# Locally generalised multi-agent reinforcement learning for demand and capacity balancing with customised neural networks

**Yutong CHEN** [a,b], **Minghua HU** [a], **Yan XU** [b,*], **Lei YANG** [a]

[a] *College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing, 210000, China*
[b] *School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, MK43 0AL, United Kingdom*

**KEYWORDS**

Demand and capacity balancing;
Air traffic flow management;
Multi-agent reinforcement learning;
Deep Q-learning network;
Ground delay program;
Flight delays;
Generalisation

**Abstract** Reinforcement Learning (RL) techniques are being studied to solve the Demand and Capacity Balancing (DCB) problems to fully exploit their computational performance. A locally generalised Multi-Agent Reinforcement Learning (MARL) for real-world DCB problems is proposed. The proposed method can deploy trained agents directly to unseen scenarios in a specific Air Traffic Flow Management (ATFM) region to quickly obtain a satisfactory solution. In this method, agents of all flights in a scenario form a multi-agent decision-making system based on partial observation. The trained agent with the customised neural network can be deployed directly on the corresponding flight, allowing it to solve the DCB problem jointly. A cooperation coefficient is introduced in the reward function, which is used to adjust the agent's cooperation preference in a multi-agent system, thereby controlling the distribution of flight delay time allocation. A multi-iteration mechanism is designed for the DCB decision-making framework to deal with problems arising from non-stationarity in MARL and to ensure that all hotspots are eliminated. Experiments based on large-scale high-complexity real-world scenarios are conducted to verify the effectiveness and efficiency of the method. From a statistical point of view, it is proven that the proposed method is generalised within the scope of the flights and sectors of interest, and its optimisation performance outperforms the standard computer-assisted slot allocation and state-of-the-art RL-based DCB methods. The sensitivity analysis preliminarily reveals the effect of the cooperation coefficient on delay time allocation.

## 1. Introduction

Recently, one of the major concerns in the global development of civil aviation is the growing imbalance between increasing total traffic volume and saturated airspace accommodation, also known as the demand-capacity mismatch. If demand exceeds capacity in a sector for a certain period, hotspots will arise, resulting in increased loads on controllers, congestion in airspace, and flight delays [1]. As a result, balancing demand and capacity has become a vital issue for the aviation industry. DCB is one of the seven operational concepts of Air Traffic Management (ATM) [2]. According to Single European Sky ATM Research (SESAR), DCB will play an essential role in the future air traffic management system as part of network management and can help to reduce flight delays [3,4].

DCB is also known as ATFM or Air Traffic Flow and Capacity Management (ATFCM) [5,6]. Depending on the advance time of operation, ATFM is divided into strategic (one year to one week), pre-tactical (one week to one day), and tactical (day of operation) [7]. The main focus of this paper is on ATFM implemented on the Day before (D-1) or on the Day (D-0) of operation. The typical operational ways of ATFM contain Ground Delay Program (GDP) [8,9], rerouting [10,11], separation management [12,13] and their combination [14,15].

ATFM methods are classified into two types based on their solution methods: exact solution methods [16,17] and approximate solution methods [18,19]. In general, the advantage of exact solution methods is obtaining a globally optimal solution. However, when the problem is too large, such methods cannot guarantee that the solution will be completed in a limited time. Besides, the computing time highly depends on cases and can vary considerably for DCB problems of similar problem scales [16,17]. As a result, exact solution methods are hardly ever applied in practice. On the other hand, Computer-Assisted Slot Allocation (CASA), an approximate algorithm, is usually used in practice. CASA is commonly used in Europe, and it is similar to the Ration By Schedule (RBS) approach as applied in the United States. Approximation solution methods

*Corresponding author
Email addresses: chenyutong@nuaa.edu.cn (Yutong CHEN), minghuahu@nuaa.edu.cn (Minghua HU), yanxu@cranfield.ac.uk (Yan XU ), laneyoung@nuaa.edu.cn (Lei YANG)

typically employ some heuristic framework or algorithm to find a locally optimal solution in a reasonable amount of time. The computation time of approximation solution methods is less sensitive to problem scale than exact solution methods. However, local optimal solutions are often not readily accepted because there is frequently a significant gap between the local and the global optimal solution. Thus, a DCB method capable of obtaining solutions with high optimisation performance in a short time is highly desired.

In recent years, reinforcement learning techniques have gradually been tried to solve DCB problems to find a good balance between computing speed and optimisation performance. RL methods train agents to obtain strategies through a large number of training scenarios and then deploy trained agents to an actual scenario problem to make quick decisions so that the solution can be obtained in a short time. It is equivalent to transferring a significant amount of solution time to the training stage, allowing it to respond quickly to actual scenarios in operation. Hence, RL-based methods have the advantage of being faster in the calculation compared with exact solution methods. Moreover, compared with approximation solution methods, RL-based methods have potential to obtain a better approximate solution. It is because approximate solution methods such as CASA are rule-based algorithms designed based on human experience, which is likely to limit optimality. RL methods were initially explored in the field of DCB. However, up to our best knowledge, it is challenging for existing RL-based DCB methods to solve the DCB problem with large-scale high-complexity real-world scenarios in a short time by directly deploying trained agents. For some existing RL-based DCB methods, it is impossible to deploy trained agents to unseen scenarios because of the model's scalability. For others, it is not trivial to achieve satisfactory solutions in unseen scenarios because of model design structure (please refer to Table 1 and corresponding discussions for details). Therefore, these methods must retrain the agent if they intend to effectively solve the DCB problem for an unseen scenario. However, training the agent is time-consuming, which offsets the advantage of RL methods.

Therefore, we propose a locally generalised MARL for real-world DCB problems to fill the gap. Our method can deploy trained agents directly to unseen scenarios in a specific ATFM region to obtain a satisfactory solution quickly (where 'locally' corresponds to 'a specific ATFM region'). In this method, a distributed decision system is constructed by deploying a customised neural network on each flight to handle the specificity of each flight in DCB problems. The cooperation coefficient is introduced to control the degree of cooperation between flights. A multi-iteration mechanism is designed to deal with problems arising from non-stationarity in MARL.

This paper is organised as follows. In the rest of Section 1, we provide a brief review of related work and introduce features and contributions of our method. Section 2 introduces notations and formulates the DCB problem. In Section 3, we discuss the construction of the reinforcement learning environment, the architecture of the network, the training method of the neural network, and the multi-iteration mechanism. In Section 4, we show the MARL training process, the performance test results, and the cooperation coefficient sensitivity analysis through simulation experiments. Finally, conclusions and future work are summarised in Section 5.

### *1.1. Related work*

Much research has been done on DCB problems, and some review articles provide good summaries of the current research progress [20–22]. RL methods were explored in various fields of ATM because they can solve a problem in a short time after completing training, such as conflict resolution [23–25], collision avoidance [26–28] and assistant control tool [29–31]. Due to the advantage of responding quickly in real scenarios, RL methods have excellent research value and potential for application in real-world DCB problems. Several representative RL-based DCB methods are summarised in Table 1, and the relevant explanations for this table are given as follows:

(1) Agent: Artificial intelligence with decision-making ability.
   - Number: The number of agents in the system, where S and M refer to single and multiple, respectively.
   - Role: The agent's role in the DCB problem, where C and F refer to controllers and flights, respectively.
   - Mode: The operating mode of agents, where C and D refer to centralized and decentralized (distributed), respectively.

(2) RL method: The method used to train the agent's policy, e.g., Q-table, Temporal-Difference (TD) learning, Q-learning, Deep Q-Learning (DQN), Proximal Policy Optimisation (PPO) and Asynchronous Advantage Actor-Critic (A3C).

(3) Sharing policy: Whether agents share the neural network parameters in the multi-agent system.

(4) Generalisation: Whether the method is generalised in a specific ATFM region. L1 means that the trained agent can be deployed directly to unseen scenarios (the model is scalable), but there is no guarantee that a satisfactory solution will be obtained. L2 means that the trained agent can obtain a satisfactory solution based on L1.

(5) ATFM method: Operational ways of ATFM, where GDP refers to ground delay program, and MIT refers to Miles-in-trail (a kind of separation management).

(6) Sector opening scheme: Whether the sector structure changes over time, as in the real world.

(7) Uncertainty: Whether to consider the uncertainty of demand and capacity forecasts and the uncertainty of flight.

(8) Elimination: Whether all hotspots can be guaranteed to be eliminated.

(9) Experimental scenario: The most complex and realistic experimental scenario in the study.
   - Real-world: Whether the experiment was based on real-world data, including flights and sectors.
   - Hotspot: The initial number of hotspots.
   - Flight scale: The number of flights (round by 100).
   - Sector scale: The number of sectors.

(10) Symbol descriptions: checkmark (✓) and circle (○) respectively mean that the method has and does not has the corresponding feature. N/A means that the feature or parameter is not applicable to the method or is not disclosed in the study.

Please note that if there are several extended methods (or variants of the basic method) introduced in a study, only the one with the highest comprehensive performance is shown in the table.

Crespo et al. [32] trained a single agent through RL to centrally assign flight delays to several airports (all flights at the same airport are delayed by the same time). Due to the limitation of the Q table, this agent can only be deployed for problems with specific sectors and airports. Agogino and Tumer [33] deployed agents on several route nodes around overloaded sectors, forming a distributed multi-agent system and employing the Miles-in-trail (MIT) method to adjust the distance between flights passing through the same node for ATFM purposes. Kravaris et al. [34] proposed a collaborative RL framework for DCB, which deploys an agent on each flight.

**Table 1** Features of DCB methods based on RL[32–42]

| No. | Study | Agent | | | RL method | Sharing policy | Generalisation | ATFM method | Sector opening scheme | Uncertainty | Elimination | Experimental scenario | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Number | Role | Mode | | | | | | | | Real world | Hotspot | Flight scale ($10^3$) | Sector scale |
| 1 | Crespo[32] | S | C | C | Q-table | N/A | N/A | GDP | ○ | ○ | ○ | ✓ | N/A | N/A | 10 |
| 2 | Agogino[33] | M | C | D | TD learning | ○ | N/A | MIT | ○ | ○ | ○ | ✓ | N/A | 1.3 | N/A |
| 3 | Kravaris[34] | M | F | D | Q-learning | ○ | N/A | GDP | ○ | ○ | ○ | ○ | N/A | 1 | 16 |
| 4 | Spatharis[35,36] | M | F | D | Q-learning | ○ | N/A | GDP | ✓ | ○ | ○ | ✓ | 53 | 6 | 169 |
| 5 | Duong[37] | M | C | D | Q-learning | ○ | N/A | GDP | ○ | ○ | ○ | ✓ | N/A | 0.4 | N/A |
| 6 | Spatharis[38,39] | M | F | D | Q-learning | ○ | N/A | GDP | ✓ | ○ | ○ | ✓ | 53 | 6 | N/A |
| 7 | Chen[40] | M | F | D | DQN | ○ | N/A | GDP | ○ | ○ | ○ | ○ | 54 | 3 | 16 |
| 8 | Tang[41] | M | F | D | PPO | ✓ | L1 | GDP | ○ | ○ | ✓ | ✓ | 31 | 8.2 | 356 |
| 9 | Huang[42] | M | F | D | A3C | ✓ | L1 | GDP | ○ | ○ | ✓ | ✓ | 31 | 8.2 | 356 |
| 10 | This paper | M | F | D | DQN | ○ | L2 | GDP | ✓ | ○ | ✓ | ✓ | 186 | 12 | 396 |

