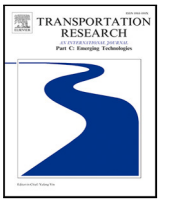
Transportation Research Part C 153 (2023) 104218



Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc



General multi-agent reinforcement learning integrating heuristic-based delay priority strategy for demand and capacity balancing

Yutong Chen [a,b], Yan Xu [a,*], Minghua Hu [b]

[a] Cranfield University, Cranfield, MK43 0AL, United Kingdom
[b] Nanjing University of Aeronautics and Astronautics, Nanjing, 210000, China

ARTICLE INFO

Keywords:
Demand and capacity balancing
Air traffic flow management
Multi-agent reinforcement learning
Heuristic algorithm
Deep q-learning network
Long short-term memory

ABSTRACT

Reinforcement learning (RL) techniques have been studied for solving the demand and capacity balancing (DCB) problem in air traffic management to exploit their full computational potential. Due to the lack of generalisation and the seemingly reduced optimisation performance affected by the training scenarios, it is challenging for existing RL-based DCB methods to be effectively applied in practice. This paper proposes a general multi-agent reinforcement learning (MARL) method that integrates a heuristic-based delay priority strategy to improve the efficiency of the solution and the generalisation of the model. The delay priority strategy is used to reduce the potential learning task and thus training difficulty. This study explores what features of the delay priority strategy are better suited to the MARL method. A long short-term memory (LSTM) network is integrated into a deep q-learning network (DQN) to ensure the model compatible with arbitrary DCB instances and to facilitate agents to identify key sectors. This study is conducted as a part of a large-scale European DCB research project, where real data from French and Spanish airspace are used for experimentation. Results suggest that the proposed method has advantages in generalisation, optimisation performance and computational performance over state-of-the-art RL-based DCB methods.

1. Introduction

Demand and capacity balancing (DCB) is one of the key components of air traffic management (ATM) (ICAO, 2008). If demand and capacity are not effectively balanced through an ATM system (e.g., on regional or national scales), it may lead to extensive congestion (demand over capacity) or inefficient use of airspace resources (capacity over demand) at a system level. In practice, the focus is more on situations where demand is higher than capacity. The importance of DCB has been increasing in recent decades due to the growing gap between the rapid growth of air traffic and the shortage of civil aviation airspace resources (Eurocontrol, 2022, 2020).

The outcome of DCB methods is usually to balance demand and capacity in the region of interest by controlling air traffic flow against airspace capacity. Therefore, DCB is also known as air traffic flow management (ATFM) (ICAO, 2016) or air traffic flow and capacity management (ATFCM) (Eurocontrol, 2021a). It is important to note that while the practical definition of DCB and ATFM may differ slightly (DCB typically used for tactical phase operations, while ATFM encompasses strategic, pre-tactical, and tactical phases), academic researchers often use DCB to represent a broader concept or to refer to ATFM. In this paper, the

* Corresponding author.
E-mail address: yanxu@cranfield.ac.uk (Y. Xu).

https://doi.org/10.1016/j.trc.2023.104218
Received 7 October 2022; Received in revised form 31 May 2023; Accepted 12 June 2023
0968-090X/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

term 'DCB method' refers to the ATFM method employed to balance demand and capacity. When a sector experiences an overload, meaning that demand exceeds capacity within a specific time window, it is referred to as a hotspot (Mannino et al., 2021). The 'DCB problem' in this paper pertains to the resolution of hotspots using the ATFM method. In practical application, demand and capacity are commonly balanced through demand management, such as ground delay program (GDP) (Montlaur and Delgado, 2020; Liu et al., 2019), rerouting (Diao and Chen, 2018; Murca, 2018), separation management (Huang et al., 2019; Zhang et al., 2019) and their combination (Sandamali et al., 2021; Guo et al., 2022).

In terms of solution methods, DCB methods are commonly categorised into two types: exact solution methods (e.g., integer programming Xu et al., 2020b,a) and approximate solution methods (e.g., heuristic algorithms (Xiao et al., 2018; Torres, 2012)). However, finding a satisfactory balance between optimisation and computing time poses a significant challenge for these methods. Exact solution methods exhibit excellent optimisation capabilities but suffer from long and unstable computing times, while approximate solution methods offer the opposite trade-off (Chen et al., 2023). Fortunately, recent research has revealed the potential of reinforcement learning (RL) to achieve satisfactory DCB solutions within acceptable timeframes (Spatharis et al., 2021; Huang and Xu, 2021). While RL methods do not guarantee optimal solutions, they possess the potential for superior optimisation performance compared to approximate solution methods based on heuristic algorithms. This advantage arises from RL methods' ability to adapt and learn the optimal policy without being excessively constrained by predefined rules.

However, there are still many challenges in applying RL methods to DCB problems. One of the challenges is the huge observation space, especially for a large-scale DCB problem (e.g., nationwide). An observation matrix of a nationwide DCB problem may contain millions of elements if the agent uses global observation (Chen et al., 2023). The huge observation space leads to great difficulty for agents to learn effective policies. Therefore, partial observation is commonly used in RL-based DCB methods (Huang and Xu, 2021; Tang and Xu, 2021), which creates another challenge: reduced accuracy of observation information. The loss of information makes it difficult for agents to identify critical flights and sectors to make appropriate decisions, reducing optimisation performance. Therefore, the efficiency of RL methods for solving DCB problems needs further improvement. Another challenge is the generalisation of RL methods, which has two meanings: whether the model can be deployed on arbitrary DCB instances (model scalability) and whether optimisation performance is significantly affected by the training data (dependency on data). For the model scalability, the arbitrariness of DCB problems mainly refers to the DCB problems with different scales, commonly including the number of flights and sectors. For a traditional DCB method, although the scale of the DCB problem may affect the model's performance, it does not usually affect the usability of the model. However, because of the neural network size for a specific RL-based DCB method, it is challenging to deploy the trained agent on a DCB problem with the scale mismatch with the neural network (Agogino and Tumer, 2012). Although some state-of-the-art RL-based DCB methods have made their models scalable, the cost is usually a significant reduction in optimisation performance (Huang and Xu, 2021; Tang and Xu, 2021; Chen et al., 2021). The existing methods are highly dependent on training data. The scenarios for their training data need to be as similar as possible to the DCB problem to be solved. Otherwise, they will likely cause unnecessary delays, which will be discussed in Section 2.1. The RL methods' advantage will be significantly limited if each specific DCB problem requires a bespoke model. Therefore, the generalisation of RL-based DCB methods is desired to be further enhanced.

This paper proposes a general reinforcement learning method which can solve DCB problem using ground delay as a demand management measure. The focus of this study is to verify the effectiveness of the proposed method. Further, its generalisation, optimisation performance and computational performance are studied to explore its potential for real-world applications. The main contributions of this study are summarised as follows:

1. The trained model is generalised to be applied to an arbitrary DCB instance, regardless of the region of the scenario and the scale of the problem, which is enabled partially by integrating a long short-term memory network encoding sector state information into the input layer of the deep q-learning network.
2. A heuristic delay priority strategy is incorporated into the multi-agent reinforcement learning framework, reducing the potential learning tasks for agents and thus reducing the learning difficulty. The study also reveals what features of the delay priority strategy are better suited to the framework.
3. Real-world data from Spain and France airspace (in addition to a toy case for comparison purpose) are used, which demonstrates that the realism of the training scenario affects the efficiency but not the effectiveness (i.e., resolving demand-capacity imbalances) of the model. Even models trained on a toy case can solve arbitrary DCB instances, yet the more realistic the training scenario, the better the optimisation performance of the model.
4. The experiments show that the proposed method outperforms the state-of-the-art RL-based DCB models (refer to Chen et al., 2023; Huang and Xu, 2021; Tang and Xu, 2021) in generalisation, optimisation and computational performance.

This paper is organised as follows. In Section 2, we briefly review related work and introduce our motivation for the study. Section 3 introduces the research context of the study and then formulates the research question addressed in this study. In Section 4, we discuss the setup of the reinforcement learning problem, the design of the delay priority strategy, the architecture of reinforcement learning, and the corresponding training method. Section 5 introduces the setup of experiments and then analyses the experimental results, including training and delay priority strategy comparison, model's generalisation, MARL-based DCB method comparison and the comparison with an optimisation method. The uncertainty and optimisation issues are further discussed in Section 6. Finally, conclusions and future work are summarised in Section 7.

2. Literature review

2.1. Related work

DCB technology has evolved over several decades, and many articles provide a good review of representative methods (Agustín et al., 2010; Kistan et al., 2017; Shone et al., 2021). As computer performance increases, RL techniques based on neural networks increasingly show advantages in solving high-dimensional complex problems. 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. The following discussion will focus on the representative RL-based DCB methods that have been reported. Based on the generalisation of the methods, existing research on the application of RL methods to DCB problems can be broadly divided into two phases: (1) no generalisation and (2) generalisation in a specific region.

In the first stage, no generalisation for RL-based DCB models has been validated. It could be considered that an RL model is designed to solve only a specific DCB problem. Crespo et al. proposed a single-agent RL model trained by Q-table to allocate flight delays to several airports centrally (Crespo et al., 2012). Because flights departing from the same airport were allocated the same amount of delay time, the optimisation performance of this method is limited. Due to the limitation of the Q-table, once the airports or sectors of interest have changed, the trained model is no longer applicable. Agogino and Tumer designed a distributed multi-agent decision-making system, where trained agents were deployed on route nodes around overloaded sectors to adjust the distance between flights passing through the same node by the Miles-in-trail (MIT) approach (Agogino and Tumer, 2012). Since the RL models were constructed based on route nodes, a trained model could only be deployed on a specific node. Kravaris et al. presented a collaborative multi-agent decision-making system, where an agent with an unshared policy was deployed on each flight (Kravaris et al., 2017). The effectiveness of this method has only been demonstrated in one tiny toy case. Spatharis et al. provided an enhanced model serving as an extension of Kravaris' method, and the effectiveness of the enhanced model has been verified in real-world scenarios with high complexity (Spatharis et al., 2018; Kravaris et al., 2018). However, the generalisation of the method has not been reported. Duong et al. integrated blockchain technology into RL models to solve DCB instances (Duong et al., 2019). Each agent was trained to allocate flight delays to a specific airport like Crespo's method, and all agents formed a multi-agent decision-making system. Due to the policy of each agent being unshared, the trained agent could not be deployed at airports not involved in training. Spatharis et al. proposed and extended a hierarchical MARL learning scheme for DCB problems, which constructed two states for agents: the ground state and abstract state (Spatharis et al., 2021, 2019). Chen et al. validated the effectiveness of DQN in the RL-based DCB method, but the experiments were not conducted with real-world data (Chen et al., 2021). At this stage, the RL-based DCB methods focus on how to use RL algorithms to obtain solutions for DCB problems rather than taking advantage of the RL method's rapid response to real scenarios. The validation of the generalisation of these methods has rarely been reported. Another common disadvantage is that none of these methods can guarantee that hotspots will be solved due to the limitation of action space or the non-stationarity in MARL.

As a result, generalisation has been further considered in the design of the RL-based DCB methods, especially the model's scalability. Trained RL models started to be tested in DCB problems with unseen scenarios. Some decision-making frameworks for DCB were designed to guarantee that hotspots will be solved. Tang and Xu proposed a time-step-based multi-agent decision-making framework for DCB, and Proximal Policy Optimisation was employed to train agents (Tang and Xu, 2021). The observation matrix was constructed based on all sectors of interest. One of the dimensions of the observation matrix is the same as the number of sectors of interest. Although partial observation was employed, the observation matrix remained large and sparse, making it difficult for agents to learn excellent policy and causing unnecessary delays. Thus, an action supervisor had to be integrated into the framework. The action supervisor forced the agent to change when it chose an action that would cause a hotspot to ensure that hotspots would be solved. Due to the limitation of the observation matrix, a trained model could only solve the DCB problem whose sectors were all included in its training scenarios. For example, assume one wants to use the model trained by DCB problems with Spanish scenarios to solve the DCB problems with French scenarios. While the observation matrix can be constructed for the model based on all sectors of France and Spain, the part of the neural network related to French sectors will not be updated appropriately. That would likely significantly reduce optimisation performance (such inference will be confirmed in the experiment in Section 5.5).

