
12. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
13. International Civil Aviation Association. *Doc 4444: Air Traffic Management - Procedures for Air Navigation Services*, 16th ed.; ICAO: Montreal, QC, Canada, 2016.
14. Watkins, C.J.; Dayan, P. Technical note: Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
15. Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-learning. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
16. Wang, Z.; Schaul, T.; Hessel, M.; van Hasselt, H.; Lanctot, M.; de Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
17. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014.
18. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
19. Busoniu, L.; Babuska, R.; Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. C Appl. Rev.* **2008**, *38*, 156–172. [CrossRef]
20. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *arXiv* **2018**, arXiv:1810.05587v3.
21. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
22. Yang, J.; Yin, D.; Xie, H. A reinforcement learning based UAVS air collision avoidance. In Proceedings of the 29th Congress of the International Council of the Aeronautical Sciences, St. Petersburg, Russia, 7–12 September 2014.
23. Regtuit, R.; Borst, C.; Van Kampen, E.J. Building strategic conformal automation for air traffic control using machine learning. In Proceedings of the 2018 AIAA Information Systems-AIAA Infotech @ Aerospace, Kissimmee, FA, USA, 8–12 January 2018.
24. Ribeiro, M.; Ellerbroek, J.; Hoekstra, J. Improvement of conflict detection and resolution at high densities through reinforcement learning. In Proceedings of the 9th International Conference on Research in Air Transportation, Virtual, 15 September 2020.
25. Hermans, M.C. Towards Explainable Automation for Air Traffic Control Using Deep Q-Learning from Demonstrations and Reward Decomposition. Master's Thesis, Delft University of Technology, Delft, The Netherlands, May 2020.
26. Brittain, M.; Wei, P. Autonomous aircraft sequencing and separation with hierarchical deep reinforcement learning. In Proceedings of the 8th International Conference on Research in Air Transportation, Barcelona, Spain, 26–29 June 2018.
27. Brittain, M.; Wei, P. Autonomous separation assurance in a high-density en route sector: A deep multi-agent reinforcement learning approach. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference, Auckland, New Zealand, 27–30 October 2019.
28. Brittain, M.; Wei, P. One to any: Distributed conflict resolution with deep multi-agent reinforcement learning and long short-term memory. In Proceedings of the 2021 AIAA Science and Technology Forum and Exposition, Nashville, TN, USA, 11–15 January 2021.
29. Brittain, M.; Yang, X.; Wei, P. A deep multi-agent reinforcement learning approach to autonomous separation assurance. *arXiv* **2020**, arXiv:2003.08353.
30. Guo, W.; Brittain, M.; Wei, P. Safety enhancement for deep reinforcement learning in autonomous separation assurance. *arXiv* **2021**, arXiv:2105.02331.
31. Pham, D.T.; Tran, N.P.; Goh, S.K.; Alam, S.; Duong, V. Reinforcement learning for two-aircraft conflict resolution in the presence of uncertainty. In Proceedings of the 2019 IEEE-RIVF International Conference on Computing and Communication Technologies, Danang, Vietnam, 20–22 March 2019.
32. Tran, N.P.; Pham, D.T.; Goh, S.K.; Alam, S.; Duong, V. An intelligent interactive conflict solver incorporating air traffic controllers' preferences using reinforcement learning. In Proceedings of the 2019 Integrated Communications, Navigation and Surveillance Conference, Herndon, VA, USA, 9–11 April 2019.
33. Wang, Z.; Li, H.; Wang, J.; Shen, F. Deep reinforcement learning based conflict detection and resolution in air traffic control. *IET Intell. Transp. Syst.* **2019**, *13*, 1041–1047. [CrossRef]
34. Zhao, P.; Liu, Y. Physics informed deep reinforcement learning for aircraft conflict resolution. *IEEE Trans. Intell. Transp. Syst.* **2021**, *1*, 1–14. [CrossRef]
35. Sui, D.; Xu, W.; Zhang, K. Study on the resolution of multi-aircraft flight conflicts based on an IDQN. *Chin. J. Aeronaut.* **2021**, *35*, 195–213. [CrossRef]
36. Wen, H.; Li, H.; Wang, Z. Application of DDPG-based collision avoidance algorithm in air traffic control. In Proceedings of the 12nd International Symposium on Computational Intelligence and Design, Hangzhou, China, 14–15 December 2019.