In Kravaris' method, each flight makes a decision independently without communicating with other flights. However, the effectiveness of this method has only been demonstrated in one tiny toy case. Spatharis et al.[35,36] further enhanced Kravaris' method and verified their method's effectiveness in real-world scenarios with high complexity. Duong et al.[37] presented a blockchain-based RL method for ATFM. Like Crespo's method, the agent in Duong's method plays a controller to assign delays to flights at its responding airport, and each agent is responsible for only one airport. Spatharis et al.[38,39] proposed and extended a hierarchical MARL learning scheme for DCB problems, which constructed two states for agents: the ground state and abstract state. A common shortcoming of the six studies mentioned above is that they all employed RL algorithms as a search algorithm; that is, the process of training is used as the process of solving. Therefore, these methods are not generalised, and the advantages of reinforcement learning in terms of rapid response are not available for the above methods. Besides, because of the non-stationarity in MARL, they cannot guarantee that the agents will always be able to eliminate all hotspots. Thanks to the development of deep neural networks in reinforcement learning applications, the DCB methods with deep neural networks have been further enhanced. Chen et al.[40] validated the effectiveness of the Deep Q-learning Network (DQN) in the RL-based DCB method. However, the experiments are not based on real-world scenarios, and the generalisation of the method has not been verified. Tang and Xu[41] integrated an action supervisor into a rule-based time-step DCB method using Proximal Policy Optimisation (PPO). Tang's method forces the agent to change its action when it chooses an action that would cause a hotspot to ensure that hotspots are eliminated. However, the addition of an action supervisor has led to a significant increase in delays. Huang and Xu[42] presented a hotspot-based DCB method with Asynchronous Advantage Actor-Critic (A3C), and this method ensures that hotspots are eliminated by allocating multiple delays. Despite Tang's and Huang's methods attempting to deploy trained agents to unseen scenarios, their method cannot be considered highly generalised due to potential flaws in the model design. For Tang's method, the observation matrix seems so sparse that it limits agent learning. For Huang's method, sector IDs are used as part of the agent observations and the ID as a code is not computationally meaningful.

In summary, it is difficult to deploy agents trained by these existing methods in Table 1 directly to an unseen scenario and obtain a satisfactory solution quickly. Therefore, there is an urgent need to improve the generalisation of RL-based DCB methods.

### *1.2. Proposed method*

This paper serves as an extension of our previous study[40]. In this paper, a locally generalised MARL method for DCB where each agent has a customised neural network is proposed (the features of our method are also summarised in Table 1).

Generalisation is one of the most critical indicators of RL methods. To our knowledge, this is a dilemma for the DCB problem, however. On the one hand, the scale of each DCB problem (the number of flights or sectors) is different, while the dimension of the observation matrix in an RL method is usually required to be fixed. On the other hand, maybe we could technically remove the limitation of the observation matrix. However, completely homogenising individuals in large-scale DCB problems could significantly decrease the solver's optimisation performance and make it difficult to meet the differentiated preferences of different flights. Hence, a balance needs to be found in this dilemma to maximise the advantages of the RL method.

Considering that flight schedules for commercial flights are usually cyclical, we can train an agent with a customised neural network for each flight schedule and deploy it in any scenario that contains that flight. Our method has both a local generalisation and the optimisation performance improvement led by heterogeneous individuals in the multi-agent system. We set the cooperation coefficient in the reward function to adjust the cooperation preference of the flight, thereby adjusting the distribution of the global delay time allocation. Our neural network outputs are only two discrete actions. We employ the state-of-the-art algorithm based on DQN, Rainbow DQN[43], which significantly improves RL efficiency by integrating multiple DQN enhancement technologies. Besides, we design a multi-iteration mechanism for the DCB decision-making framework to deal with problems arising from non-stationarity in MARL, thereby enabling the solver to eliminate hotspots.

### *1.3. Summary of contributions*

(1) Trained agents of the proposed method can be deployed directly to unseen scenarios in a specific ATFM region to obtain a satisfactory solution quickly.
(2) A cooperation coefficient is introduced to adjust the distribution of flight delay time allocation.
(3) A multi-iteration mechanism is designed for the DCB decision-making framework to enable the solver to eliminate all hotspots.
(4) Systematic experiments based on large-scale high-complexity real-world scenarios are performed to verify the proposed method's effectiveness and efficiency.

## 2. Problem formulation

The demand and capacity balancing problem to be handled in this article is ensuring that the number of flights entering the sector per unit time does not exceed its capacity (that is, all hotspots are eliminated) by implementing the ground delay program before the operation.

### 2.1. Demand and capacity

All initial flight schedules can be obtained one day before the flight operation. The flight schedule of the $i$th flight is denoted by $f_i$, which consists of a set of binary arrays, as shown in Fig. 1:

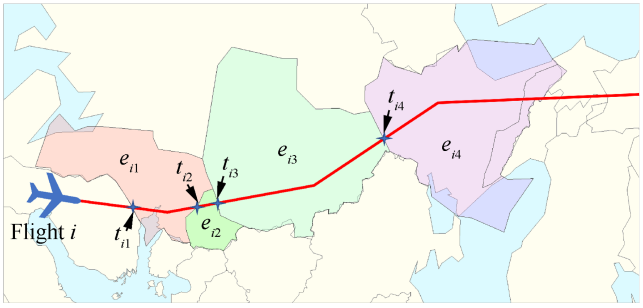



**Fig. 1** Flight schedule

$$f_i = \left\{ \left( e_{ij}, t_{ij} \right) | j = 1, 2, ..., M_i \right\} \quad i \in I \tag{1}$$

where $e_{ij}$ denotes the $j$th sector through which the $i$th flight is scheduled to pass, $t_{ij}$ denotes the time of entry into that sector, $M_i$ denotes the number of sectors through which the $i$th flight is scheduled to pass, and $I$ denotes the set of flights. The set of initial flight schedules $F^{\text{Initial}}$ is represented as

$$F^{\text{Initial}} = \{ f_i | i \in I \} \tag{2}$$

To simplify the problem, we only consider the current day's flight segments and use the next day's flight segments as input to the DCB problem the following day. Every sector has its specific capacity, which changes over time, and the update circle is $\tau$ (typically 20 min). As a result, one day is divided into $N^{\text{T}}$ time windows of $\tau$ width ($N^{\text{T}} \times \tau = 24$ hours), with $C_{jt}$ denoting the capacity of the $j$th sector in the $t$th time window. $W_t$ denotes the time range of the $t$th time window, where $T$ is the set of time windows and $T = \{1, 2, ..., N^{\text{T}}\}$:

$$W_t = [(t-1)\tau, t\tau) \quad t \in T \tag{3}$$

For example, if the width of time windows $\tau$ is 20 min, the time range of the 10th time window is $[180, 200)$, which corresponds to 03:00–03:20 of the operation day (not including 03:20).

In practice, sectors are divided into two types, elementary sectors and collapsed sectors. An elementary sector is the most fundamental sector unit in the airspace, while a collapsed sector comprises several adjacent elementary sectors. The basic sector units operate as elementary sectors or collapsed sectors depending on the sector opening scheme. In this paper, the sector opening scheme is considered, and the state of the sectors changes over time windows.

For example, in a sector opening scheme as shown in Fig. 2, there are four elementary sectors ($e_1$, $e_2$, $e_3$ and $e_4$) and two collapsed sectors ($e_5$ and $e_6$). $e_5$ is made up of $e_2$ and $e_3$. $e_6$ is made up of $e_3$ and $e_4$. If the $j$th sector is closed in the $t$th time window, $C_{jt}$ equals 0. The $i$th flight is scheduled to

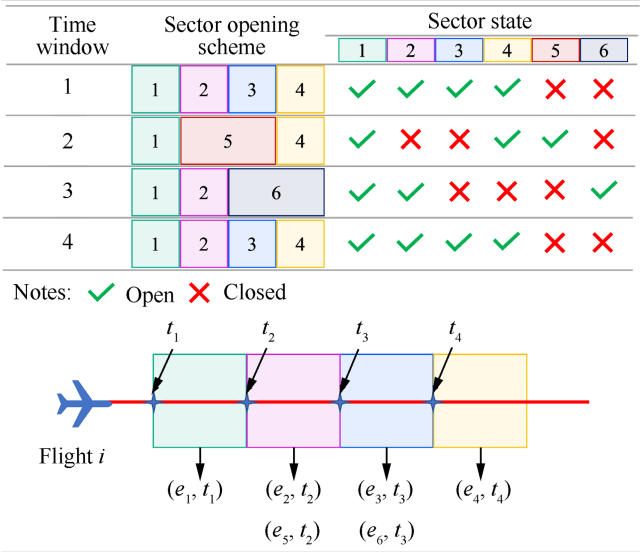



**Fig. 2** Sector opening scheme

pass through the four elementary sectors in turn, and their entry times, respectively, are $t_1$, $t_2$, $t_3$ and $t_4$. $f_i$ will be represented as $\{(e_1, t_1), (e_2, t_2), (e_5, t_2), (e_3, t_3), (e_6, t_3), (e_4, t_4)\}$. $D_{jt}$ denotes the demand of the $j$th sector in the $t$th time window. It is defined as the number of flights entering the $j$th sector in the $t$th time window. Please note that when calculating $D_{jt}$, only the time the flight enters the sector is considered, regardless of how long it will be in that sector. For example, if the $i$th flight is scheduled to enter the $j$th sector in the $t$th time window and leave in the $(t+2)$th time window, it is only taken as the demand of the $j$th sector in the $t$th time window, but not the $(t+1)$th and $(t+2)$th time windows. If the $j$th sector is closed in the $t$th time window, both $C_{jt}$ and $D_{jt}$ equal 0. $S$ denotes the sector opening scheme, where $J$ denotes the set of sectors:

$$S = \left\{ C_{jt} | j \in J, t \in T \right\} \tag{4}$$

In summary, the DCB problem in this paper is defined as balancing the traffic demand with the airspace capacity through shifting flight take-off time to avoid hotspots. The sector opening scheme is considered. The optimisation objective is to minimise the total delay time for all flights.

### 2.2. Partially observable decision-making

We propose a multi-agent decision-making framework to tackle the DCB problem through the MARL method, as shown in Fig. 3. All flights are divided into $N^T$ groups according to the time window to which their scheduled departure time belongs. Each flight will be decided to depart or hold in this time window from the first time window by the agent deployed on the flight. In each time window, the agents are distributed. If the flight is decided to hold, it will be delayed for $\tau$, moving to the next time window. Time windows are operated sequentially until all flights get their departure time or are delayed until the next day. The DCB problem is formulated as a partially observable decision-making problem in our method. A flight in the current time window $t$ has access to an observation $\boldsymbol{o}_t^i$, which only provides partial knowledge about the environment and computes a decision-making action $a_t^i$ that tries to avoid hotspots.