Likewise, Huang and Xu presented a hotspot-based DCB method with Asynchronous Advantage Actor-Critic (Huang and Xu, 2021). Because recurrent neural networks were compatible with input matrices of different dimensions, the trained model could be deployed on DCB problems with arbitrary scales. However, the sector number was taken as observation information to help agents identify sectors. Although the sectors of unseen DCB problems can also be numbered, significant reductions in optimisation performance will likely occur if the trained model is used for unseen DCB problems (such inference will be confirmed in the experiment in Section 5.5). That is because the sector number has no physical meaning in calculating input for neural networks. Chen et al. designed a distributed multi-agent decision-making system for DCB problems where each flight was deployed with a customised neural network (Chen et al., 2023). Although the customised agent adapts to the specificity of the different flights, the model requires a strict range of training scenarios. For example, the trained model can efficiently solve DCB problems with French scenarios if the training scenario contains all possible flights passing through France. However, ensuring that all possible future flights exist in the training dataset is challenging, and the trained model cannot be applied to regions other than those included in the training scenario. At this stage, RL-based DCB methods are functionally capable of solving DCB problems with different scales but are limited to a specific region (related to the training scenarios). Besides, it is observed that scalability usually comes at the cost of reduced optimisation performance.

2.2. Motivation

As reviewed above, for the state-of-the-art RL-based DCB methods, it might still be challenging to directly deploy the trained models to solve DCB problems in practice due to the limited generalisation capability and the low optimisation performance in many cases. As such, it is desired to overcome those drawbacks to improve the relevant research's technical readiness level (TRL). Specifically, this study serves as an extension to our previous work on the RL-based DCB method (Huang and Xu, 2021), aiming at improving its generalisation, optimisation and computational performance. From a present-day perspective, various aspects of our previous method could be improved, although it was advanced to some extent when first proposed. For example, our previous method is not completely scalable, and the optimisation performance is overly dependent on particular training scenarios. Also, the continuous action space leads to excessive iterations, thus increasing the solution time of the trained model. Due to the observation matrix and neural network design, it has a great potential for improvement in identifying critical sectors across the network, just to name a few.

This study has been conducted under the umbrella of a set of holistic research initiatives. We have considered the key factors as realistic as possible, such as the sector opening schemes and weather-affected capacity reductions. A multi-agent decision-making framework based on an iteration mechanism is designed to deal with the problems arising from non-stationarity in MARL. The proposed method is guaranteed to solve all the hotspots. Concretely, the agent observes the demand and capacity of the time window in which the flight crosses the sector and that of the next time window for a longer-term perspective. The action space is designed as a discrete space with an accuracy that matches the practical application. Such action modelling is expected to reduce the number of iterations and thus reduce the solution time. Some delay priority strategies are applied to select the flights that are key to solving the DCB problem. In such a case, agents would only need to learn to allocate proper delays with regard to the selected flight, whereas there is no need to learn how to identify the critical flights. That could be a possible way to cope with the difficulty of agent learning to solve large-scale DCB problems. In detail, the delay priority strategy is devised based on a heuristic algorithm, where four models are designed and compared to explore which one best matches the MARL method. A DQN is then employed for reinforcement learning, which is enhanced by Double DQN (Van Hasselt et al., 2016), Duelling DQN (Wang et al., 2016) and Prioritised Replay Buffer (Schaul et al., 2015). In fact, we decompose the DCB problem into a sequence of sub-problems for solving. In each sub-problem, the proposed method performs two operations:

1. 'cherry picking': selecting the flights that should be delayed first (through the delay priority strategy); and
2. delay allocation: allocating a specific delay time to each selected flight (through the MARL).

An LSTM network is integrated into the input layer of the DQN, and its functions are mainly two-folded. One is to make the model compatible with DCB instances on different scales. The other is to ensure that the observations of a sector correspond one-vs-one to an LSTM cell, thus using the characteristics of the LSTM to let the agent learn to identify critical sectors. Although our previous model also has a recurrent neural network (Huang and Xu, 2021), it is fed with all observations of all flights into one network, resulting in considerable learning burdens, which is different from the network used in this paper.

3. Demand and capacity balancing problem

This section first introduces the research context of this study. Then, the DCB problem to be solved is defined to clarify the research scope of this study further. Next, a multi-agent decision-making framework is proposed, and the DCB problem is formulated to build the basis of the RL environment.

3.1. Research context

This study was performed as a part of a SESAR exploratory research project: Artificial Intelligence Solutions to METEO-Based DCB Imbalances for Network Operations Planning (ISOBAR). In the ISOBAR framework (as shown in Fig. 1 (SESAR, 2021)), there are four artificial intelligence (AI) components: METEO (meteorology) AI component, hotspot detection AI component, AU (airspace users) preference AI component and hotspot mitigation AI component. The METEO AI component will enable airspace managers to obtain a probabilistic forecast of convective weather in advance. The hotspot detection AI component consists of three sub-modules: the demand, capacity and imbalance modules. On the capacity side, the capacity module will predict reductions in sector capacity due to convective weather and other events that may affect capacity. On the demand side, the demand module will provide a demand prediction based on the 4D trajectories derived from historical AU behaviours. The imbalance module will input those predictions from both the demand and capacity modules and provide predictions on areas of the network with demand-capacity imbalance, namely the hotspots. In addition, the AU preference AI component will provide some optional (rerouting) trajectories for specific flights aligned with their historical choices. Finally, once the hotspots have been identified, the hotspot mitigation AI component will come into play and provide an optimisation solution to offload the hotspots, which is exactly the focus of this study. The ISOBAR operational framework also extends the implementation of the above pipeline to different phases (i.e., the D-1 and D0 day of operation). The reader is directed to ISOBAR ConOps for more details.

The DCB problem to be handled in this study is to ensure that any sector's demand does not exceed its capacity (i.e., all hotspots are solved) by implementing the Ground Delay Program during the pre-tactical phase, where the demand refers to the entry count (i.e., the number of flights entering any sector per unit time). It is assumed that the initial traffic demand (flight sequence) is

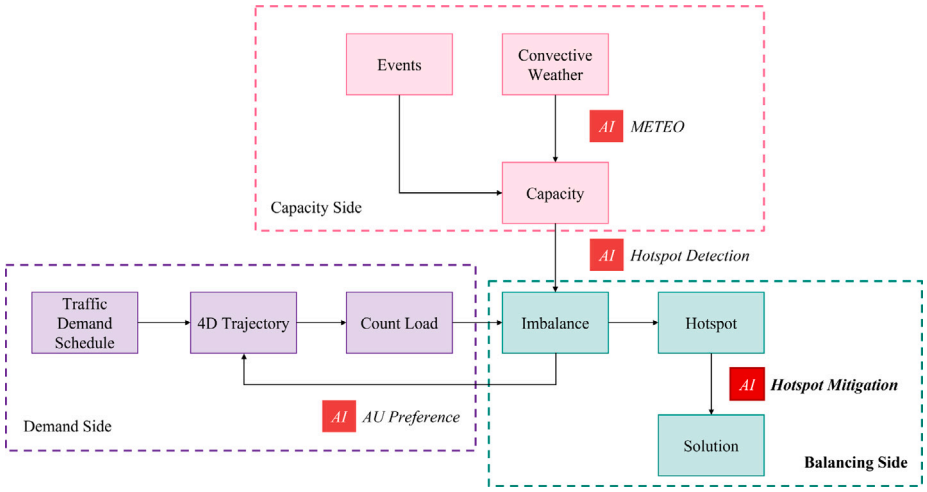


Fig. 1. Research context from the ISOBAR project, where this study focuses specifically on the hotspot mitigation AI component.

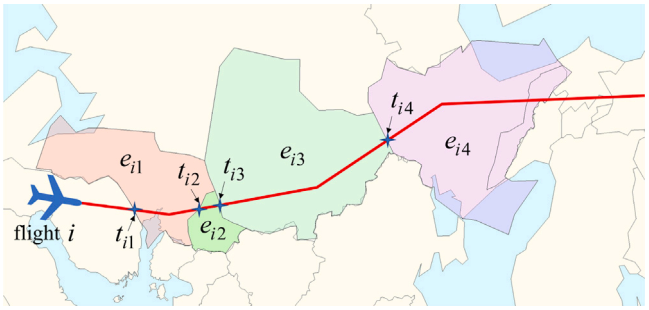


Fig. 2. Diagram of a flight sequence crossing associated sectors. The four-pointed star indicates the position of the flight into the sector.

available on the D-1 day of operation. f_i denotes the flight sequence of the ith flight. The flight sequence consists of a set of binary tuples (as shown in Fig. 2):

$$f_i = \{(e_{ij}, t_{ij}) \,|\, j = 1, 2, \ldots, M_i\} \tag{1}$$

where e_{ij} denotes the j-th sector through which the ith flight is scheduled to pass, t_{ij} denotes the time of entry into that sector, and M_i denotes the number of sectors through which the ith flight is scheduled to pass. The initial traffic demand (flight sequences) $\mathbb{F}_{\text{Initial}}$ is represented as:

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

where I denotes the set of flights.

A scenario for a specific time range in a specific region is constructed as a DCB problem. The time range can usually be several hours, the region being several air traffic control regions. For a DCB problem, only the flight segments of the time range of interest are considered. The time range of interest is divided into several time windows whose width is typically 20 min. For example, if time starts at 00:00, the first time window is 00:00–00:20, and if the time interval is also set to 20 min (which can be lower than that as well), then there will be 72 time windows for an operation day. The time window/interval is the basic unit used to detect hotspots and balance the demand and capacity of a sector.

In practice, sectors are divided into two types, elementary sectors and collapsed sectors. An elementary sector is the fundamental sector unit in the airspace, while a collapsed sector comprises several adjacent elementary sectors. The fundamental sector units operate as elementary or collapsed sectors at specific times, depending on the sector opening scheme. In this study, the sector opening scheme is considered, and the state of the sectors changes over time windows. For each time window, a sector (either elementary or collapsed sector) has a specific demand D_{jw} and capacity C_{jw}, provided respectively by the demand module and capacity module of the hotspot detection AI component. D_{jw} and C_{jw} denote the demand and capacity of the jth sector in the wth time window.

As an example, see an illustrative sector opening scheme shown in Fig. 3. 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 comprises e_2 and e_3. e_6 comprises e_3 and e_4 (we have omitted flight numbers i in e and t for

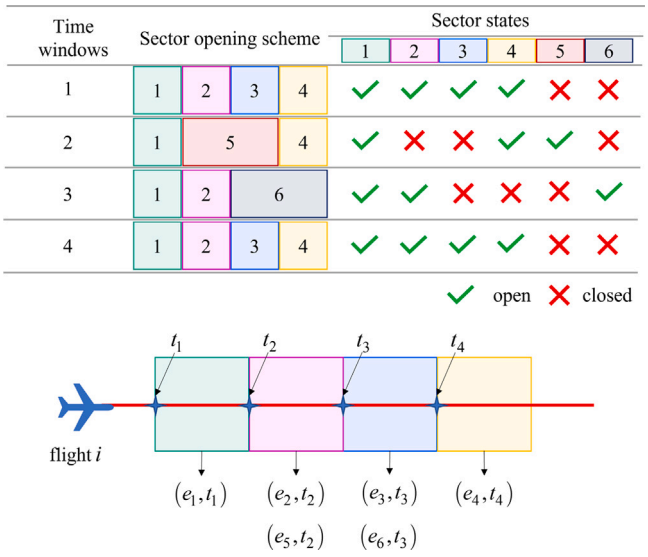


Fig. 3. Diagram of sector opening scheme.