37. Li, S.; Egorov, H.; Kochenderfer, M.J. Optimizing collision avoidance in dense airspace using deep reinforcement learning. In Proceedings of the 13th USA/Europe Air Traffic Management Research and Development Seminar, Vienna, Austria, 17–21 June 2019.
38. Mollinga, J.; Hoof, H. An autonomous free airspace en-route controller using deep reinforcement learning techniques. In Proceedings of the 9th International Conference on Research in Air Transportation, Virtual, 15 September 2020.
39. Dalmau, R.; Allard, E. Air traffic control using message passing neural networks and multi-agent reinforcement learning. In Proceedings of the 10th SESAR Innovation Days, Budapest, Hungary, 7–10 December 2020.
40. Ghosh, S.; Laguna, S.; Lim, S.H.; Wynter, L.; Poonawala, H. A deep ensemble method for multi-agent reinforcement learning: A case study on air traffic control. In Proceedings of the 31st International Conference on Automated Planning and Scheduling, Guangzhou, China, 2–13 August 2021.
41. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.* **2013**, *47*, 253–279. [CrossRef]
42. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A physics engine for model-based control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 December 2012.
43. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540
44. Kelly, S. Basic introduction to PyGame. In *Python, PyGame and Raspberry Pi Game Development*; Apress: Berkeley, CA, USA, 2019; pp. 87–97.
45. Flight Control Exercise. Available online: https://github.com/devries/flight-control-exercise (accessed on 26 March 2022).
46. ELSA Air Traffic Simulator. Available online: https://github.com/ELSA-Project/ELSA-ABM (accessed on 26 March 2022).
47. Hoekstra, J.; Ellerbroek, J. BlueSky ATC simulator project: An open data and open source approach. In Proceedings of the 7th International Conference on Research in Air Transportation, Philadelphia, PA, USA, 20–24 June 2016.
48. BlueSky-The Open Air Traffic Simulator. Available online: https://github.com/TUDelft-CNS-ATM/bluesky (accessed on 26 March 2022).
49. Ng, A.; Harada, D.; Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In Proceedings of the 16th International Conference on Machine Learning, Bled, Slovenia, 27–30 June 1999.
50. Kanervisto, A.; Scheller, C.; Hautamäki, V. Action space shaping in deep reinforcement learning. In Proceedings of the 2020 IEEE Conference on Games, Osaka, Japan, 24–27 October 2020; pp. 479–486.
51. Hermes, P.; Mulder, M.; Paassen, M.M.; Boering, J.H.L.; Huisman, H. Solution-space-based complexity analysis of the difficulty of aircraft merging tasks. *J. Aircr.* **2009**, *46*, 1995–2015. [CrossRef]
52. Ellerbroek, J.; Brantegem, K.C.R.; Paassen, M.M.; Mulder, M. Design of a coplanar airborne separation display. *IEEE Trans. Hum. Mach. Syst.* **2013**, *43*, 277–289. [CrossRef]
53. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
54. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 34th Advances in Neural Information Processing Systems, Long Beach, CA, USA, 6–12 December 2017.
55. Wu, Z.; Pan, S.; Chen, J.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
56. Gilmer, J.; Schoenholz, S.S.; Patrick, S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
57. Ormoneit, D.; Sen, Ś. Kernel-based reinforcement learning. *Mach. Learn.* **2002**, *49*, 161–178. [CrossRef]
58. Bouton, M.; Julian, K.; Nakhaei, A.; Fujimura, K.; Kochenderfer, M.J. Utility decomposition with deep corrections for scalable planning under uncertainty. In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 10–15 July 2018.
59. Hoekstra, J.; Gent, R.; Ruigrok, R. Designing for safety: The 'free flight' air traffic management concept. *Reliab. Eng. Syst. Saf.* **2002**, *75*, 215–232. [CrossRef]
60. Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Osband, I.; et al. Deep Q-learning from demonstrations. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–9 February 2018.
61. Hong, Y.; Kim, Y.; Lee, K. Application of complexity map to reduce air traffic complexity in a sector. In Proceedings of the 2014 AIAA Guidance, Navigation, and Control Conference, National Harbor, MD, USA, 13–17 January 2014.