In our formulation, the observation $\boldsymbol{o}_t^i$ is drawn from the global state $\boldsymbol{s}_t^i$ ($\boldsymbol{o}_t^i \sim O\left(\boldsymbol{s}_t^i\right)$), and it only provides partial information about the state $\boldsymbol{s}_t^i$. It is easy to obtain the global state $\boldsymbol{o}_t^i$ in our model, but

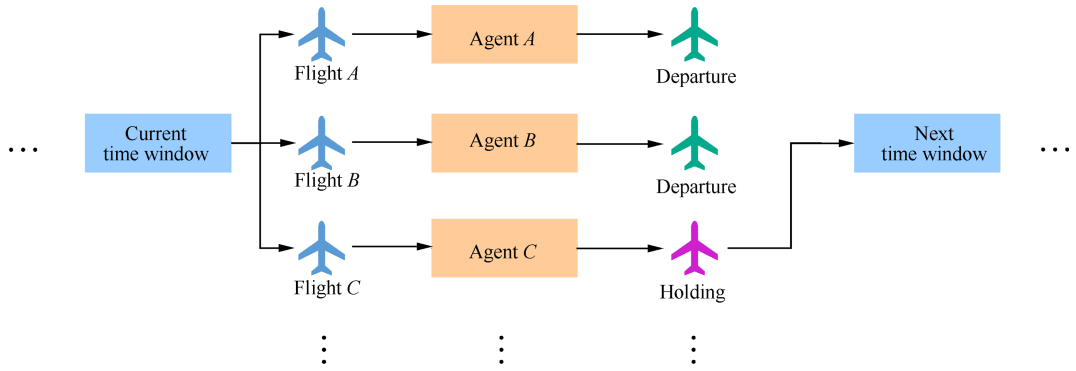



**Fig. 3** Multi-agent decision-making framework (top-level framework of the proposed method) where all agents are distributed.

the dimensions of the global state are too large. For example, the global state $\boldsymbol{o}_t^i$ is represented by an $M \times N$ matrix; $M$ and $N$ denote the number of sectors and flights, respectively; the elements in the matrix are $t_{ij}$. If so, for example, in a scenario with 10000 flights and 400 sectors, the dimensions of the global state are $10000 \times 400$, that is, $4 \times 10^6$ elements are contained. Technically, convolutional neural networks can reduce the dimensionality of the global state. However, we found by experiments that the optimisation results were not satisfactory if using convolutional neural networks. It may be due to the loss of crucial information in the global state during the convolution process, which severely limits decision optimisation. Hence, partial observation is applied in our method to keep the dimensions of observation in an acceptable range.

When flights are in the current time window, given the partial observation $\boldsymbol{o}_t^i$, each flight independently computes a decision-making action $a_t^i$, sampled from its specific policy $\boldsymbol{\pi}_{\theta_i}^i$, where $\theta_i$ denotes the policy parameters:

$$a_t^i \sim \boldsymbol{\pi}_{\theta_i}^i \left( a_t^i \middle| \boldsymbol{o}_t^i \right) \tag{5}$$

$\boldsymbol{\Pi}_\Theta$ denotes the set of policies, where $\Theta$ is the parameter:

$$\boldsymbol{\Pi}_\Theta = \left\{ \boldsymbol{\pi}_{\theta_i}^i \,|i \in I \right\} \tag{6}$$

We can take $\boldsymbol{\Pi}_\Theta$ as a DCB solver, and $\boldsymbol{\Pi}_\theta \left( F^{\text{Initial}} | S \right)$ denotes the optimal solution of the solver for the DCB problem with the initial flight schedules $F^{\text{Initial}}$ and the sector opening scheme $S$, where $\Delta t_i$ denotes how long the $i$th flight is expected to be delayed in its departure airport:

$$\boldsymbol{\Pi}_\Theta \left( F^{\text{Initial}} | S \right) = \{ \Delta t_i \,|i \in I \} \tag{7}$$

$\Delta t_i = k\tau$, where $k$ is a non-negative integer in our model. $f_i\left(\Delta t_i\right)$ denotes the flight schedule of the $i$th flight with the implementation of GDP:

$$f_i\left(\Delta t_i\right) = \left\{ \left( e_{ij}, t_{ij} + \Delta t_i \right) | j \in J_i \right\} \tag{8}$$

The set of flight schedules that meets DCB requirements can be represented as

$$F = \left\{ f_i(\Delta t_i) \quad i \in I \middle| \begin{array}{l} \Delta t_i \sim \boldsymbol{\Pi}_\Theta \left( F^{\text{Initial}} | S \right) \quad j \in J, t \in T : \\ D_{jt} \leq C_{jt} \end{array} \right\} \tag{9}$$

To find an optimal policy, we adopt an objective by minimizing the expectation of the average delay time of all flights in the same scenario, which is defined as

$$\arg\min_{\boldsymbol{\Pi}_\Theta} E \left[ \frac{1}{N} \sum_{i \in I} \Delta t_i \,|\boldsymbol{\Pi}_\Theta \right] \tag{10}$$

where $N$ is the number of flights. The average delay time will also be an essential metric to evaluate the trained policy in Section 4.

## 3. Approach

In this section, we first introduce the key ingredient of our reinforcement learning setup. Then, the detail of the architecture of the policy network based on a neural network is proposed. Next, we present the MARL training algorithm based on Rainbow DQN and the design of training scenarios. Finally, we discuss the multi-iteration mechanism.

### *3.1. Reinforcement learning setup*

We assume all flights in the environment are heterogeneous to tackle the DCB problem through the MARL method. Each flight has different aircraft types, routes, and preferences. More importantly, most commercial flight plans are generally repeated every week, and there is a limited amount of flight routes in the real world. Thus, we treat all flights over time as a set of flights based on historical data and train an agent for each flight using reinforcement learning methods. If so, we can solve DCB problems based on a subset of this set of flights by deploying corresponding agents to flights. Based on the problem formulation in Section 2.2, the DCB problem can be transferred into a Partially Observable Markov Decision Process (POMDP) solved with MARL. A POMDP can be described as a six-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \boldsymbol{\Omega}, O)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state-transition model, $\mathcal{R}$ is the reward function, $\boldsymbol{\Omega}$ is the observation space ($\boldsymbol{o} \in \boldsymbol{\Omega}$), and $O$ is the observation probability distribution given the system state ($\boldsymbol{o} \sim O(\boldsymbol{s})$), where $\boldsymbol{s}$ is the global state. An agent is deployed on each aircraft in this environment, and all the agents form a distributed decision-making system. Therefore, a multi-aircraft state-transition model $\mathcal{P}$ determined by the aircraft's delay time and uncertainty is unnecessary. The action space $\mathcal{A}$, the observation space $\boldsymbol{\Omega}$ and the reward function $\mathcal{R}$ are defined as below.

#### *3.1.1. Action space*

As mentioned in Section 2.2, there are two actions for agent $i$ to choose; namely, $a_i^t \in \{0, 1\}$. When $a_i^t = 0$, it means to hold; when $a_i^t = 1$, it means to choose to take off. If the agent chooses to hold

on the ground, it will be delayed by a time window width time ($\tau$) moving to the next time window; if the agent chooses departure, its departure time will be determined as the currently scheduled departure time.

*3.1.2. State space*

In our model, the observation $\boldsymbol{o}_t^i$ consists of the current state of the sectors that the flight $i$ is scheduled to pass through $\boldsymbol{\Phi}_t^i$, the current time window $t$ and the number of holdings that the flight has performed $h_t^i$:

$$\boldsymbol{o}_t^i = \left[\boldsymbol{\Phi}_t^i, t, h_t^i\right] \tag{11}$$

The current state of the sectors $\boldsymbol{\Phi}_t^i$ contains all the sectors that the flight $i$ is scheduled to pass through:

$$\boldsymbol{\Phi}_t^i = \left\{\boldsymbol{\phi}_{j\chi_j^{it}} \,|j \in J_i\right\} \tag{12}$$

where $\boldsymbol{\phi}_{j\chi_j^{it}}$ denotes the state of sector $j$ in time window $\chi_j^{it}$, and $\chi_j^{it}$ represents the time window when flight $i$ will enter sector $j$ if it takes off in time window $t$. $\boldsymbol{\phi}_{j\chi_j^{it}}$ consists of the remaining capacity $c_{j\chi_j^{it}}^{\text{Remaining}}$ and potential demand $d_{j\chi_j^{it}}^{\text{Potential}}$ of sector $j$ in time window $\chi_j^{it}$:

$$\boldsymbol{\phi}_{j\chi_j^{it}} = \left[c_{j\chi_j^{it}}^{\text{Remaining}}, d_{j\chi_j^{it}}^{\text{Potential}}\right] \tag{13}$$

$c_{j\chi_j^{it}}^{\text{Remaining}}$ is represented as:

$$c_{j\chi_j^{it}}^{\text{Remaining}} = \begin{cases} C_{j\chi_j^{it}} - \sum\limits_{i' \in I_{t-1}^{\text{Departure}}} \psi_{j\chi_j^{it}}^{i'} & C_{j\chi_j^{it}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where $C_{j\chi_j^{it}}$ is the capacity of sector $j$ in time window $j\chi_j^{it}$, $I_{t-1}^{\text{Departure}}$ is the set of flights departing before or in time window $(t-1)$ and $\psi_{j\chi_j^{it}}^{i'}$ is a binary parameter. If flight $i'$ enters sector $j$ in time window $\chi_j^{it}$, then $\psi_{j\chi_j^{it}}^{i'} = 1$; otherwise, $\psi_{j\chi_j^{it}}^{i'} = 0$. Therefore, $c_{j\chi_j^{it}}^{\text{Remaining}}$ refers to the remaining capacity based on the demand that has been identified so far. If sector $j$ is not open in time window $\chi_j^{it}$ (that is, $C_{j\chi_j^{it}} = 0$), the remaining capacity $c_{j\chi_j^{it}}^{\text{Remaining}}$ will be set to zero. $d_{j\chi_j^{it}}^{\text{Potential}}$ is represented as

$$d_{j\chi_j^{it}}^{\text{Potential}} = \begin{cases} \sum\limits_{i' \in I_t} \zeta_{j\chi_j^{it}}^{i'} & C_{j\chi_j^{it}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $I_t$ is the set of flights in time window $t$ currently, and $\zeta_{j\chi_j^{it}}^{i'}$ is a binary parameter. If flight $i'$ chooses to take off in time window $t$ and enters sector $j$ in time window $\chi_j^{it}$, then $\zeta_{j\chi_j^{it}}^{i'} = 1$; otherwise, $\zeta_{j\chi_j^{it}}^{i'} = 0$. Therefore, $d_{j\chi_j^{it}}^{\text{Potential}}$ refers to the potential demand based on the assumption that all flights in time window $t$ choose to take off in the current time window. Similar to $c_{j\chi_j^{it}}^{\text{Remaining}}$, if sector $j$ is not open in time window $\chi_j^{it}$, the potential demand $d_{j\chi_j^{it}}^{\text{Potential}}$ will be set to zero.

As mentioned in Section 2.1, since our method only considers flight segments on the operation day, no sector opening scheme for the next day is inputted in the solver. However, as the flight is delayed in the decision-making framework, part of the flight segment may be moved to the next day. To handle this problem, we set the corresponding remaining capacity $c_{j\chi_j^{it}}^{\text{Remaining}}$ and potential demand $d_{j\chi_j^{it}}^{\text{Potential}}$ to zero for any part on the next day.

The number of elements in observation $\boldsymbol{o}_t^i$ is $(2M_i + 2)$, where $M_i$ denotes the number of sectors that flight $i$ is scheduled to pass through. The size of observation $\boldsymbol{o}_t^i$ is much smaller than the global state $\boldsymbol{s}_t^i$ (mentioned in Section 2.1).