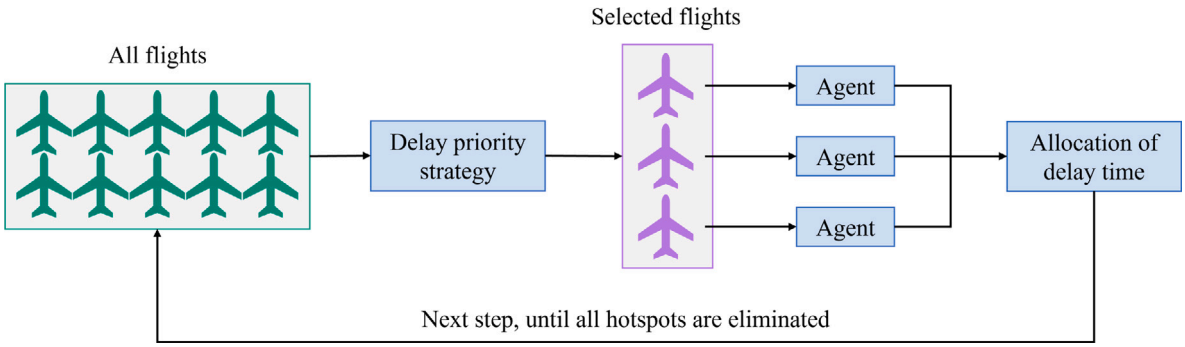


Fig. 4. Multi-agent decision-making framework where all agent are distributed.

legibility). The ith flight is scheduled to pass through the four elementary sectors in turn and the corresponding entry times are t_1, t_2, t_3 and t_4. f_i will be represented as $\{(e_1,t_1),(e_2,t_2),(e_5,t_2),(e_3,t_3),(e_6,t_3),(e_4,t_4)\}$. For calculating the demand D_{jt}, only the time when the flight enters the sector is considered, regardless of how long it will be in that sector. For example, if the ith flight is scheduled to enter the jth sector in the wth time window and leave in the $(w+2)$-th time window, it is only as the demand of the jth sector in the wth time window, but not the $(w+1)$-th and $(w+2)$-th time windows. If the jth sector is closed in the wth time window, C_{jw} and D_{jw} are equal to zero. $\mathbb{S}$ denotes the sector opening scheme where J denotes the set of sectors and T denotes the set of time windows:

$$\mathbb{S} = \{C_{jw} \mid j \in J, w \in W\} \tag{3}$$

In summary, the DCB problem in this study is defined as balancing the traffic demand against the airspace capacity considering sector opening schemes, through shifting the flight take-off time so as to mitigate all the hotspots. The optimisation objective is to minimise the total delay time for all flights. Note that, as this paper focuses on verifying the feasibility of the method, the proposed model does not consider uncertainty in real-world DCB instances. However, how the proposed method can be extended to deal with uncertainties will be discussed in Section 6.1.

3.2. Problem statement

A multi-agent decision-making framework was proposed to tackle the DCB problem through the MARL method (as shown in Fig. 4). In the framework, a delay priority strategy (refer to Section 4.3) was designed to select flights which would be allocated delay time. An agent determines the length of the delay time. This process is performed several times until all hotspots have been solved.

The DCB problem was formulated as a partially observable decision-making problem in our method. A flight at the tth step has access to an observation $\mathbf{o}_t^i$, which only provides partial knowledge about the environment and computes a decision-making action a_t^i that tries to avoid hotspots. The observation $\mathbf{o}_t^i$ is drawn from the global state $\mathbf{s}_t^i$ ($\mathbf{o}_t^i \sim \mathcal{O}(\mathbf{s}_t^i)$), and it only provides partial information about the state $\mathbf{s}_t^i$. In our model, it is possible to obtain the global state $\mathbf{o}_t^i$, but if so, the size of the global state will be huge. For example, the global state $\mathbf{o}_t^i$ is represented by an $M \times N$ matrix where M and N denote the number of sectors and flights, respectively, and the elements in the matrix are t_{ij}. If so, e.g., in a scenario with 10000 flights and 400 sectors, the size of the global state is 10000×400, i.e., 4×10^6 elements are contained. Even if we compress the data, for example, the global state $\mathbf{o}_t^i$ is represented by an $N \times P \times 2$ matrix where P denotes the number of time windows, and two elements corresponding to the specific sector of a time window are its capacity and the current demand. If so, e.g., in a scenario with 400 sectors and 72 time windows of a day, the size of the global state is $400 \times 72 \times 2$, i.e., 5.76×10^4 elements are contained, which is still huge. Although we could have reduced the size of the global state by using convolutional neural networks, our previous research has found that doing so gave unsatisfactory results. That might be due to the loss of crucial information in the global state abstraction during the convolution process, severely limiting decision optimisation. Hence, partial observation was applied in our method to keep the size of the observation matrix within an acceptable range.

Given the partial observation $\mathbf{o}_t^i$, each flight independently computes a decision-making action a_t^i, sampled from its specific policy $\boldsymbol{\pi}_{\theta_i}^i$, where θ_i denotes the policy parameters:

$$a_t^i \sim \boldsymbol{\pi}_{\theta_i}^i \left(a_t^i \left| \mathbf{o}_t^i \right.\right) \tag{4}$$

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

$$\boldsymbol{\Pi}_{\Theta} = \left\{ \boldsymbol{\pi}_{\theta_i}^i \left| i \in I \right. \right\} \tag{5}$$

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

$$\boldsymbol{\Pi}_{\Theta}\left(\mathbb{F}^{\text{Initial}} \left| \mathbb{S} \right.\right) = \left\{ \Delta t_i \left| i \in I \right. \right\} \tag{6}$$

$f_i\left(\Delta t_i\right)$ denotes the flight sequence of the ith flight with the implementation of GDP:

$$f_i\left(\Delta t_i\right) = \left\{ \left(e_{ij}, t_{ij} + \Delta t_i\right) \left| j \in J_i \right. \right\} \tag{7}$$

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

$$\mathbb{F} = \left\{ f_i(\Delta t_i), i \in I \left| \begin{array}{l} \Delta t_i \sim \boldsymbol{\Pi}_{\Theta}\left(\mathbb{F}^{\text{Initial}} \left| \mathbb{S} \right.\right), j \in \boldsymbol{J}, w \in \boldsymbol{W} : \\ \boldsymbol{D}_{jw} \le \boldsymbol{C}_{jw} \end{array} \right. \right\} \tag{8}$$

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

$$\arg\min_{\boldsymbol{\Pi}_{\Theta}} \mathbb{E}\left[\frac{1}{N} \sum_{i \in I} \Delta t_i \left| \boldsymbol{\Pi}_{\Theta} \right. \right] \tag{9}$$

where N is the number of flights. The average delay time will also be used as an important metric to evaluate the trained policy in Section 5.3.

4. Approach

This section first introduces the reinforcement learning setup. Then, we present the network architecture, which focuses on the deep q-learning algorithm integrating a long short-term memory network. Next, the pseudo-code for reinforcement learning to solve the formulated problem is shown to illustrate the exact operational flow of the method. Lastly, a heuristic-based delay priority strategy is introduced, which contains four optional priority assessment models and a limited delay mechanism.

4.1. Reinforcement learning setup

Based on the problem statement in Section 3.2, we construct a MARL environment. In this environment, the agent is deployed on each aircraft, forming a distributed decision-making system. The action space, state space and reward function are defined as follows:

4.1.1. Action space

Since the lowest unit of flight delay time is usually minutes, a discrete action space was adopted in this model. The action space $\mathbf{A}$ for each agent is denoted as:

$$\mathbf{A} = \left\{1, 2, \ldots, T_{\text{W}}\right\} \tag{10}$$

where T_{W} is the width of a time window. a_t^i denotes the action of flight i at the tth step, and therefore $a_t^i \in A$. For example, $a_t^i = 5$ means that the departure of flight i is delayed by 5 min based on its current flight sequence. This action space allows the agent to shift the entry time of a flight into any sector to the next time window.

4.1.2. State space

In our model, the observation $\mathbf{o}_t^i$ consists of the current state of each sector that flight i is scheduled to pass through:

$$\mathbf{o}_t^i = \left\{ \boldsymbol{\Phi}_{ij}^t \,|j \in J_i \right\} \tag{11}$$

where $\boldsymbol{\Phi}_{ij}^t$ denotes the partial observation (inputted state for the agent) of sector j about flight i at step t. $\boldsymbol{\Phi}_{ij}^t$ consists of the demand and capacity of the time window when flight i entry into the sector, those of the next time window, and an entry timestamp:

$$\boldsymbol{\Phi}_{ij}^t = \left[d_{jw_{ij}^t}^t, c_{jw_{ij}^t}^t, d_{j\left(w_{ij}^t+1\right)}^t, c_{j\left(w_{ij}^t+1\right)}^t, p_{jw_{ij}^t}^t \right] \tag{12}$$

$d_{jw_{ij}^t}^t$ and $c_{jw_{ij}^t}^t$ denote sector j's demand and capacity in time window w_{ij}^t at step t. w_{ij}^t represents the time window when flight i will enter sector j if it departs as scheduled at step t. Hence, $d_{j\left(w_{ij}^t+1\right)}^t$ and $c_{j\left(w_{ij}^t+1\right)}^t$ can be used to respectively denote the demand and capacity of the next time window of time window w_{ij}^t at step t. The entry timestamp $p_{jw_{ij}^t}^t$ indicates the entry time based on the time window range. For example, at step t, if flight i enters sector j at 14:12 and the range of time window w_{ij}^t is 14 : 00–14 : 20, the entry timestamp $p_{jw_{ij}^t}^t$ equals 12.

In summary, observation $\mathbf{o}_t^i$ in matrix form can be represented as:

$$\mathbf{o}_t^i = \begin{bmatrix} \boldsymbol{\Phi}_{i1}^t \\ \boldsymbol{\Phi}_{i2}^t \\ \vdots \\ \boldsymbol{\Phi}_{iM_i}^t \end{bmatrix} \tag{13}$$

that is,

$$\mathbf{o}_t^i = \begin{bmatrix} d_{1w_{i1}^t}^t & c_{1w_{i1}^t}^t & d_{1\left(w_{i1}^t+1\right)}^t & c_{1\left(w_{i1}^t+1\right)}^t & p_{1w_{i1}^t}^t \\ d_{2w_{i2}^t}^t & c_{2w_{i2}^t}^t & d_{2\left(w_{i2}^t+1\right)}^t & c_{2\left(w_{i2}^t+1\right)}^t & p_{2w_{i2}^t}^t \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{M_iw_{iM_i}^t}^t & c_{M_iw_{iM_i}^t}^t & d_{M_i\left(w_{iM_i}^t+1\right)}^t & c_{M_i\left(w_{iM_i}^t+1\right)}^t & p_{M_iw_{iM_i}^t}^t \end{bmatrix} \tag{14}$$

In the observation matrix, each row represents a sector, arranged in the order of the flights passing through. Agents are expected to learn by the demand and capacity whether the flight should be shifted from the current time window to the next one of the sector. Not only are demand and capacity observed for time window w_{ij}^t, but also for time window $\left(w_{ij}^t + 1\right)$ in our model. That is because when the next time window of the hotspot a flight passes through is also a hotspot (especially if the next time window is more overloaded), shifting the flight to the next time window at the current step may not be the best choice. Agents are also expected to learn by the entry timestamp of each sector the flight pass through how many minutes of delay it will take. For sectors that pass through, flights are expected to be shifted to the next time window in some sectors while be remained in the current time window in other sectors. The agent needs to find a suitable delay time to meet this expectation as much as possible.

4.1.3. Reward function

Homogeneous agents are considered, with the same objective to avoid overload and minimise delay. A reward function was designed to guide agents to achieve this objective:

$$r_t^i = a \times \left(r^{\text{Overload}}\right)_t^i + b \times \left(r^{\text{Delay}}\right)_t^i \tag{15}$$

Then reward r_t^i received by flight i at step t considers overload $\left(r^{\text{Overload}}\right)_t^i$ and delay $\left(r^{\text{Delay}}\right)_t^i$. a and b are weighting factors. Specifically, $\left(r^{\text{Overload}}\right)_t^i$ is used to encourage the agent to avoid overload. It is represented as the difference in the sum of the number of overloads in sectors passed by the flight:

$$\left(r^{\text{Overload}}\right)_t^i = \sum_{j \in J_i} u_{jw_{ij}^{t+1}}^{t+1} - \sum_{j \in J_i} u_{jw_{ij}^t}^t \tag{16}$$

$u_{jw_{ij}^t}^t$ denotes the number of overloads of sector j in time window w_{ij}^t:

$$u_{jw_{ij}^t}^t = \Re\left(d_{jw_{ij}^t}^t - c_{jw_{ij}^t}^t\right) \tag{17}$$

where $\Re$ is the Rectified Linear Unit (ReLU), that is, $\Re(x) = \max(x, 0)$. For sector j, $u_{jw_{ij}^t}^t$ is the number of overloads when the demand exceeds capacity; otherwise, $u_{jw_{ij}^t}^t$ is 0. $\left(r^{\text{Overload}}\right)_t^i$ is used to encourage the agent to minimise the delay. It is represented as the delay time of flight i at step t:

$$\left(r^{\text{Delay}}\right)_t^i = h_t^i \tag{18}$$

where h_t^i is the delay time of flight i at step t. Since in our model the action number means the time of delay, $h_t^i = a_t^i$.