*3.1.3. Reward design*

Our objective is to avoid overloads and minimise the mean delay time of all flights. A reward function is designed to guide a set of flights to achieve this objective. $r_t^i$ denotes the reward for action $a_t^i$, which is received by flight $i$ at time window $t$. $r_t^i$ is represented as a sum of three terms:

$$r_t^i = \left(r^{\text{Overload}}\right)_t^i + \left(r^{\text{Optimisation}}\right)_t^i + \left(r^{\text{Holding}}\right)_t^i \tag{16}$$

where $\left(r^{\text{Overload}}\right)_t^i$ is used to avoid overloads; $\left(r^{\text{Optimisation}}\right)_t^i$ is used to reduce the total delay time; $\left(r^{\text{Holding}}\right)_t^i$ is used to adjust the distribution of delays.

Specifically, when flight $i$ chooses to take off and thus causes overloading of any sector it is scheduled to pass through, it is penalised by $\left(r^{\text{Overload}}\right)_t^i$:

$$\left(r^{\text{Overload}}\right)_t^i = \begin{cases} -\kappa \times \sigma_t^i & a_t^i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\kappa$ is the overload penalty coefficient, and $\sigma_t^i$ denotes the sum of the overloads in the sectors through which the flight $i$ is scheduled to pass based on the actions of flights $i \in I_t^{\text{Departure}}$:

$$\sigma_t^i = \sum_{j \in J_i} \Re\left(\sum_{i' \in I_t^{\text{Departure}}} \psi_{j\chi_j^{it}}^{i'} - C_{j\chi_j^{it}}\right) \tag{18}$$

where $\Re$ is the Rectified Linear Unit (ReLU) and $\Re(x) = \max(x, 0)$. When flight $i$ chooses to take off, it will receive $\left(r^{\text{Optimisation}}\right)_t^i$ based on the total delay time for all flights:

$$\left(r^{\text{Optimisation}}\right)_t^i = \begin{cases} -\frac{1}{N} \sum\limits_{i' \in I} h_{\text{final}}^{i'} & a_t^i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where $h_{\text{final}}^i$ represents the number of times the flight $i$ ended up holding and $N$ is the number of flights. To encourage the flight to choose a suitable departure time based on its collaboration preferences, we give the flight $r^{\text{Holding}}$ for its holding:

$$\left(r^{\text{Holding}}\right)_t^i = \begin{cases} -\frac{1}{\mu_i} \times \frac{h_t^i}{N} & a_t^i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where $\mu_i$ is the coefficient of cooperation. The larger the cooperation coefficient $\mu_i$ of flight $i$, the smaller the penalty it receives per holding and the more it prefers to cooperate (to delay itself to reduce the total delay time for all flights).

Note that, unlike traditional optimisation methods, reinforcement learning methods cannot add constraints to the model; instead, the constraints are transferred to the reward function. Therefore, the reward function needs to fulfil the functions of both optimisation objectives and constraints. The three terms of reward function $r_t^i$ in our model play different roles. $\left(r^{\text{Overload}}\right)_t^i$ serves as a constraint to avoid sector overload for safety. $\left(r^{\text{Optimisation}}\right)_t^i$ and $\left(r^{\text{Holding}}\right)_t^i$ together serve as an optimisation objective to reduce the total delay while considering the delay distribution.

For reward functions of RL-based DCB models, factors of interest (such as safety) must be designed carefully[44]. Due to reward function $r_t^i$ being a sum of the three terms (refer to Eq. (16)), the weights of each part can be treated as $\kappa$, 1 and $1/\mu_i$ for $\left(r^{\text{Overload}}\right)_t^i$, $\left(r^{\text{Optimisation}}\right)_t^i$ and $\left(r^{\text{Holding}}\right)_t^i$, respectively (refer to Eqs. (17), (19) and (20)). In practice, the cooperation factor for each flight can vary according to ATM requirements and airline preference. To simplify the problem, we set all the cooperation coefficients to the same value ($\mu_i = 1, i \in I$) as a basic model, and more discussion about the cooperation coefficient can be seen in Section 4. To fulfil the constraint function of $\left(r^{\text{Overload}}\right)_t^i$, we set $\kappa$ to 100, which is much greater than other weights, so that it can be prioritised.

The three terms of reward function $r_t^i$ are negative or zero. The closer the agent's action is to the expectation, the closer $r_t^i$ is to 0. In our method, the agent chooses the action with the greatest value; the agent only cares about the relative value of the rewards for different actions, regardless of whether they are positive or negative.

### 3.2. Network architecture

This paper's policy network consists of a four-layer neural network ,as shown in Fig. 4. Firstly, the observation matrix $\boldsymbol{o}_t^i$ is composed of $\boldsymbol{\Phi}_t^i$, $t$ and $h_t^i$, where $\boldsymbol{\Phi}_t^i$ is obtained by partial observation of the environment. Then, $\boldsymbol{o}_t^i$ is flattened as the input layer of the neural network. The number of neurons in the input layer is equal to the number of elements in the observation $\boldsymbol{o}_i^t$, that is, $(2M_i + 2)$. The hidden_1 layer is the main hidden layer (like other standard three-layer neural networks) which contains $(M_i + 1)$ neurons. The hidden_2.A layer and hidden_2.V layer, respectively, represent the advantage function and the value function for the Duelling DQN technology (please refer to Section 3.3). The hidden_2.A layer and hidden_2.V layer consist of two neurons and one neuron, respectively. The values of the output layer's neurons are obtained by a linear combination of the values of the neurons in the hidden_2.A layer and hidden_2.V layer (please refer to Algorithm 1, line 9 ). Therefore, the output layer consists of two neurons. The input, hidden_1, hidden_2.A and hidden_2.V layers are all fully connected. The selected action $a_t^i$ corresponds to the neuron with the maximum value in the output layer. If the action is to hold, the flight will be moved to the next time window. Then it repeats the above process until it chooses departure.

### 3.3. Training method

#### 3.3.1. Training algorithm

Because the proposed reinforcement learning model's action is discrete and has only two actions, DQN is ideal as an efficient reinforcement learning algorithm for discrete actions[45]. In recent years, several DQN performance improvement techniques have been proposed. We adaptively employ and combine the Double-DQN[46], Duelling DQN[47] and Prioritised Replay Buffer[48] to enhance the performance of DQN, inspired by a recently proposed advanced DQN method, Rainbow DQN[43].

The algorithm of learning for DCB is summarised in Algorithm 1[46–49]. In this algorithm, each episode consists of two parts. One is the simulation and collecting transitions (lines 4–29), and the other is the policy update (lines 31–37).

**Simulation and collecting transitions** The simulation scenario is reset first (line 4, please refer to Section 3.3.2). From the first time window (line 5), when the current time window is on the current day, all flights in the current time window evaluate the value of each action $Q\left(\boldsymbol{o}_i^t, a; \theta\right)$ based on Duelling DQN (line 9)[47] and then choose the action $a_t^i$ with the greatest value by $\varepsilon$-greedy method (line 10). If flight $i$ chooses to hold, the number of its holdings will be updated (line 12), and it will be moved to the next time window (line 13). Next, the holding reward $\left(r^{\text{Holding}}\right)_t^i$ is collected (line 15). Then, the data of sector remaining capacity and potential demand is updated based on actions in the current time window (line 17). Moreover, for every action in the current time window, the number of overloads of flight $i$ is calculated so that the overload reward $\left(r^{\text{Overload}}\right)_t^i$ can be collected (line 20). Then the next observation $\boldsymbol{o}_i^{t+1}$ is collected. The above process will be repeated in the next time window (line 23) until all the time windows in the current day are iterated. After that, for every action in the current episode, the optimisation reward $\left(r^{\text{Optimisation}}\right)_t^i$ is calculated and collected (line 26). Then, we collect the reward $r_t^i$ because we have collected each component of it (line 27). Finally, the transition $(\boldsymbol{o}_t^i, a_t^i, r_t^i, \boldsymbol{o}_{t+1}^i)$ is stored in buffer $\wp_i$ (line 28).

**Policy update** Transitions stored in buffer $\wp_i$ are sampled by SumTree, one of the key technologies in Prioritised Replay Buffer (line 33)[48]. Then, the loss function $l$ is calculated by Double DQN (line 34)[46]. Next, the evaluation network $\pi_{\theta_i}^i$ is updated by gradient descent every step (line 35), while the target network $\pi_{\theta_i'}^i$ is updated by copying the parameters of $\pi_{\theta_i}^i$ every period (line 36)[46]. Finally, the trained policy $\pi_{\theta_i}^i$ is obtained.

As agents in our MARL model are distributed, several parts of the algorithm can run in parallel, including lines 7–16, lines 18–22 and lines 31–37.

#### 3.3.2. Training scenarios

This paper's scenario data consists of the sector opening scheme and flight schedule. We use the real data of French and Spanish airspace, as shown in Fig. 5. Specifically, 12187 flights on 2019-08-27 and 12138 flights on 2019-08-28 are selected for training (a total of 24325 flights), and a unique agent is deployed on each of them. The sector opening schemes for each day in August 2019 (31 sector opening schemes in total) are used for training scenarios. Three hundred ninety-six sectors are considered, including elementary sectors and collapsed sectors. Based on the real data mentioned above, each training scenario comprises a set of flight schedules and a sector opening scheme for any day. Thus, there are 62 different training scenarios in total, and one of them is randomly selected in each episode for training (refer to Algorithm 1, line 4).

### 3.4. Multi-iteration mechanism

Due to the non-stationarity in MARL, in our model, the trained agents do not guarantee that hotspots will be wholly eliminated with just one iteration. One iteration refers to that the trained agents make action decisions for all flights of the day according to the

**Algorithm 1** Learning to solve DCB problems[46–49].

**Input:** The dataset of simulation scenarios, including flight schedule and sector opening scheme

Initialise the evaluation network $\pi^i_{\theta_i}$ and target network $\pi^i_{\theta_i'}$ for flight $i \in I$, and set hyper-parameters as shown in Table 2
**for** episode = 1, 2, ..., **do**
  # Simulation and collecting transitions
  Reset the simulation scenario
  $t \leftarrow 0$ # the first time window
  **while** $t < N^{\mathrm{T}}$ **do**
    **for** flight $i \in I_t$ **do**
      # in parallel
      Evaluate each action value based on Duelling DQN: $Q\left(\boldsymbol{o}_t^i, a; \theta_i\right) = V\left(\boldsymbol{o}_t^i\right) + A\left(\boldsymbol{o}_t^i, a\right) - \frac{\sum_{a' \in \{0,1\}} A(\boldsymbol{o}_t^i, a')}{2}, a \in \{0, 1\}$[47]
      Choose ($\varepsilon$-greedy) the action with the greatest value: $a_t^i = \arg\max_a Q\left(\boldsymbol{o}_t^i, a; \theta_i\right)$[49]
      **if** $a_t^i = 0$ **then**
        $h_t^i \leftarrow h_t^i + 1$
        Move flight $i$ from $I_t$ to $I_{t+1}$
      **end if**
      Collect the holding reward $\left(r^{\text{Holding}}\right)_t^i$
    **end for**
    Update sector remaining capacity $c_{jt}^{\text{Remaining}}$ and potential demanding $d_{jt}^{\text{Potential}}$, $j \in J, t \in T$
    **for** every $a_t^i$ in this time window **do**
      # in parallel
      Calculate overloads $\sigma_t^i$ and collect overload reward $\left(r^{\text{Overload}}\right)_t^i$
      Collect the observation of the next step $\boldsymbol{o}_i^{t+1}$
    **end for**
    $t \leftarrow t + 1$ # next time window
  **end while**
  **for** every $a_t^i$ in this episode **do**
    Calculate the total holding times for all flights and collect the optimisation reward $\left(r^{\text{Optimisation}}\right)_t^i$
    Collect the reward $r_t^i = \left(r^{\text{Overload}}\right)_t^i + \left(r^{\text{Optimisation}}\right)_t^i + \left(r^{\text{Holding}}\right)_t^i$
    Store the transition $(\boldsymbol{o}_t^i, a_t^i, r_t^i, \boldsymbol{o}_{t+1}^i)$ in the buffer $\wp_i$ whose capacity is $n_{\text{BC}}$
  **end for**
  # Policy update
  **for** flight $i \in I$ **do**
    # in parallel
    Sample $n_{\text{BS}}$ transitions by Prioritised Replay Buffer (SumTree)[48] from the buffer $\wp_i$
    Calculate the loss function by Double DQN : $l = \left(r_{t+1}^i + \gamma Q\left(\boldsymbol{o}_t^i, \arg\max_{a^*} Q\left(\boldsymbol{o}_{t+1}^i, a^*; \theta_i\right); \theta_i'\right) - Q\left(\boldsymbol{o}_t^i, a_t^i; \theta_i\right)\right)^2$[46]
    Update the evaluation network $\pi^i_{\theta_i}$: $\pi^i_{\theta_i} \leftarrow \pi^i_{\theta_i} - \alpha \frac{\partial l}{\partial \pi^i_{\theta_i}}$ # every step[46]
    Update the target network $\pi^i_{\theta_i'}$: $\pi^i_{\theta_i'} \leftarrow \pi^i_{\theta_i}$ # every $n_{\text{TN}}$ steps[46]
  **end for**
**end for**

**Output:** Policy $\pi^i_{\theta_i}$

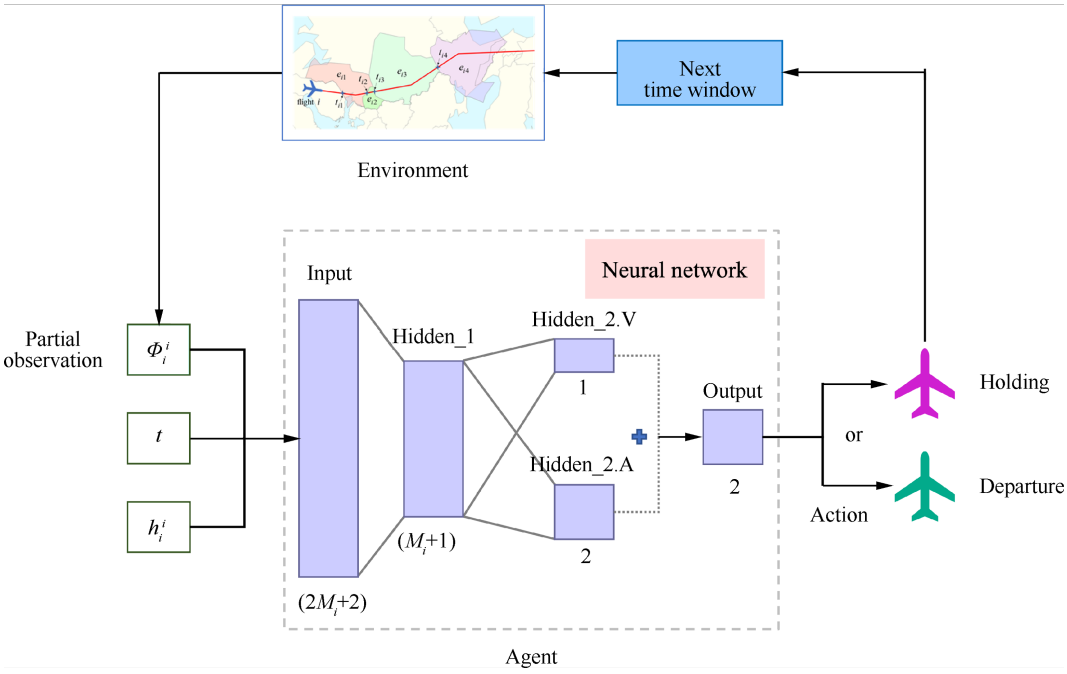



**Fig. 4** Architecture of policy network, showing the working process of an agent and focusing on neural network.

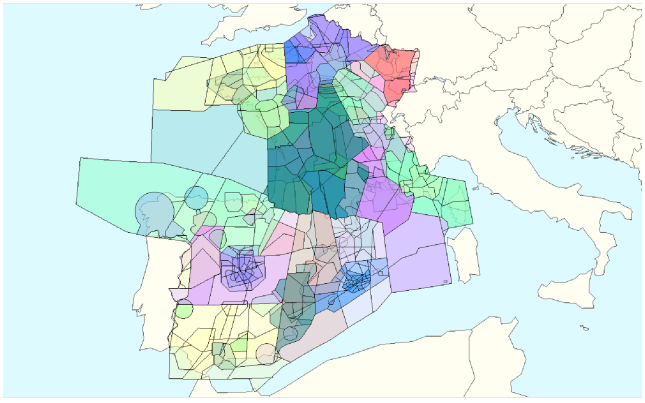


**Fig. 5** French and Spanish sectors.

order of the time window until all flights are assigned a departure time or delayed to the next day. Note that, to date, there are few proven methods to handle non-stationarity in MARL effectively[50]. Therefore, a multi-iteration mechanism is introduced to deal with the problem arising from the non-stationarity in MARL. If there are still hotspots that have not been eliminated after the first iteration, multiple iterations are performed based on the assigned departure time until all hotspots are eliminated. Please note that the iteration following the first iteration differs from the first iteration. For legibility, we call an iteration after the first iteration a subsequent iteration. The framework of the subsequent iteration in the multi-iteration mechanism is shown in Fig. 6. In each time window of the subsequent iteration, flights are divided into two categories:

(1) One is the flight (represented by Flight *A* in Fig. 6) which does not cause hotspots based on the currently assigned departure time. Its departure time will remain unchanged; it is scheduled to take off in the current time window.
(2) The other is the flight (represented by Flight *B* in Fig. 6) which causes hotspots based on the currently assigned departure time. Through its trained neural network, the agent deployed on the flight will decide whether the flight will take off at the current time window or be delayed to the next time window.

Please note that this multi-iteration mechanism is only used for trained agents to solve a DCB problem and not for training. In training, only one iteration is performed. In solving, the first iteration (refer to Fig. 3) is performed firstly. If, after the first iteration, there are still hotspots that have not been eliminated, subsequent iterations (refer to Fig. 6) are performed until all the hotspots have been eliminated.

## 4. Simulation experiments

In this section, we first introduce the training setup and present the training results. Next, we test the proposed method's generalisation and compare it with state-of-the-art RL-based DCB methods. Finally, we discuss the sensitivity analysis of the cooperation coefficient.

### *4.1. Training*

The neural networks are constructed based on the Pytorch library in the Python environment and are trained on a computer with one i5-8300H CPU. The Adam optimisation algorithm[51] updates the network parameters. We determine the values of the hyper-parameters in Algorithm 1 through tuning experiments, and they are summarised in Table 2. Specifically, the exploration rate $\varepsilon$ is set to 0.95; the buffer capacity $n_{\mathrm{BC}}$ is set to 200; the batch size $n_{\mathrm{BS}}$ is set to 20; the discount rate $\gamma$ is set to 0.9; the learning rate $\alpha$ is set to $1 \times 10^{-3}$; the target network update cycle $n_{\mathrm{TN}}$ is set to 100. In each training episode, one of the 62 scenarios in the training dataset is randomly input to the reinforcement learning environment as the training scenario (refer to Section 3.3.2 for details).

To study the training process and the effectiveness of each DQN enhancement technology in our model, we introduce and compare the training results of five models: Model-1, Model-2, Model-3,

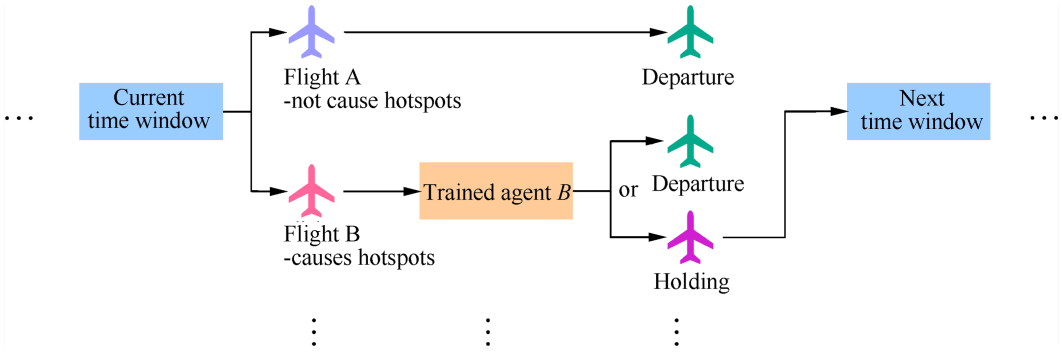



**Fig. 6** Framework of the subsequent iteration in the multi-iteration mechanism.

**Table 2** Hyper-parameter settings for training algorithm.

| Hyper-parameter | Value |
|---|---|
| Exploration rate $\varepsilon$ in line 10 | 0.95 |
| Buffer capacity $n_{\mathrm{BC}}$ in line 28 | 200 |
| Batch size $n_{\mathrm{BS}}$ in line 33 | 20 |
| Discount rate $\gamma$ in line 34 | 0.9 |
| Learning rate $\alpha$ in line 35 | $1 \times 10^{-3}$ |
| Target network update cycle $n_{\mathrm{TN}}$ in line 36 | 100 |

Model-4 and Model-5. The features of each model are summarised in Table 3, where the checkmark (✓) means that the model uses the corresponding DQN enhancement technology while the circle (∘) means not.

**Table 3** Model features.

| Model | DQN enhancement technology | | |
|---|---|---|---|
| | Double DQN | Duelling DQN | Prioritised replay buffer |
| 1 | ✓ | ✓ | ✓ |
| 2 | ∘ | ✓ | ✓ |
| 3 | ✓ | ∘ | ✓ |
| 4 | ✓ | ✓ | ∘ |
| 5 | ∘ | ∘ | ∘ |

Model-1 is the DQN method used in our MARL method, and it is enhanced by Double DQN, Duelling DQN and Prioritised Replay Buffer technology. Model-2, Model-3 and Model-4 are enhanced by corresponding two of the three enhancement technologies. Model-5 is the original DQN method without any enhancement technology. In each episode, all neural networks are evaluated as a whole by the average reward $\xi_p^{\mathrm{Ave}}$, where $p$ is the episode number, $N^p$ denotes the number of flights in the $p$th episode, $I^p$ is the set of flights in the $p$th episode, $\left(\mathbf{\Lambda}^i\right)^p$ represents the set of actions of flight $i$ in the $p$th episode and $\left(a_t^i\right)^p$ denotes the action in it:

$$\xi_p^{\mathrm{Ave}} = \frac{1}{N^p} \sum_{i \in I^p} \sum_{\left(a_t^i\right)^p \in \left(\mathbf{\Lambda}^i\right)^p} r_t^i \tag{21}$$

The training result is shown in Fig. 7. The light curves are the actual result data and the dark ones are fitted curves based on the former by 90% smoothing coefficient for legibility. We can find that the trend of change of these curves is the same; after a short fluctuation, it rises rapidly and then tends to stabilise and gradually converge. Model-1 and Model-2 grow the fastest and converge around the 800th episode. The other three curves stabilised around the 1600th episode. In terms of performance, according to the average reward, Model-1 is the best, and Model-2 is the second. Model-3 and Model-4 are nearly identical. Model-5 is significantly inferior to the others. In summary, the training of our neural networks is effective, and the DQN enhancement technologies in our model can effectively improve training efficiency and model performance.