In Eq. (15), a and b are negative. If the number of overloads is decreased by action a_t^i, $a \times \left(r^{\text{Overload}}\right)_t^i$ will be positive; if increased, negative. The greater the variation in the number of overloads, the greater the corresponding reward or penalty of overloads. $b \times \left(r^{\text{Delay}}\right)_t^i$ is always negative, and thus it is a penalty. The longer the delay, the greater the penalty of the delay. Considering $\left(r^{\text{Overload}}\right)_t^i$ serves as a constraint and $\left(r^{\text{Delay}}\right)_t^i$ serves as an optimisation objective in the RL model, the absolute value of a should be much greater than that of b. In the case of this study, we set $a = -10$ and $b = -1$. As the main purpose is to verify the feasibility of the proposed method, this paper only provides the experiential value of this study for readers' reference, which may not be the optimal value. The systematic parameter study will be the future research work.

4.2. Reinforcement learning architecture

As we intended to make the RL-based DCB solver as scalable as possible, the neural network parameters would be shared between the homogeneous agents. In general, parameters can only be shared between neural networks of the same size. However, in our model, the observation matrix size is determined by the number of sectors the flight pass through, so it varies from flight to flight (refer to Eq. (14)). Therefore, long short-term memory was applied to tackle the problem of varying input sizes. The observation matrix of the flight can be inputted into the LSTM neural network as a sequence of sectors since each sector's observation matrix size is the same. Due to the feature of LSTM enhanced by attention mechanism being able to retain only important information in the input sequence (Hochreiter and Schmidhuber, 1997), for example, in the case where the agent finds a delay of four minutes is the best choice based on the observation information for sector one and seven minutes for sector two, it will likely forget the information about sector one and end up with a seven-minute delay as an output action (since a delay of seven minutes can solve both hotspots). Moreover, considering the action space in the model is discrete, Deep Q-learning Network was employed for parameter update. In our method, all agents share neural network parameters.

The reinforcement learning architecture is shown in Fig. 5. First, the delay priority strategy (which will be introduced in Section 4.3) selects several flights based on the current environment. Then, the observation matrix $\mathbf{o}_t^i$ of each selected flight is obtained, and the observation information of each sector $\boldsymbol{\Phi}_{ij}^t$ in it is inputted into the LSTM in turn. Next, the last LSTM cell's output is hidden layer 1 of the DQN neural network. Following hidden layer 1, hidden layers 2A and 2V are used to respectively represent the advantage function and value function for the Duelling DQN technology (which will be detailed in Section 4.4). The neuron values in the output layer are obtained by a linear summation of the neuron values in hidden layers 2A and 2V (refer to Algorithm 2, line 9). Next, action a_t^i is obtained by the output layer of the DQN neural network. After all selected flights have taken an action, they are delayed by the corresponding action, and the environment is thus updated. Finally, reward r_t^i is generated from the latest environment and used to update the neural network parameters (i.e., DQN&LSTM). If there are still hotspots, the next step is executed until all hotspots have been solved.

4.3. Delay priority strategy

A delay priority strategy was designed to select the appropriate number of flights that would yield more significant potential benefits (i.e., more likely to reduce overloads) at each step. The delay priority strategy combines two parts, a priority assessment model and a limited delay mechanism. We could also let the agent decide whether to delay itself at each step, i.e., removing the delay priority strategy and set $\mathbf{A} = \{0, 1, \ldots, T_W\}$ (note that it is different from Eq. (10)). Through experiments (where no delay priority strategy was applied), we have found that some agents that do not need to be delayed often delay themselves, resulting in a reduction in on-time performance and an increase in total delay time. This study provided four priority assessment models and one limited delay mechanism. Thus, by combining them, there are four delay priority strategies.

4.3.1. Priority assessment model

Priority assessment models are used to assess which flights are potentially more effective in reducing overloads by delaying them in the proposed multi-agent decision-making framework (refer to Fig. 4). The design of the priority assessment models considers two questions. One relates to the scope of the assessed time window: i.e., which time window is assessed for a given flight through a sector? The other is the importance of each hotspot: i.e., is each hotspot weighted equally?

For the scope of assessed time windows, we proposed two methods. A seemingly reasonable thought is that the higher the number of hotspots a flight passes, the more priority it should be delayed. Therefore, assessing the time window when the flight enters the sector is a method that can be considered. However, as mentioned in Section 4.1.2, if the next time window of the hotspot is also a hotspot, shifting the flight at the current step may not be the best choice. Taking the scenario in Fig. 6 as an example, flight 1 passes through two hotspots (time window 1 of sector 1 and time window 4 of sector 4) while flight 2 passes through only a hotspot (time window 1 of sector 1). If only the situation of the time window in which the flight enters the sector is used as a reference, it seems sensible to give priority to delaying flight 1. However, if the entry times of both flights are shifted to the corresponding next time windows in all sectors they pass through, flight 1 will pass through three hotspots (time window 3 of sector 2, time window 5 of sector 4 and time window 7 of sector 6) while flight 2 will not pass through any hotspot. As a result, the number of hotspots passed by flight 1 will increase by two, while those passed by flight 2 will decrease by one. In this case, prioritising delaying flight 2 seems to be the better choice at the current step. Thus, assessing the time window when the flight enters the sector and the next time window is another method that can be considered.

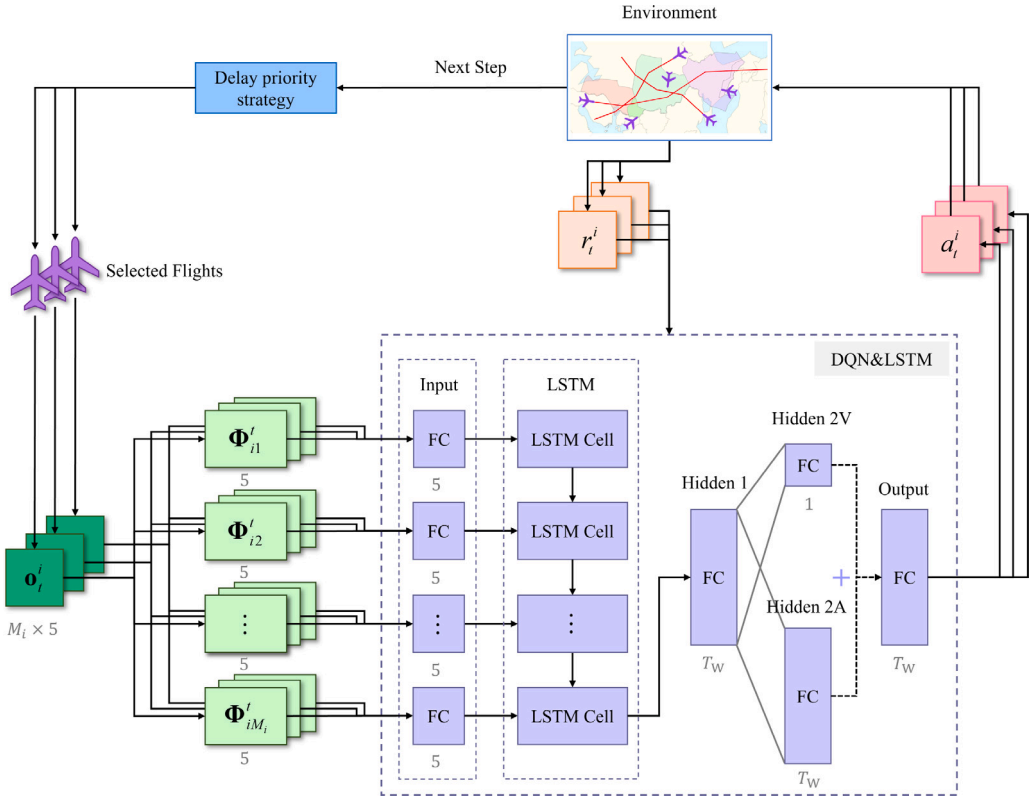


Fig. 5. Reinforcement learning architecture. The DQN&LSTM network is shared between agents. FC means a fully connected layer. The grey formula and numbers below some boxes denote the corresponding matrix size or layer size.

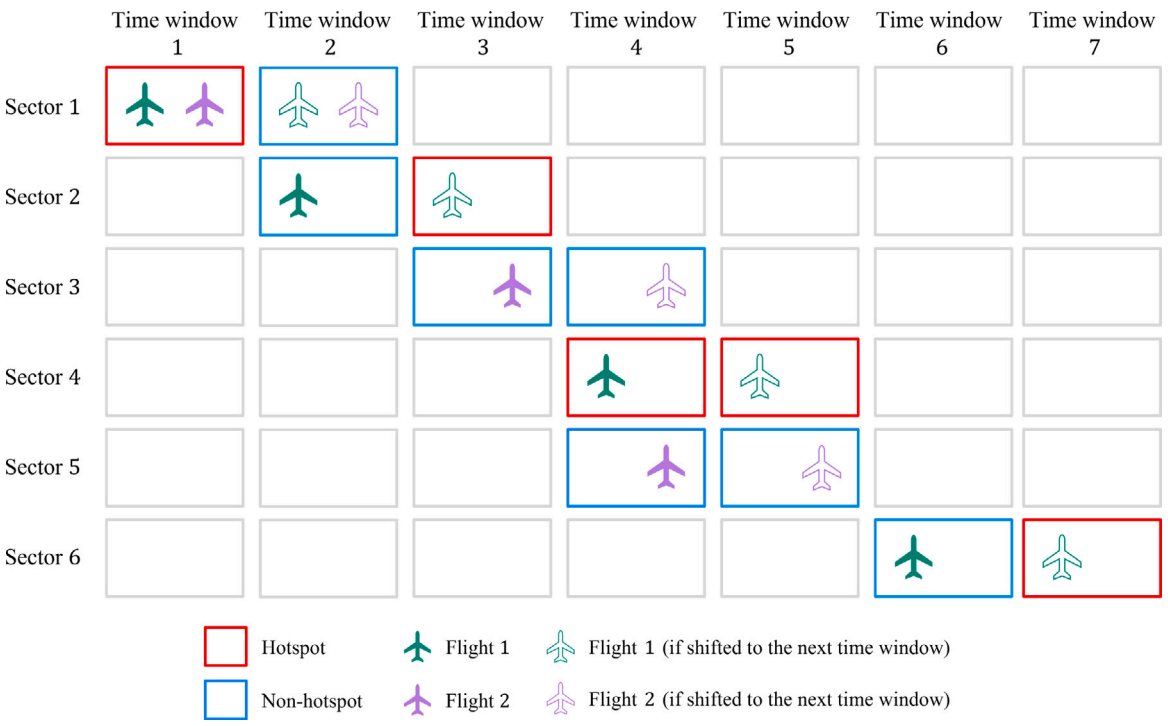


Fig. 6. An example to illustrate the importance of assessing the situation of the next time window of flight entering.

Table 1
Features of priority assessment models.

Priority assessment models	Scope of assessed time windows	Importance of each hotspot	Equations
1	Only the time window of flight entering	Consistency	Eq. (19)
2	Only the time window of flight entering	Inconsistency	Eq. (20)
3	The time window of flight entering and the next one	Consistency	Eq. (21)
4	The time window of flight entering and the next one	Inconsistency	Eq. (22)

For the importance of each hotspot, it seems possible to regard them as either consistent or inconsistent. On the one hand, if we consider reducing the number of hotspots, the flight that crosses more hotspots should be prioritised for delays in removing hotspots. For this reason, each hotspot's importance is preferred to be considered consistent. On the other hand, if a flight passes through a hotspot with a higher overload ratio (the ratio of demand to capacity), it is more expected to be prioritised for delays, thus relieving the highly congested sector as soon as possible. For this reason, each hotspot's importance is preferred to be considered inconsistent.