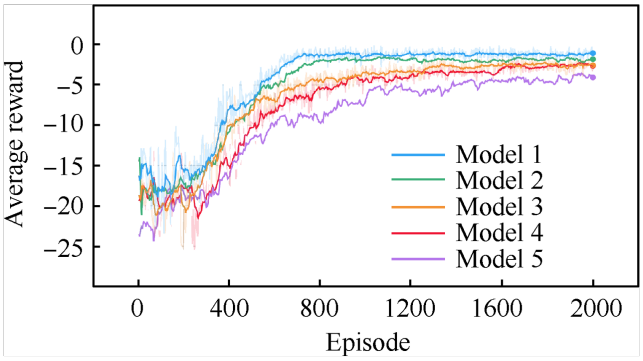



**Fig. 7** Training result.

### 4.2. *Performance test*

#### 4.2.1. *Benchmark and test scenario*

To verify the optimisation performance of the proposed MARL method, we use CASA as a benchmark[52]. CASA is a DCB solver based on a heuristic algorithm, and it is often used in actual ATM operations. It has the advantages of requiring less computing time and being simple to use in the real world. CASA is a general term for a class of methods whose specific algorithms are often fine-tuned to suit specific scenarios. For ease of comparison, in this paper, we use a standard CASA introduced in ATFCM OPERATIONS MANUAL - Network Manager by EUROCONTROL[53], and the CASA's source code based on Python3 is available on our Github.

To test the performance of our proposed method in diverse scenarios different from the training scenarios, we use sector opening schemes in July 2019 and combine the flights on 2019-08-27 and 2019-08-28 to generate test scenarios, as shown in Fig. 8. Three sets of test scenarios are generated for three corresponding tests to verify the generalisation of our method for sector opening schemes, flight schedules and problem scales, respectively. The numbers shown in Fig. 8 represent the number of the corresponding element. For example, in Test 2, it means that there are 7 sets of flight schedules and 31 sector opening schemes, and they make up a total of 217 scenarios. Besides, we compare our method with two state-of-the-art RL-based DCB methods. The experimental design

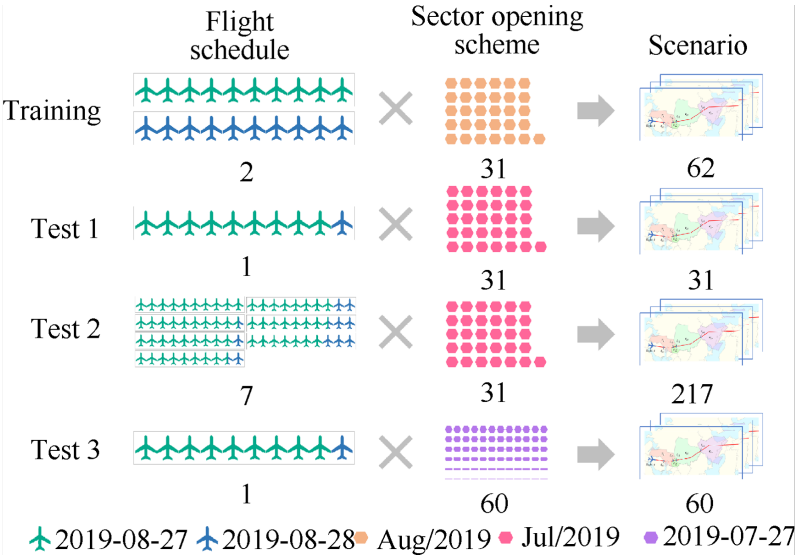



**Fig. 8** Scenarios generation for performance tests.

and results are shown as follows.

*4.2.2. Generalisation for sector opening scheme (Test 1)*

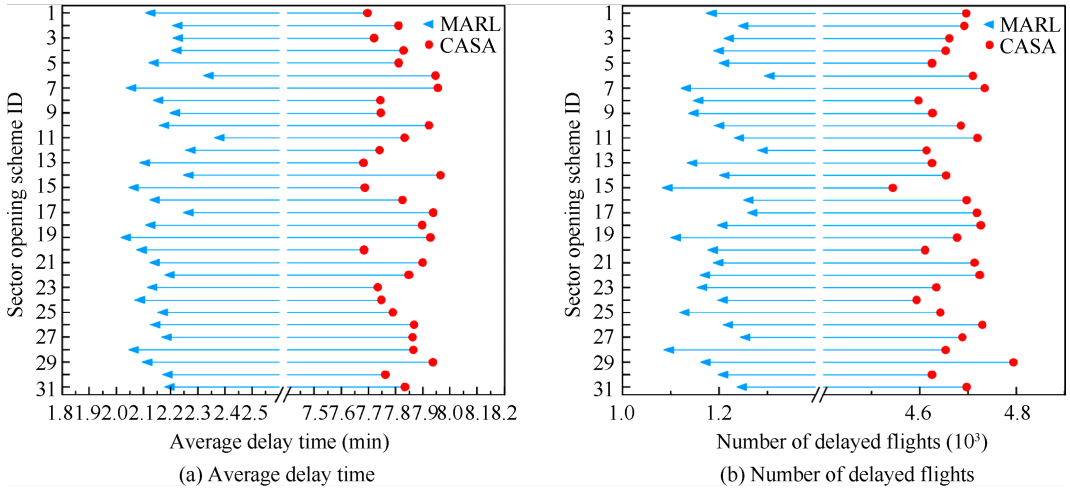
Test 1 aims to verify the generalisation of our method for the sector opening scheme. Although commercial flights are repeated for a certain period, the flight schedules for corresponding days will not be the same. Therefore, we randomly select 90% of flights on 2019-08-27 and 10% of flights on 2019-08-28 to form a new set of flight schedules which contains 12182 flights. To differentiate from the training scenario, we use the 31 sector opening schemes in July 2019 in the test scenarios. The new set of flight schedules and the 31 sector opening schemes form 31 test scenarios (also refer to Fig. 8). Two performance indicators are used to reflect the optimisation performance of the method, namely the number of delayed flights and the average delay time. Please note that the average delay time is the mean for all flights in a scenario. The 31 test scenarios are solved by our proposed MARL method and CASA. The results are shown in Fig. 9, where the sector opening scheme ID corresponds to the day in July 2019 (for example, 01 refers to 2019-07-01). Compared with CASA, MARL decreases the average delay time and the number of delayed flights by about 72.5% and 74.6%, respectively. Since the performance of our method, according to the benchmark, does not vary significantly in different scenarios of Test 1, it can be considered generalised for various sector opening schemes.

*4.2.3. Generalisation for flight schedule (Test 2)*

Test 2 aims to verify the generalisation of our method for flight schedules. We generate seven sets of flight schedules by combining the flights on 2019-08-27 and 2019-08-28 with different proportions. We use the rate of difference to represent the proportion of non-same-day flights. Assuming that the flights on 2019-08-27 are taken as main flights in a test scenario while the test set of flight schedules consists of $x$ flights on 2019-08-27 and $y$ flights on 2019-08-28, the rate of difference is $\frac{y}{x+y}$. We randomly select flights on 2019-08-27 and 2019-08-28 based on seven specific rates of difference and test them in 31 sector opening schemes in July 2019; there are 217 test scenarios in total (also refer to Fig. 8). The results of Test 2 are shown in Fig. 10, where C and M in the boxes below the figure represent CASA and MARL, respectively (grey represents CASA and red represents MARL; this representation is also used in Fig. 11). IQR is interquartile range. Although the average delay time and the number of delayed flights of the results when the rate of difference is 0 in both are the smallest on average, the two performance indicators, by comparing with the results of CASA, show no significant uptrend as the rate of difference increases. In other words, the method does not significantly decrease performance due to the rise in the rate of difference in flights. Therefore, it is demonstrated that the proposed method can cope with limited flight combinations.

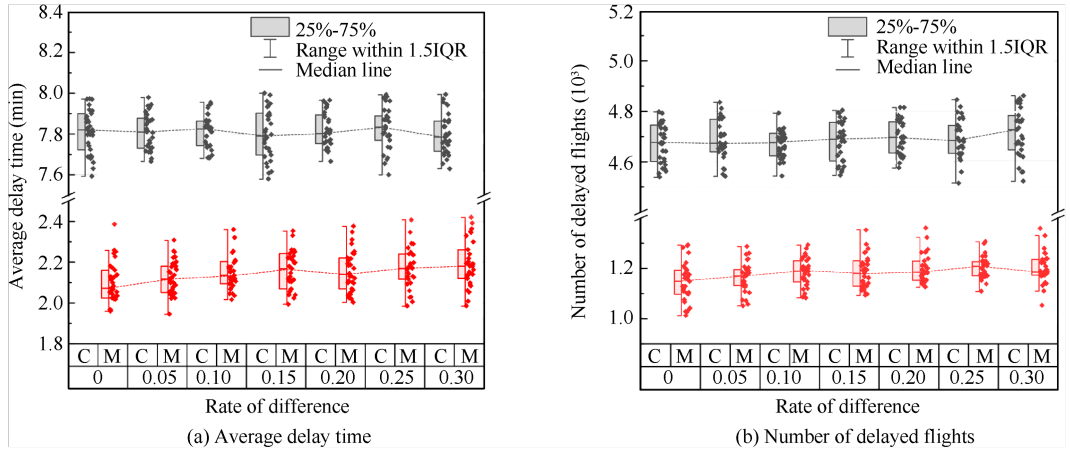
*4.2.4. Generalisation for problem scale (Test 3)*

Test 3 aims to verify the generalisation of our method for problem scale. To obtain several scenarios with different scales, we first select the sector opening scheme of 2019-07-27 and then randomly select a specific number of sectors to form a new sector opening scheme. We generate 6 sets of 10 sector opening schemes in total, with each set's sector opening schemes containing a different number of sectors. The flight schedule in Test 1 is tested in each opening scheme separately for 60 scenarios in Test 3 (also refer to Fig. 8). The results of Test 3 are shown in Fig. 11. In addition to the average delay time and the number of delayed flights, each scenario's computing time and initial condition are also presented for reference. For the 60 scenarios in Test 3, the initial number of hotspots and the initial number of flights increase with the number of sectors, as shown in Fig. 11(d). As expected, the average delay time and the number of delayed flights increase as the problem scale increases. The growth trends of the two optimisation indicators for MARL are linear, similar to the growing trend of the problem scale. Therefore, the optimisation performance of our proposed method is similar in scenarios of different problem scales, which demonstrates its generalisation for problem scales. As shown in Fig. 11(c), although MARL's computing time is much longer than CASA's in the same scenario, its growth trend is almost linear rather than exponential as the problem scale increases, and the computing time for the scenario with 300 sectors and more than 10000 flights is about 30 s, which is acceptable for a DCB problem of this scale.




(a) Average delay time (b) Number of delayed flights

**Fig. 9** Experimental results of method generalisation test for sector opening schemes.




(a) Average delay time (b) Number of delayed flights

**Fig. 10** Experimental results of method generalisation test for flight schedules.

*4.2.5. Comparison with state-of-the-art RL-based DCB methods*

This test aims to compare the proposed method with the state-of-the-art RL-based DCB methods. Because Tang's[41] and Huang's[42] methods have the potential for generalisation, we compare our method with them in the computing time and average delay time. The same way is followed to train them as with the proposed method. Because the two methods do not consider sector opening schemes, we upgrade them to enable dealing with DCB problems with sector opening schemes. The comparison is based on ten random scenarios of Test 2, and the experimental results are shown in Fig. 12, where MARL represents the proposed method, T represents Tang's method[41] and H represents Huang's method[42]. Although this test is for comparing RL-based DCB methods, the performance data of CASA are still added to Fig. 12 for reference. We can find that the proposed method outperforms the two state-of-the-art RL-based DCB methods. In terms of the average delay time, compared with the proposed method, Tang's method and Huang's method increase by about 116% and 79%, respectively. The reason may be that their methods do not have enough generalisation for the test scenarios. In terms of the computing time, compared with the proposed method, Tang's method and Huang's method increase by about 43% and 620%, respectively. The delay time allocated by the agent in Huang's method is tiny at each iteration, so a considerable number of iterations are needed to eliminate hotspots, which is significantly time-consuming. The decision-making framework of Tang's method is similar to ours; a longer average time means that agents make more actions, which is also time-consuming. It tends to explain why the computing time for Tang's method is longer than ours but very close.

*4.3. Sensitivity analysis*

Sensitivity analysis on the coefficient of cooperation (refer to Eq. (20)) is performed to explore the impact on our proposed method. We re-train another four sets of neural networks in the training scenarios (mentioned in Section 3.3) with different coefficients of cooperation. With the set of neural networks trained on the base model, there are five sets of neural networks for training flights; the coefficient of cooperation, respectively, is set to 0.25, 0.5, 1, 2 and 4. Then, the five sets of neural networks are deployed on the corresponding flights and tested in the scenarios of Test 1, respectively. The experimental results in the average delay time and the number of delayed flights are shown in Fig. 13. From a statistical point of view, as the coefficient of cooperation increases, the average delay time decreases while the number of delayed flights increases. The reason is that the greater the coefficient of cooperation, the lower the proportion of the penalty of holding in the total reward, and the more likely the flight is to promote the achievement of the global optimisation goal even if its own delay time is longer. Thus, in practice, an appropriate cooperation coefficient should be set according to the actual needs of air traffic operations to balance the average

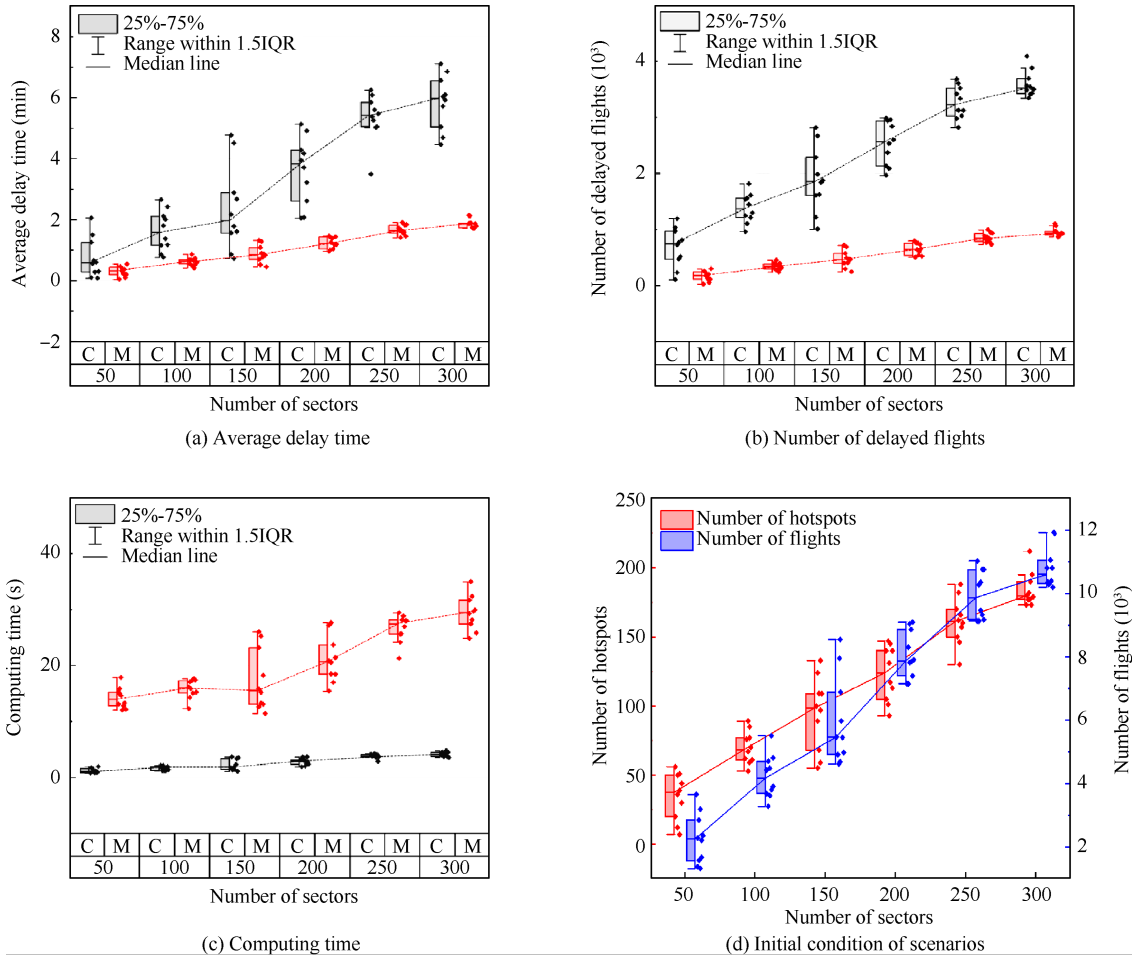



**Fig. 11** Experimental results of the method generalisation test for problem scales.

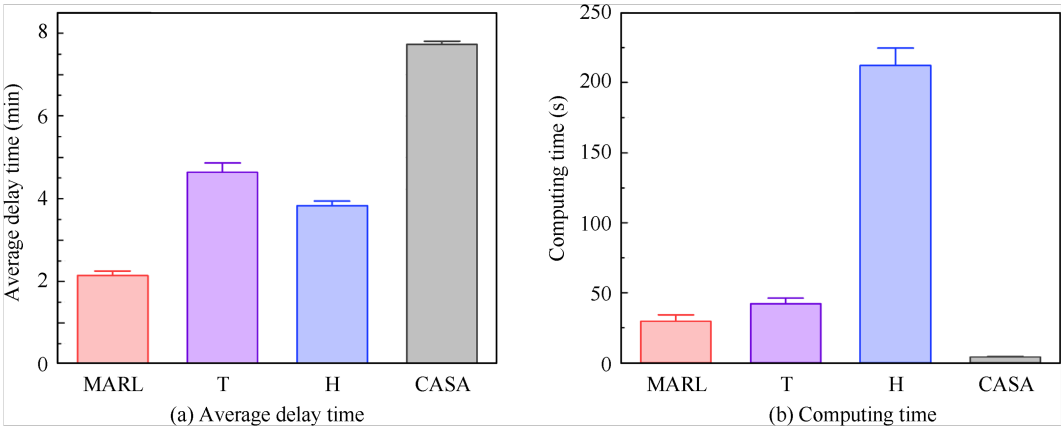



**Fig. 12** Experimental results of the comparison of RL-based DCB methods.

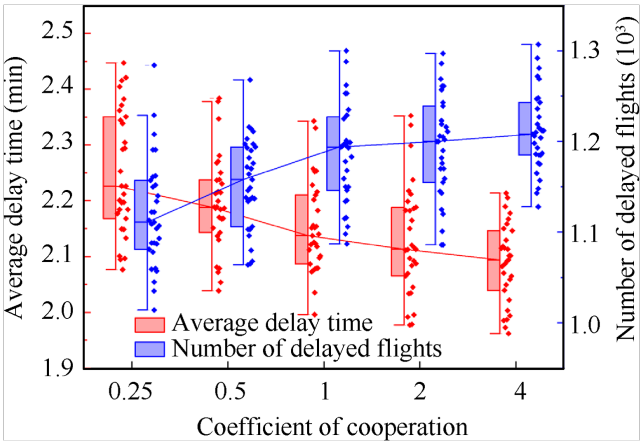



**Fig. 13** Experimental results of sensitivity analysis on coefficient of cooperation.

delay time and the number of delayed flights.

## 5. Conclusions

In this paper, we have presented a locally generalised multi-agent reinforcement learning for demand and capacity balancing with customised neural networks. DQN-based enhancement technologies (i.e., Double DQN, Duelling DQN and Prioritised Replay Buffer) have been proven to improve the learning efficiency of agents. We have evaluated the performance of our method using a series of comprehensive experiments based on large-scale high-complexity real-world scenarios. By analysing the experimental results, we can draw the following conclusions:

(1) The proposed method is generalised for sector opening schemes, flight schedules and problem scales in the scope of interest.
(2) The proposed method outperforms CASA in optimisation performance.
(3) The proposed method outperforms the state-of-the-art RL-based DCB methods [41 42] in computing time and optimisation performance.
(4) The trend of average delay time and the number of delayed flights are negatively correlated as the cooperation coefficient changes in the proposed method.

This work serves as our first step toward reducing the gap between the theory and practice of multi-agent reinforcement learning methods used for demand and capacity balancing. We are fully aware that the situation in practice is far more complicated than that in our experiments. Compared with traditional methods (such as exact solution methods), reinforcement learning methods have the potential to deal with uncertainty because they are based on probability inherently. In the future, the uncertainty should be taken into account in RL-based DCB methods to improve the potential for practical application. Furthermore, without compromising the optimisation performance, we will further enhance the generalisation of the technique to reduce the training cost. For example, all agents with the same flight path share the same neural network.

## Acknowledgements

The research described in this paper has been partly conducted in the framework of a project that has received funding from the SESAR Joint Undertaking within the framework of the European Union's Horizon 2020 research and innovation programme under grant agreement No 891965. This study is co-funded by the National Natural Science Foundation of China (No.61903187), the National Key R&D Program of China (No.2021YFB1600500), the China Scholarship Council (No.202006830095), the Natural Science Foundation of Jiangsu Province (No.BK20190414) and the Jiangsu Province Postgraduate Innovation Fund (No.KYCX20_0213).