Based on the above discussion, we presented four priority assessment models and their features are summarised in Table 1. They were formulated by Eqs. (19), (20), (21) and (22), respectively:

$$\left(\varsigma_i^t\right)_1 = \sum_{j\in J_i} k^t_{jw^t_{ij}} \tag{19}$$

$$\left(\varsigma_i^t\right)_2 = \sum_{j\in J_i} \aleph\left[\frac{k^t_{jw^t_{ij}} d^t_{jw^t_{ij}}}{c^t_{jw^t_{ij}}}, m^-, m^+\right] \tag{20}$$

$$\left(\varsigma_i^t\right)_3 = \sum_{j\in J_i} \left(k^t_{jw^t_{ij}} - k^t_{j(w^t_{ij}+1)}\right) \tag{21}$$

$$\left(\varsigma_i^t\right)_4 = \sum_{j\in J_i} \aleph\left[\left(\frac{k^t_{jw^t_{ij}} d^t_{jw^t_{ij}}}{c^t_{jw^t_{ij}}} - \frac{k^t_{j\left(w^t_{ij}+1\right)} d^t_{j\left(w^t_{ij}+1\right)}}{c^t_{j\left(w^t_{ij}+1\right)}}\right), m^-, m^+\right] \tag{22}$$

where $\left(\varsigma_i^t\right)_1$, $\left(\varsigma_i^t\right)_2$, $\left(\varsigma_i^t\right)_3$ and $\left(\varsigma_i^t\right)_4$ denote the scores of flight i at step t by *Priority assessment model 1*, *Priority assessment model 2*, *Priority assessment model 3* and *Priority assessment model 4*, respectively. $k^t_{jw^t_{ij}}$ denotes whether time window w^t_{ij} of sector j is a hotspot at step t. If it is a hotspot, $k^t_{jw^t_{ij}} = 1$; if not, $k^t_{jw^t_{ij}} = 0$. $\aleph$ is a clip function, and it is used to limit the impact of a single sector on the overall score:

$$\aleph(x,a,b) = \begin{cases} a & x < a \\ b & x > b \\ x & \text{otherwise} \end{cases} \tag{23}$$

In this study, $m^- = -2$ and $m^+ = 2$. Compared with *Priority assessment model 1* and *Priority assessment model 3*, *Priority assessment model 2* and *Priority assessment model 4* further consider the overload ratio. Assume that sector j's time windows w^t_{ij} and $\left(w^t_{ij}+1\right)$ are both hotspots at step j. If the overload ratio of time window w^t_{ij} is greater than that of time window $\left(w^t_{ij}+1\right)$, the corresponding flight tends to be delayed at step t; if smaller, it will be delayed with a lower priority.

4.3.2. Limited delay mechanism

A limited delay mechanism decides which flights will be delayed at a given step. If the number of delayed flights is too high at a step, more unnecessary delays may be incurred; if too low, optimisation performance may be limited. Therefore, at each step, only the necessary number of flights are selected to be delayed. 'Necessary' means that it is just enough to solve all the hotspots if 'cancel' all the selected flights. It is not a genuine 'cancel', but delete the demand of these flights when detecting hotspots. In other words, agents can solve all hotspots in the current step if there is sufficient remaining capacity in the next time window for all sectors through which these selected flights pass. Note that this is ideal, as some sectors may not have enough remaining capacity, even if they are also hotspots. Nonetheless, this idea of selecting flights is still reasonable.

The limited delay mechanism was designed as shown in Algorithm 1, selecting flights based on each hotspot. First, scores ς_i^t of each flight are inputted, and the set of selected flights I^t_{Selected} is initialised (line 1). Then, each time window of each sector is checked. If it is a hotspot, $u^t_{jw^t_{ij}}$ flights passing through the hotspot with the highest ς_i^t are selected (line 5). After all the hotspots are checked, the set of selected flights I^t_{Selected} is finally outputted.

In line 5 of Algorithm 1, there could be several flights with the highest score. If one of them is selected randomly, the method will have uncertainty. Since the proposed method aims to be applied in practice where the stability of the methods is one of the priority issues, to solve the uncertainty, we follow the following priority selection rules in order: the flight who (a) passes through

Algorithm 1 Limited delay mechanism

Input: Scores ς_i^t, $i \in I$, t is the current step
1: Initialise the set of selected flights I^t_{Selected}
2: **for** sector j in J **do**
3: **for** time window w in W **do**
4: **if** time window w of sector j is a hotspot **then**
5: Append $u^t_{jw_{ij}^t}$ flights $i \in I^t_{jw}$ with the highest ς_i^t into I^t_{Selected}, where I^t_{jw} is the set of flights whose entry time of sector j is in time window w at step t
6: **end if**
7: **end for**
8: **end for**
Output: I^t_{Selected}

more hotspots, (b) passes through more sectors, (c) has an earlier scheduled departure time and (d) has a smaller flight number will be selected first.

To sum up, four delay priority strategies were designed in this study, each consisting of a specific priority assessment model and the limited delay mechanism. They are named *Delay priority strategy 1*, *Delay priority strategy 2*, *Delay priority strategy 3*, and *Delay priority strategy 4*. The number corresponds to the specific priority assessment model.

4.4. Training method

The DQN-based reinforcement technologies, such as Double DQN (Van Hasselt et al., 2016), Duelling DQN (Wang et al., 2016) and Prioritised Replay Buffer (Schaul et al., 2015), were used in this study to improve the learning efficiency. The corresponding algorithm is as shown in Algorithm 2. It contains two main parts, i.e., simulation & collecting transitions (lines 4–19) and neural network update (lines 21–23). Since all agents share neural network parameters, the neural network $\boldsymbol{\pi}^i_{\theta_i}$ (refer to Eq. (4)) is written as $\boldsymbol{\pi}_\theta$ in Algorithm 2 for legibility. The maximum number of episodes N^{E} is pre-set (line 2). The algorithm will end when N^{E} episodes of training have been completed. Each episode performs a random scenario of training.

During the simulation and collecting transitions, the simulation scenario is reset first (line 4, refer to Section 5.1.1). From the first step (line 5), the set of selected flights I^t_{Selected} is generated by the delay priority strategy (line 7). For flight i in I^t_{Selected}, the action value is evaluated by the DQN&LSTM network based on Duelling DQN (line 9) (Wang et al., 2016), and the action with the greatest value is chosen by ε-greedy (line 10) (Dabney et al., 2020). Then, the environment is updated by the joint action $\mathbf{A}_t = \left\{ a_t^i \middle| i \in I^t_{\text{selected}} \right\}$ (line 12).

Next, the reward r_t^i and the observation of the next step $\mathbf{o}_i^{t+1}$ are obtained for each flight i in I^t_{Selected} (lines 14–15). After that, the transition $(\mathbf{o}_t^i, a_t^i, r_t^i, \mathbf{o}_{t+1}^i)$ is stored in the buffer $\wp$, whose capacity is n_{BC} (line 16). The process (lines 7–18) is repeated until all hotspots have been solved.

During the neural network update, transitions of the buffer $\wp$ are sampled by SumTree, one of the critical technologies in Prioritised Replay Buffer (line 21) (Schaul et al., 2015). Then, the loss function l is calculated by Double DQN (Van Hasselt et al., 2016) (line 22). Next, the evaluation network π_θ is updated by gradient descent every step (line 23), while the target network $\pi_{\theta'}$ is updated by copying the parameters of π_θ every period (line 24) (Van Hasselt et al., 2016). Finally, the trained neural network π_θ is obtained.

5. Simulation experiments

This section first describes the experimental settings for training and testing in the experiments, respectively. Next, we present four separate experiments in turn, with the respective objectives of:

1. assessing the training effects and comparing the performance of four alternative delay priority strategies (Section 5.3),
2. demonstrating the generalisation of the proposed method (Section 5.4), and
3. comparing the proposed method against state-of-the-art MARL-based DCB methods (Section 5.5).
4. comparing the proposed method against an optimisation method (Section 5.6).

The experiments were conducted on a computer with i5-8300H CPU. The neural networks were constructed based on Pytorch library in Python3.8 environment.

5.1. Training setup

5.1.1. Training scenarios

In this study, three datasets were used for training (refer to Fig. 7 for the diagram of training scenarios). One training dataset is based on a customised toy case, and the other two training datasets are on real cases. In the toy case, the airspace consists of

Algorithm 2 Multi-agent reinforcement learning for solving DCB problems

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

Initialise the evaluation network $\boldsymbol{\pi}_\theta$ and the target network $\boldsymbol{\pi}_{\theta'}$, and set hyper-parameters (as shown in Table 4)
for episode = $1, 2, ..., N^{\mathrm{E}}$ **do**
  # Simulation & collecting transitions
  Reset the simulation scenario
  $t \leftarrow 1$ # the first step
  repeat
    Generate the selected flight set I^t_{Selected} by the delay priority strategy
    for flight i in I^t_{Selected} **do**
      Evaluate action value by the DQN&LSTM network (based on Duelling DQN):
      $Q\left(\mathbf{o}^i_t, a; \theta\right) = V\left(\mathbf{o}^i_t\right) + A\left(\mathbf{o}^i_t, a\right) - \frac{\sum_{a' \in \mathbf{A}} A\left(\mathbf{o}^i_t, a'\right)}{T_{\mathrm{W}}}, a \in \mathbf{A}$ (Wang et al., 2016)
      Choose the action with the greatest value by ε-greedy:
      $a^i_t = \arg\max_a Q\left(\mathbf{o}^i_t, a; \theta\right)$ (Dabney et al., 2020)
    end for
    Update the environment by the joint action $\mathbf{A}_t = \left\{ a^i_t \middle| i \in I^t_{\mathrm{selected}} \right\}$
    for flight i in I^t_{Selected} **do**
      Calculate the reward r^i_t by Eq. (15)
      Collect the observation of the next step $\mathbf{o}^{t+1}_i$
      Store the transition $(\mathbf{o}^i_t, a^i_t, r^i_t, \mathbf{o}^i_{t+1})$ in the buffer $\wp$ whose capacity is n_{BC}
    end for
    $t \leftarrow t + 1$ # the next step
  until there are no hotspots
  # Neural network update
  Sample n_{BS} transitions by Prioritised Replay Buffer (SumTree) (Schaul et al., 2015) from the buffer $\wp$
  Calculate the loss function by Double DQN:
  $l = \left(r^i_{t+1} + \gamma Q\left(\mathbf{o}^i_t, \arg\max_{a^*} Q\left(\mathbf{o}^i_{t+1}, a^*; \theta \right); \theta' \right) - Q\left(\mathbf{o}^i_t, a^i_t; \theta \right) \right)^2$ (Van Hasselt et al., 2016)
  Update the evaluation network $\boldsymbol{\pi}_\theta$: $\boldsymbol{\pi}_\theta \leftarrow \boldsymbol{\pi}_\theta - \alpha \frac{\partial l}{\partial \boldsymbol{\pi}_\theta}$ # every step (Van Hasselt et al., 2016)
  Update the target network $\boldsymbol{\pi}_{\theta'}$: $\boldsymbol{\pi}_{\theta'} \leftarrow \boldsymbol{\pi}_\theta$ # every n_{TN} steps (Van Hasselt et al., 2016)
end for

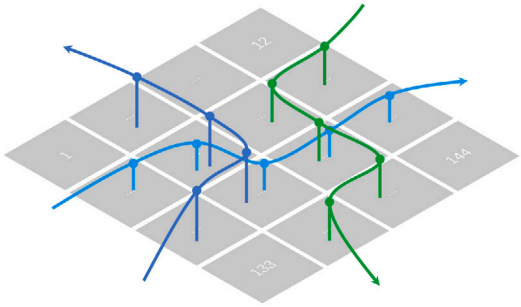
Output: Neural network $\boldsymbol{\pi}_\theta$

144 sectors; flight sequences and sector opening schemes are generated randomly (as shown in Fig. 7(a)). In the two real cases, one is based on the historical data of French airspace, and the other is on the historical data of Spanish airspace (two corresponding examples are shown in Fig. 7(b) and Fig. 7(c), respectively). For the historical data used in training scenarios, the sectors' capacities are their declared values derived from the EUROCONTROL Demand Data Repository v2 (DDR2) database.

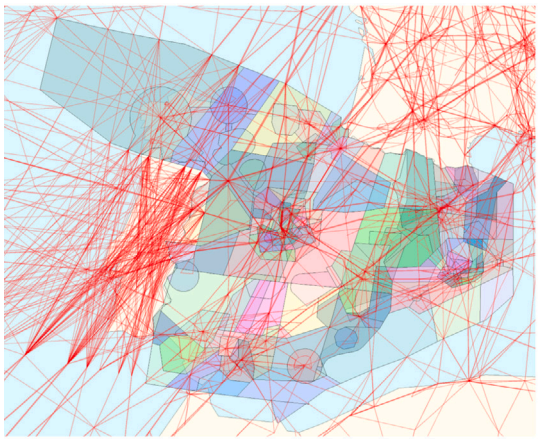
Fig. 8 shows the generation of the three training datasets. In training dataset T, a set of flight sequences and a sector opening scheme generated randomly based on the toy sectors were used to simulate a day's operational data (refer to Eqs. (1), (2) and (3) for the construction of the generated data). Sixty sets of operational data were generated. Thus there were a total of sixty simulated scenarios. In training dataset ES, Spanish sectors were taken as the sectors of interest. Fifty-nine days' historical data for July and August 2019 were used to generate 59 real scenarios. July and August were supposed to have 62 days, but the data of 27/Jul/2019, 27/Aug/2019 and 28/Aug/2019 were selected for test scenarios in the ISOBAR project. Thus, only the other 59 days' operational data were used in this training dataset. In training dataset FR, 59 real scenarios based on French sectors were obtained for training in the same way as in training dataset ES.

The problem sizes of training datasets are shown in Table 2. Note that, in training datasets ES and FR, the number of sectors includes all defined elementary and collapsed sectors. Table 3 shows the uses of training datasets. The check mark (✓) means that the training dataset was used for the corresponding experiment, while the circle mark (○) means not.

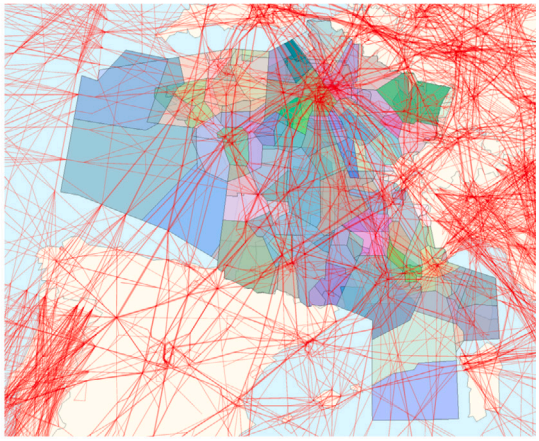
It is worth noting that, although the training dataset is based on real data, the training scenario is constructed essentially using only flight sequences and sector opening schemes (and their capacities), rather than historical delay allocations. The training scenario is then injected into an in-house simulator that mimics the DCB environment. Accordingly, the agent itself makes the delay allocation based on its own observation on the simulation environment, which then receives a feedback from the environment as a result of the action. Thus, no previous DCB experience will be learnt from the historical training dataset.



(a) A toy case



(b) A Spanish scenario



(c) A French scenario

Fig. 7. Schematic of training scenarios.

Table 2
Problem sizes of training datasets.

Training dataset	Scopes of interest	Numbers of sectors of interest	Numbers of flights of interest
T	Toy sectors	144	5000
ES	Spanish sectors	≈150[a]	≈7000[a]
FR	French sectors	≈200[a]	≈11000[a]

[a] As the number of open sectors and flights varies daily for training dataset ES and FR, only approximate numbers are given in the table for reference.

Table 3
Uses of training datasets.

Experiments	T	ES	FR	Sections
Training and delay priority strategy comparison	∘	✓	∘	Section 5.3
Model generalisation	✓	✓	✓	Section 5.4
MARL-based DCB method comparison	∘	✓	✓	Section 5.5
Comparison with an optimisation method	∘	✓	∘	Section 5.6

5.1.2. Hyper-parameter setting

We determine the values of the hyper-parameters in Algorithm 2 through tuning experiments, summarised in Table 4. The Adam optimisation algorithm (Kingma and Ba, 2014) has been used to update the network parameters. Note that the hyper-parameter settings in this paper are based on the experiential values of this study (not necessarily the optimal values) for readers' reference only. The systematic parameter experiment will be the future research work.

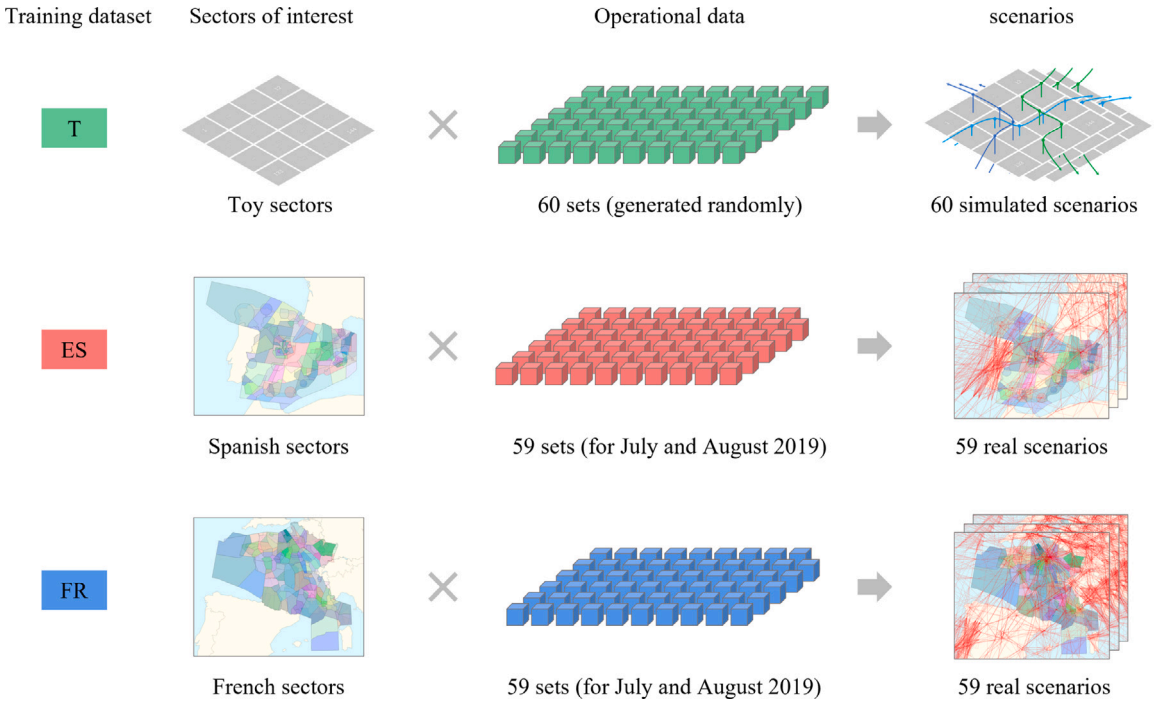


Fig. 8. Datasets for training. A coloured cube refers to a day's operational data, consisting of flight sequences and a sector opening scheme.

Table 4
Hyper-parameter settings for training algorithm.

Hyper-parameters	Values
Greedy rate ε in line 10	0.95
Buffer capacity n_{BC} in line 16	20000
Batch size n_{BS} in line 21	2000
Discount rate γ in line 22	0.9
Learning rate α in line 23	1×10^{-3}
Target network update cycle n_{TN} in line 24	50

5.2. Test setup

5.2.1. Test scenarios

In the ISOBAR project, three dates 27/Jul/2019, 27/Aug/2019 and 28/Aug/2019 were selected as evaluation days (i.e., test scenarios). In the three evaluation day scenarios, the flights' departure/arrival times and airports used are the same as the historical data, but the trajectories used are not necessarily the same as the historical data. The trajectories used were generated by the AU preference AI component, which obtains trajectories preferred by AUs through machine learning methods based on their historical data, i.e., linking weather scenarios to re-routing demands (refer to ISOBAR ConOps for more details). Nevertheless, we have removed the three days' historical data from the training dataset (also mentioned in Section 5.1.1) to prevent the training scenarios from being too similar to the ones for testing purposes.

The parameters of the test scenarios are shown in Table 5. For clarity, the rules of naming test scenarios are as follows:

1. the first two capital letters represent the area of interest (i.e., ES for Spanish and FR for French),
2. the middle four numbers represent the evaluation day in 2019 (e.g., 0727 refers to 27/Jul/2019), and
3. the last letter represents the capacity type (N for nominal capacities; P for predicted capacities, which came from the METEO AI component).

Note that the predicted capacities considered the capacity reduction caused by convective weather. Thus, some predicted capacities are (much) less than the corresponding nominal (declared) capacities on an evaluation day, as a result of the convective weather. The number of sectors of interest refers to the number of sectors through which the flights of interest pass, including elementary and collapsed sectors.

The uses of test scenarios are shown in Table 6. ES_N represents the three test scenarios with nominal capacities based on Spanish sectors, ES_P represents the three test scenarios with predicted capacities based on Spanish sectors, and FR_P represents the three

Table 5
Parameters of test scenarios.

Test scenarios	Evaluation days	Scopes of interest	Capacity types	Numbers of sectors of interest	Numbers of flights of interest
ES_0727_N	27/Jul/2019	Spanish	Nominal	154	7006
ES_0727_P	27/Jul/2019	Spanish	Predicted	154	7006
FR_0727_P	27/Jul/2019	French	Predicted	202	10041
ES_0827_N	27/Aug/2019	Spanish	Nominal	146	6559
ES_0827_P	27/Aug/2019	Spanish	Predicted	146	6559
FR_0827_P	27/Aug/2019	French	Predicted	204	10037
ES_0828_N	28/Aug/2019	Spanish	Nominal	140	6611
ES_0828_P	28/Aug/2019	Spanish	Predicted	140	6611
FR_0828_P	28/Aug/2019	French	Predicted	187	10039

Table 6
Uses of test scenarios.

Experiments	ES_N	ES_P	FR_P	Sections
Training and delay priority strategy comparison	✓	✓	∘	Section 5.3
Model generalisation	∘	∘	✓	Section 5.4
MARL-based DCB method comparison	∘	∘	✓	Section 5.5
Comparison with an optimisation method	∘	∘	✓	Section 5.6

test scenarios with predicted capacities based on French sectors. The check mark (✓) means that the test scenarios were used for the corresponding experiment, while the circle mark (∘) means not.

5.2.2. Benchmark for testing

A standard computer-assisted slot allocation (CASA) was appointed as a benchmark DCB approach in ISOBAR. CASA is a rule-based heuristic method with deterministic solutions, often used in actual operations. It requires less computing power and is simple to use in the real world (Ivanov et al., 2017). CASA is a general term for a class of methods whose algorithms are often fine-tuned to suit specific scenarios. The appointed CASA is introduced in *ATFCM Operations Manual - Network Manager* by EUROCONTROL (Eurocontrol, 2021a), and the source code of a customised CASA based on Python3 is available on our Github repository. In general, three simple key principles have been considered when producing this benchmark CASA solver, namely:

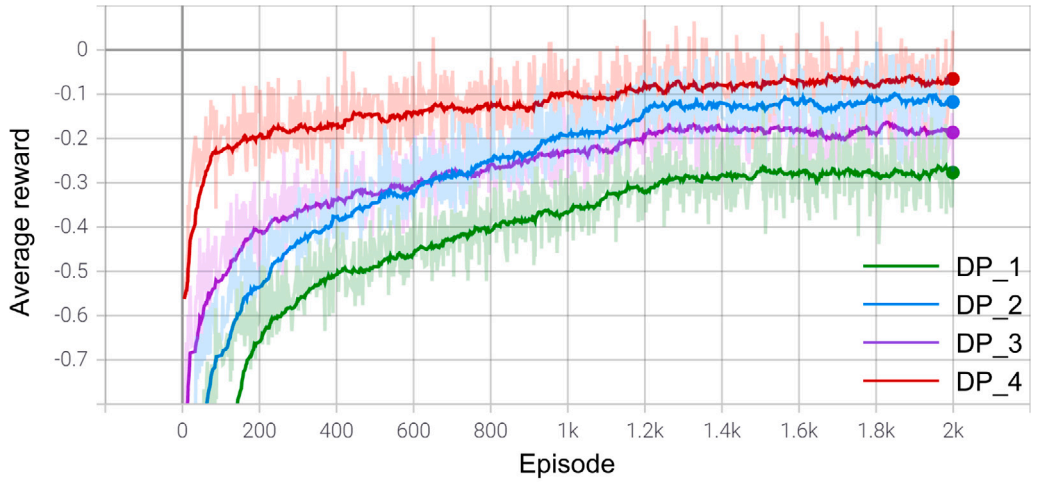
1. Following the rule of First-Come, First-Served, the system extracts all the flights entering the specified sector and sequences them in the order they are planned to arrive without any restriction.
2. At each sector, the time interval between any two flights' entries is guaranteed to be greater than a specific time T_{Interval}. T_{Interval} is equal to the reciprocal of the sector capacity. For example, if a sector's capacity is 25 aircraft entries per 20 min, T_{Interval} is considered as 0.8 min.
3. A flight crossing several sectors may be assigned with different amounts of delay (to ensure the time interval as stated above), the largest of which will be adopted for this flight (i.e., max penalty) in the current iteration.

In order to ensure the fairness of comparison, we set the same parameters and constraints for the CASA algorithm as the proposed reinforcement learning method. This study did not use the existing ISA-CASA algorithm (where ISA means instantaneous self-assessment of workload) in the EUROCONTROL's Network Strategic Tool (NEST) for the following reasons.