## References

1. Mannino C, Nakkerud A, Sartor G. Air traffic flow management with layered workload constraints. *Comput Oper Res* 2021;**127**:105159.
2. ICAO. Global Air Traffic Management Operational Concept. Montreal: ICAO; 2005. Report No.: Doc-9854.
3. EUROCONTROL. European ATM master plan - digitalising Europe's aviation infrastructure [Internet]. 2019 Dec [cited2022 Mar 23]. Available from: https://www.sesarju.eu/masterplan2020.
4. EUROCONTROL. Exploring the boundaries of air traffic management - a summary of SESAR exploratory research results [Internet]. 2018 [cited 2022 Mar 23]. Available from: https://www.sesarju.eu/sites/default/files/documents/reports/ER_Results_2016_2018.pdf.
5. Odoni AR. The flow management problem in air traffic control. *Flow control of congested networks*. Berlin: Springer; 1987. p. 269-88.
6. Choroba P, Van Der Hoorn L. Towards a more harmonised and wider use of Short-Term Atfcm Measures (STAM). *30th congress of the international council of the aeronaurical sciences*. 2016 Sep 25-30; Daejeon, Korea. Bonn: ICAS; 2016. p. 1-10.
7. Ivanov N, Jovanović R, Fichert F, et al. Coordinated capacity and demand management in a redesigned Air Traffic Management value-chain. *J Air Transp Manag* 2019;**75**:139-52.
8. Montlaur A, Delgado L. Flight and passenger efficiency-fairness trade-off for ATFM delay assignment. *J Air Transp Manag* 2020;**83**:101758.
9. Liu YL, Liu Y, Hansen M, et al. Using machine learning to analyze air traffic management actions: Ground delay program case study. *Transp Res E: Logist Transp Rev* 2019;**131**:80-95.
10. Diao XD, Chen CH. A sequence model for air traffic flow management rerouting problem. *Transp Res E: Logist Transp Rev* 2018;**110**:15-30.
11. Murca MCR. Collaborative air traffic flow management: Incorporating airline preferences in rerouting decisions. *J Air Transp Manag* 2018;**71**:97-107.
12. Huang JL, Xu QC, Yan YJ, et al. Generalized method of modeling minute-in-trail strategy for air traffic flow management. *Math Probl Eng* 2019;**2019**:1-14.
13. Zhang Y, Xu DH, Guo YCF, et al. Research on performance analysis of air traffic flow management combined strategy impacted by uncertainty. *Proceedings of the 4th international conference on Humanities Science, Management and Education Technology (HSMET 2019)*. 2019 Jun 21-23; Singapore, Singapore. Paris: Atlantis Press; 2019. p. 238-44.
14. Sandamali GGN, Su R, Sudheera KLK, et al. A safety-aware real-time air traffic flow management model under demand and capacity uncertainties. *IEEE Trans Intell Transp Syst* 2022;**23**(7):8615-28.
15. Guo YCF, Hu MH, Zou B, et al. Air traffic flow management integrating separation management and ground holding: An efficiency-equity bi-objective perspective. *Transp Res B Methodol* 2022;**155**:394-423.
16. Xu Y, Prats X, Delahaye D. Synchronised demand-capacity balancing in collaborative air traffic flow management. *Transp Res C Emerg Technol* 2020;**114**:359-76.
17. Xu Y, Dalmau R, Melgosa M, et al. A framework for collaborative air traffic flow management minimizing costs for airspace users: Enabling trajectory options and flexible pre-tactical delay management. *Transp Res B Methodol* 2020;**134**:229-55.
18. Xiao MM, Cai KQ, Abbass HA. Hybridized encoding for evolutionary multi-objective optimization of air traffic network flow: A case study on China. *Transp Res E: Logist Transp Rev* 2018;**134**:35-55.
19. Torres S. Swarm theory applied to air traffic flow management. *Procedia Comput Sci* 2012;**12**:463-70.
20. Agustín A, Alonso-Ayuso A, Escudero L, et al. Mathematical opti-

mizationg models for air traffic flow management: A review [Internet]. 2010 Feb [cited 2022 Mar 23]. Available from: https://burjcdigital.urjc.es/bitstream/handle/10115/3405/ATFM_SOTA.pdf?sequence=1&isAllowed=y.
21. Kistan T, Gardi A, Sabatini R, et al. An evolutionary outlook of air traffic flow management techniques. *Prog Aerosp Sci* 2017;**88**:15-42.
22. Shone R, Glazebrook K, Zografos KG. Applications of stochastic modeling in air traffic management: Methods, challenges and opportunities for solving air traffic problems under uncertainty. *Eur J Oper Res* 2021;**292**(1):1-26.
23. Pham DT, Tran PN, Alam S, et al. Deep reinforcement learning based path stretch vector resolution in dense traffic with uncertainties. *Transp Res C Emerg Technol* 2022;**135**:103463.
24. Zhao P, Liu YM. Physics informed deep reinforcement learning for aircraft conflict resolution. *IEEE Trans Intell Transp Syst* 2022;**23**(7):8288-301.
25. Sui D, Xu P Wei, Zhang K. Study on the resolution of multi-aircraft flight conflicts based on an IDQN. *Chin J Aeronaut* .
26. Yilmaz E, Sanni O, Kotwicz Herniczek M, et al. Deep Reinforcement Learning Approach to Air Traffic Optimization Using the MuZero Algorithm. Reston: AIAA; 2021. Report No.: AIAA-2021-2377.
27. Li S, Egorov M, Kochenderfer M. Optimizing collision avoidance in dense airspace using deep reinforcement learning [Internet]. 2019 Dec [cited 2022 Mar 23]. Available from: https://arxiv.org/pdf/1912.10146.pdf.
28. Wen H, Li H, Wang Z, et al. Application of DDPG-based collision avoidance algorithm in air traffic control. *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*. 2019 Dec 14-15; Hangzhou, China. Piscataway: IEEE Press; 2019. p. 130-3.
29. Tran NP, Pham DT, Goh SK, et al. An intelligent interactive conflict solver incorporating air traffic controllers' preferences using reinforcement learning. *2019 Integrated Communications, Navigation and Surveillance conference (ICNS)*. 2019 Apr 9-11; Herndon, USA. Piscataway: IEEE Press; 2019. p. 1-8.
30. Ghosh S, Laguna S, Lim SH, et al. A deep ensemble multi-agent reinforcement learning approach for air traffic control [Internet]. 2020 Apr [cited 2022 Mar 23]. Available from: https://arxiv.org/pdf/2004.01387.pdf.
31. Zhang K, Yang YP, Xu CT, et al. Learning-to-dispatch: Reinforcement learning based flight planning under emergency. *2021 IEEE international Intelligent Transportation Systems Conference (ITSC)*. 2020 Sep 19-21; Indianapolis, USA. Piscataway: IEEE Press; 2021. p. 1821-6.
32. Crespo AMF, Gang L, De Barros AG. Reinforcement learning agents to tactical air traffic flow management. *Int J Aviat Manag* 2012;**1**(3):145-61.
33. Agogino AK, Tumer K. A multiagent approach to managing air traffic flow. *Auton Agents Multi-Agent Syst* 2012;**24**(1):1-25.
34. Kravaris T, Vouros GA, Spatharis C, et al. Learning policies for resolving demand-capacity imbalances during pre-tactical air traffic management. *German conference on multiagent system technologies*. 2017 Aug 23-26; Leipzig, Germany. Cham: Springer International Publishing; 2017. p. 238-55.
35. Spatharis C, Kravaris T, Vouros GA, et al. Multiagent reinforcement learning methods to resolve demand capacity balance problems. *Proceedings of the 10th hellenic conference on artificial intelligence*. 2018 Jul 9-12; Patras, Greece. New York: ACM; 2018. p. 1-9.
36. Kravaris T, Spatharis C, Blekas K, et al. Multiagent reinforcement learning methods for resolving demand-capacity imbalances. *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. 2019 Sep 23-27; London, UK. Piscataway: IEEE Press; 2018. p. 1-10.
37. Duong T, Todi KK, Chaudhary U, et al. Decentralizing air traffic flow management with blockchain-based reinforcement learning. *2019 IEEE 17th International conference on Industrial informatics (INDIN)*. 2019 Jul 23-25; Helsinki, Finland. Piscataway: IEEE Press; 2019. p. 1795-800.
38. Spatharis C, Blekas K, Bastas A, et al. Collaborative multiagent reinforcement learning schemes for air traffic management. *2019 10th international conference on Information, Intelligence, Systems and Applications (IISA)*. 2019 Jul 15-17; Patras, Greece. Piscataway: IEEE Press; 2019. p. 1-8.
39. Spatharis C, Bastas A, Kravaris T, et al. Hierarchical multiagent reinforcement learning schemes for air traffic management. *Neural Comput Applic* 2023;**35**:147-59.
40. Chen YT, Xu Y, Hu MH, et al. Demand and capacity balancing technology based on multi-agent reinforcement Learning. *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. 2021 Oct 3-7; San Antonio, USA. Piscataway: IEEE Press; 2021. p. 1-9.
41. Tang YF, Xu Y. Multi-agent deep reinforcement learning for solving large-scale air traffic flow management problem: A time-step sequential decision approach. *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. 2021 Oct 3-7; San Antonio, USA. Piscataway: IEEE Press; 2021. p. 1-10.
42. Huang C, Xu Y. Integrated frameworks of unsupervised, supervised and reinforcement learning for solving air traffic flow management problem. *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. 2021 Oct 3-7; San Antonio, USA. Piscataway: IEEE Press; 2021. p. 1-10.
43. Hessel M, Modayil J, Van Hasselt H, et al. Rainbow: Combining improvements in deep reinforcement learning. *32nd AAAI conference on artificial intelligence*. 2018 Feb 2-7; New Orleans, USA. Reston: AIAA; 2018. p. 1-14.
44. Cruciol LL, De Arruda Jr AC, WG Li, et al. Reward functions for learning to control in air traffic flow management. *Transp Res C Emerg Technol* 2013;**35**:141-55.
45. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015;**518**(7540):529-33.
46. Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI conference on artificial intelligence*. 2016 Feb 12-17; Phoenix, USA. Reston: AIAA; 2016. p. 1-7.
47. Wang ZY, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning. *Proceedings of the 33rd international conference on machine learning*. 2016 Jun 20-22; New York, USA. New York: PMLR; 2016. p. 1995-2003.
48. Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay [Internet]. 2016 Feb [cited 2022 Mar 23]. Available from: https://arxiv.org/pdf/1511.05952.pdf.
49. Dabney W, Ostrovski G, Barreto A. Temporally-extended $epsilon$-greedy exploration [Internet]. 2020 Jun [cited 2022 Mar 23]. Available from: https://arxiv.org/pdf/2006.01782.pdf.
50. Papoudakis G, Christianos F, Rahman A, et al. Dealing with non-stationarity in multi-agent deep reinforcement learning [Internet]. 2020 Jun [cited 2022 Mar 23]. Available from: https://arxiv.org/pdf/1906.04737.pdf.
51. Kingma DP, Ba J. Adam: A method for stochastic optimization [Internet]. 2014 Dec [cited 2022 Mar 23]. Available from: https://arxiv.org/pdf/1412.6980.pdf?source=post_page----------.
52. Ivanov N, Netjasov F, Jovanović R, et al. Air traffic flow management slot allocation to minimize propagated delay and improve airport slot adherence. *Transp Res A Policy and Pract* 2017;**95**:183-97.
53. EUROCONTROL. ATFCM operations manual - network manager [Internet]. 2021 Jan [cited 2022 Mar 23]. Available from: https://www.eurocontrol.int/sites/default/files/2021-01/eurocontrol-atfcm-operations-manual-24-1-18012021.pdf.