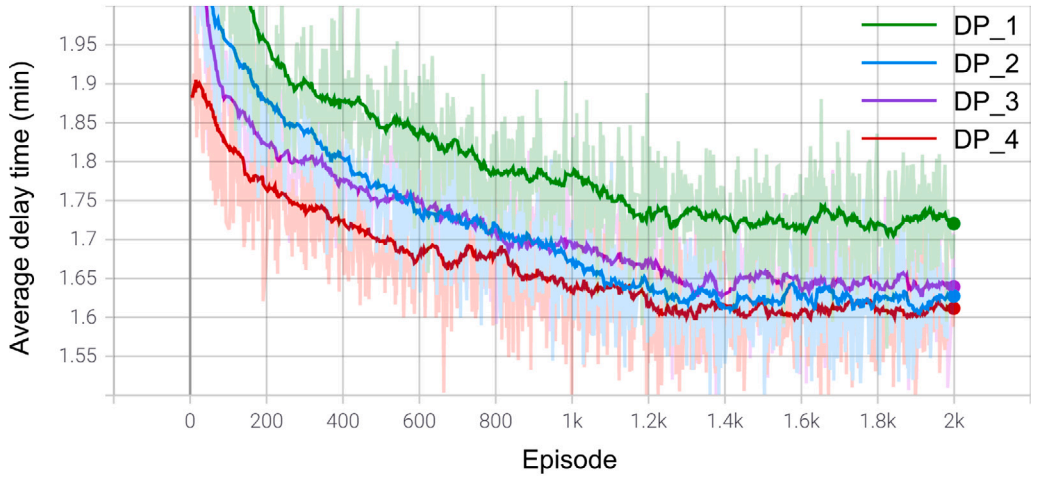
1. Practical aspect: The execution scope of ISA-CASA in NEST is challenging to customise. For example, ISOBAR only considers capacity limits in the sectors of interest (French and Spanish). In contrast, ISA-CASA might consider sector constraints outside the area of interest (for example, on a flight from Heathrow to Charles de Gaulle, ISA-CASA might additionally consider sector limits in the UK).
2. Theoretical aspect: Since the operating rules of ISA-CASA in NEST are not publicly available, it is challenging for us to deal with reason (1) by re-coding ISA-CASA. Without the operating rules, rigorous theoretical analysis is also difficult to carry out.

5.3. Training and delay priority strategy comparison

In this part, we discuss the training results of the proposed method and the comparison of the four delay priority strategies (refer to Section 4.3). Training dataset ES was used for this experiment. In each episode, the environment was reset by a random scenario from training dataset ES (refer to line 4 in Algorithm 2). Five models were trained in this experiment corresponding to the four delay priority strategies and one random strategy, named DP_1, DP_2, DP_3, DP_4 and R, respectively. DP refers to the delay priority strategy. For example, model DP_1 means that it applied *delay priority strategy 1*. R refers to a delay priority strategy in which flight score ς_i^t is replaced by a random value. The random strategy model was trained to verify the effectiveness of delay priority strategies.



(a) Average reward



(b) Average delay time

Fig. 9. Training results. The drawing of the figure is supported by TensorboardX (a Python toolkit for machine learning data visualisation). The light curves are the actual result data, and correspondingly, the dark ones are fitted based on the former by a 95% smoothing coefficient for legibility. Since the figure focuses on showing the process of convergence, the parts for the last 1800 episodes are zoomed in. As such, some values for the first 200 episodes are not completely shown. As the curves of model R are far from others (the performance is much lower than others), they are not shown in this figure (interest readers are directed to Appendix B).

The following average reward was defined to assess the training effectiveness:

$$r_p^{\text{Average}} = \frac{\sum_{t=1}^{t_p^{\text{Last}}} \sum_{i \in \left(I^t_{\text{Selected}}\right)_p} \left(r_i^t\right)_p}{\sum_{t=1}^{t_p^{\text{Last}}} \left(n^t_{\text{Selected}}\right)_p} \tag{24}$$

where r_p^{Average} refers to the average reward of episode p. t_p^{Last} denotes the last step of episode p; I^t_{Selected}, r_i^t and n^t_{Selected} denote the set of selected flights, the reward of flight i and the number of selected flights at step t of episode p, respectively. Therefore, here the average reward r_p^{Average} means the average rewards of all actions in episode p.

Fig. 9(a) shows the average reward of training for the four MARL models with different delay priority strategies. In the first two hundred episodes, the average reward increased rapidly. After about one thousand and six hundred episodes, the four models

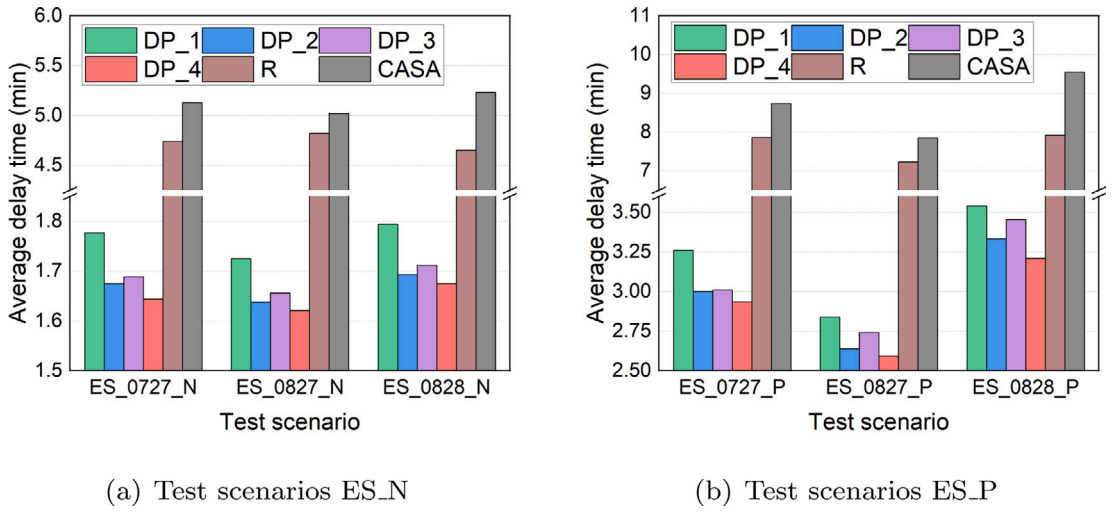


Fig. 10. Delay priority strategy comparison in average delay time.

converged. Therefore, training can be considered effective. In average reward alone, the performance of the four models in order was DP_4 > DP_2 > DP_3 > DP_1. However, a reward in this study reflects the action's merit at the current step but does not accurately reflect the model's performance in solving a DCB problem. Thus, we also present training results in average delay time for the four models (as shown in Fig. 9(b)). The average delay time is equal to the total delay time divided by the number of all flights in a DCB instance. Fig. 9(b) presents similar performance trends to Fig. 9(a); that is, roughly, the greater the reward, the less the delay. Therefore, the performance order mentioned before is reasonable. Specifically, DP_4 was slightly better than DP_2. Although DP_2 was behind DP_3 in the beginning phase, DP_2 overtook DP_3 after about one thousand episodes. DP_1 was significantly weaker than the other three models.

We tested the five trained models (including model R) in test scenarios to further substantiate the above conclusions. Fig. 10(a) and Fig. 10(b) show the average delay time of test scenarios ES_N and ES_P, respectively. The order of performance they present is consistent with that in the training results, which can support the above conclusions about the performance order of the four models. The average delay times for CASA were approximately three times as large as the four delay priority strategy models (i.e., DP_1, DP_2, DP_3 and DP_4). The average delay time for model R was close to that for CASA. As mentioned, the proposed method performs two operations in each iteration, namely 'cherry picking' (through heuristics) and delay allocation (through reinforcement learning). Model R uses random selection instead of 'cherry picking', which make its function incomplete, and its optimality naturally is significantly reduced.

Suppose we compare the two average delay times for the four delay priority strategy models:

1. training value: the fitted value of the training results in the last episode, and
2. test value: the mean of the test results for three test scenarios of ES_N.

Then, compared with CASA, the average delay times of test scenarios ES_N for MARL models were close to those of training results (refer to Fig. 9(b)). More details can be found in Table 7. Generally speaking, test results are usually better than training results for the same scenario since there is uncertainty in the decisions made by agents with greedy rate ε. However, the average delay times of test scenarios ES_N (test values) were greater than training values. That is most likely because the test scenarios were unseen to agents. Nevertheless, the optimisation performance seems to be acceptable if compared with CASA. Thus, we can tentatively infer that the trained models can apply to unseen scenarios without significant degradation in optimisation performance. More discussion on models' generalisation can be found in Section 5.4, where the similarity between the training and test scenarios is lower than in the experiment in Section 5.3. Since the test scenarios ES_P, as more realistic scenarios, considered the reduction of capacity based on real weather data (as mentioned in Section 5.2.1), the average delay times of test scenarios ES_P were greater than those of test scenarios ES_N but were still much less than CASA.

Based on the training results above, we can infer below two features of the delay priority strategy that would have benefited the MARL model:

(a) not only the time window of flight entering but also the next one are assessed, and
(b) the importance of each hotspot is considered inconsistent.

Feature (a) can be considered more significant than feature (b) since DP_2 was finally better than DP_3. Since DP_4 had the best performance, it was used to represent the proposed method and named MARL_DP in the following experiments.

Table 7
Comparison of training values with test values in the average delay time.

Models	Training values (min)	Test values (min)	CASA (min)	Increase (test over training)	Increase (CASA over training)
DP_1	1.725	1.765	5.127	2.34%	197.20%
DP_2	1.624	1.669	5.127	2.75%	215.68%
DP_3	1.642	1.692	5.127	3.07%	212.22%
DP_4	1.611	1.647	5.127	2.21%	218.23%

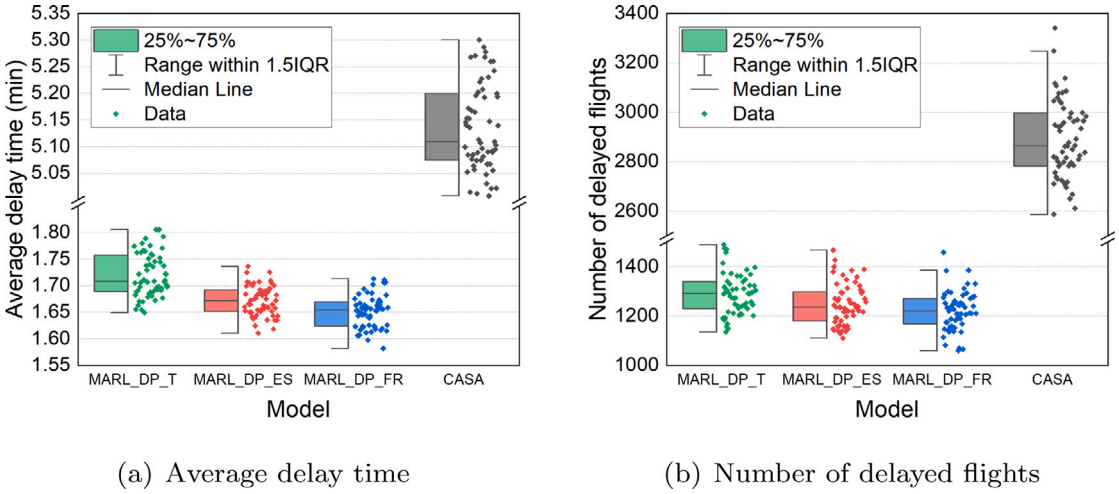


Fig. 11. Model comparison in training dataset FR (French scenarios with nominal capacities), where MARL_DP trained by different training datasets. Models MARL_DP_T (green), MARL_DP_ES (red) and MARL_DP_FR (blue) refer to the model MARL_DP trained with training dataset T, training dataset ES and training dataset FR, respectively. CASA (grey) is used as a benchmark.

5.4. Model generalisation

In this part, we discuss the generalisation of the proposed method. We explored two critical issues through the experimentation:

1. whether the trained model can efficiently solve the DCB problem for unseen scenarios, and
2. how the training data (training scenarios) affect the model's performance.

To this end, we trained three MARL_DP models with training dataset T, training dataset ES and training dataset FR, respectively. The three models are named MARL_DP_T, MARL_DP_ES and MARL_DP_FR, respectively.

To verify the effectiveness of the trained models statistically, we tested them by training dataset FR, which includes 59 French scenarios with nominal capacities. Note that the scenarios of training dataset FR are unseen for MARL_DP_T and MARL_DP_ES but seen for MARL_DP_FR. Therefore, we can analyse the difference in performances of the three methods to explore the two concerns. Figs. 11(a) and 11(b) show the test results in the average delay time and the number of delayed flights for dataset FR, respectively. The average delay time and the number of delayed flights had the same trend. The three MARL_DP models significantly outperformed CASA (in the optimisation of DCB problems, the average delay time and the number of delayed flights are generally considered to be the smaller the value, the better). For the test scenarios based on training dataset FR, the order of optimisation performance of these three MARL_DP models was MARL_DP_FR > MARL_DP_ES > MARL_DP_T. MARL_DP_FR's training scenarios were the same as its test scenarios. If we assume that the training scenarios of MARL_DP_ES are more similar to the training scenarios of MARL_DP_FR (i.e., training dataset FR) than of MARL_DP_T, the optimisation performance is proportional to the similarity between the training and test scenarios. The assumption is reasonable based on the following two facts. First, on the one hand, the training scenarios of MARL_DP_ES are based on the real world, and the flights in the Spanish scenarios have the same parts as the flights in the French scenarios on the same day. Second, on the other hand, the training scenarios of MARL_DP_T are based on toy cases, and the operational data are generated randomly. If we use the median to represent the performance of the corresponding model, the increase rates for other models compared with MARL_DP_FR are shown in Table 8 (refer to the row for training dataset FR). Since the increase rates of MARL_DP_T and MARL_DP_ES were much less than CASA, it could be considered that the optimisation performance is not significantly degraded by the reduced similarity between the training and test scenarios.

To further explore the generalisation of the proposed model, we tested the three MARL_DP models in test scenarios FR_P, which are less similar to the test scenarios and more realistic than the above test scenarios (training dataset FR). Fig. 12(a) and 12(b) show the test results in the average delay time and the number of delayed flights for test scenarios FR_P, respectively. They show the same trend as those in Fig. 11. If we use the mean of the three test scenarios of FR_P to represent the performance of the corresponding

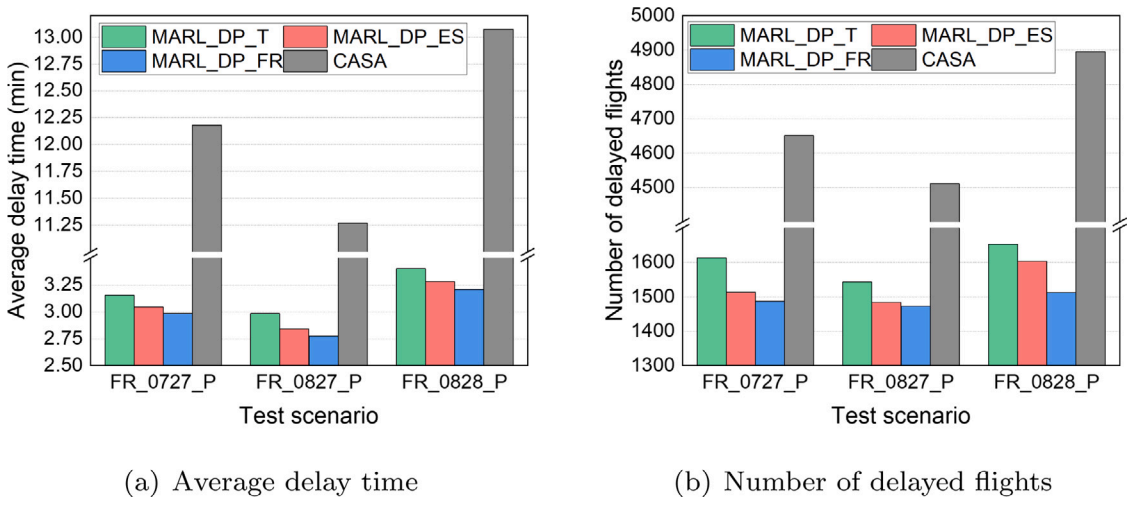


Fig. 12. Model comparison in test scenarios FR_P (French scenarios with predicted capacities), where MARL_DP trained by different training datasets. Models MARL_DP_T (green), MARL_DP_ES (red) and MARL_DP_FR (blue) refer to the model MARL_DP trained with training dataset T, training dataset ES and training dataset FR, respectively. CASA (grey) is used as a benchmark.

Table 8
Increase rates compared with MARL_DP_FR.

Test scenarios	Average delay time			Number of delayed flights		
	MARL_DP_T	MARL_DP_ES	CASA	MARL_DP_T	MARL_DP_ES	CASA
Training dataset FR (59 French scenarios with nominal capacities)	3.25%	1.04%	208.79%	5.90%	1.31%	134.75%
Test scenarios FR_P (3 French scenarios with predicted capacities)	6.37%	2.23%	307.27%	7.51%	2.86%	214.26%

model, the increase rates for other models compared with MARL_DP_FR are shown in Table 8 (refer to the row for FR_P). The test results of FR_P support the conclusions from the tests based on training dataset FR. Besides, we could infer that as sector capacities decrease (predicted capacities were less than nominal capacities in this study), the reduction in the similarity between training and test scenarios may result in a greater loss of optimisation performance (by comparing the rows for training dataset FR and FR_P in Table 8).

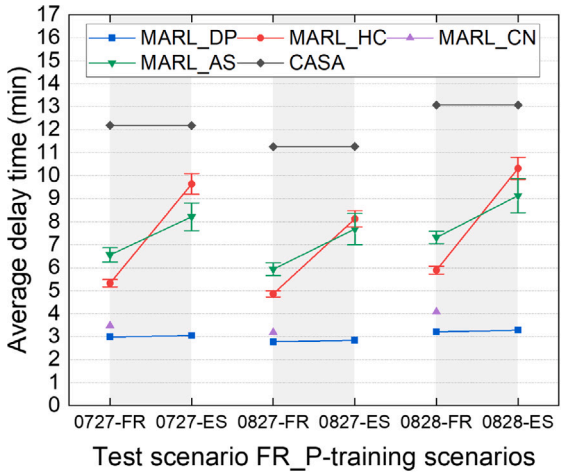
Although the optimisation performance decreases slightly with the reduction in the similarity between training and test scenarios, it is reasonable to consider that the proposed model can solve DCB problems with unseen scenarios after well trained. Therefore, the historical operational data of the region to which the model will be applied are preferred to be the model's training dataset. If the target region's data are unavailable in practice, a model trained by the training data of other regions or even by simulated training data appropriately is acceptable to some degree.

5.5. MARL-based DCB method comparison

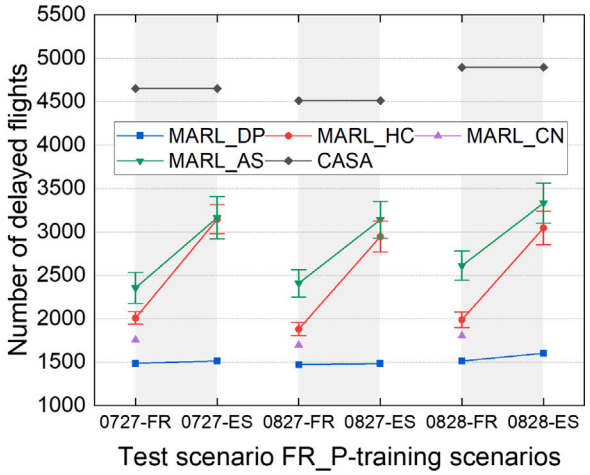
This part compares the proposed method and three state-of-the-art MARL-based DCB methods. There are a total of four MARL methods in this experiment. This experiment aimed to explore the difference between these MARL methods in terms of optimisation performance, computational performance and generalisation. The three state-of-the-art methods were selected because they are all capable of solving unseen problems in specific situations. They are named based on their technical highlights: MARL_HC (Huang and Xu, 2021), MARL_CN (Chen et al., 2023) and MARL_AS (Tang and Xu, 2021), where HC, CN and AS refer to Heuristic Cluster, Customised Network and Action Supervisor, respectively.

Each MARL method was trained by training dataset ES and FR, respectively. Therefore, two trained models were obtained for each MARL method. These two trained models of each MARL method were tested in test scenarios FR_P (French scenarios). As the three state-of-the-art MARL-based DCB methods have shortcomings in function or application, we have improved them in this study to enable the comparative experiments. Some important notes are as follows:

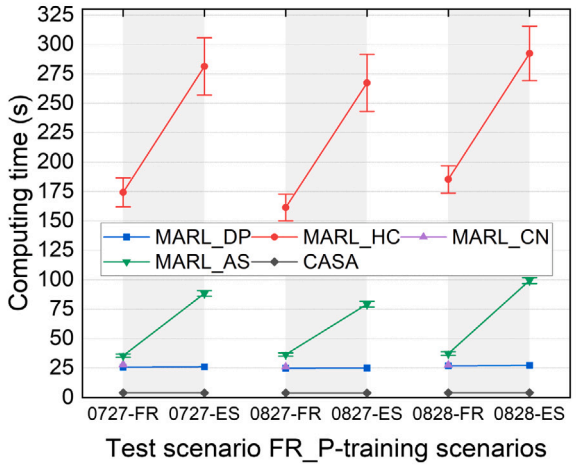
1. Considering that MARL_HC and MARL_AS could not deal with DCB problems with sector opening schemes, we have upgraded these two methods to be compatible with sector opening schemes.



(a) Average delay time



(b) Number of delayed flights



(c) Computing time

Fig. 13. MARL-based DCB method comparison in test scenarios FR_P. In the label of the horizontal axis, the four numbers to the left of the dash refer to the evaluation day of the test scenario FR_P (French scenarios with predicted capacities), and the two capital letters to the right of the dash refer to the models' training dataset (scope of interest of training scenarios). For example, 0727-ES means that the test scenario is the French scenario with predicted capacities on 27/Jul/2019, while the models' training scenarios are Spanish scenarios with nominal capacities. Due to the uncertainty of MARL_HC and MARL_AS, error bars represent the range of results. MARL_CN trained by training dataset ES is not applicable for test scenarios of FR_P.

2. Since MARL_HC takes the sector number as a part of the observation matrix, we numbered all Spanish and French sectors. The Spanish sector numbers were used only for the training, while the French sector numbers were used for both the training and test.
3. The observation matrix of MARL_AS is based on all sectors of interest. Therefore, when we trained MARL_AS by training dataset ES, the observation matrix must be constructed based on all Spanish and French sectors because the model was going to be used to solve the DCB problems with French scenarios.
4. Due to uncertainties in the decision-making in MARL_HC and MARL_AS, they were tested 20 times in each test scenario to obtain results statistically.
5. MARL_CN can only solve the DCB problem for the region where it is trained. Thus, the MARL_CN model trained by training dataset ES cannot be applied for the test scenarios FR_P.

Fig. 13 shows the comparison results between the proposed method and the three state-of-the-art MARL-based DCB methods.

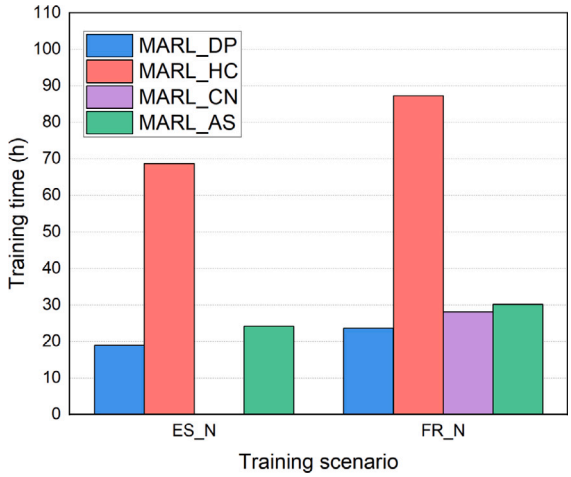


Fig. 14. Training time. Because model MARL_CN trained by training scenario ES_N cannot be used in the comparative experiment, its corresponding training time is not shown.

Fig. 13(a) shows the results in the average delay time. For the four MARL models trained by training dataset FR (where the region is the same as the test scenarios FR_P), MARL_DP outperformed other models. MARL_CN was very close to MARL_DP. MARL_HC and MARL_AS both under-performed although they were much better than CASA. However, for the models trained by training dataset ES (where the region is different from the test scenarios FR_P), the results presented some difference from those trained by training dataset FR. Because of the reduction in the similarity between training and test scenarios, the three MARL models (not including MARL_CN) trained by training dataset ES led to more average delay time than the corresponding models trained by training dataset FR. Specifically, the increases (the mean for the three test scenarios) for MARL_DP, MARL_HC and MARL_AS were 2.23%, 74.61% and 26.33%, respectively. The increase in the average delay time for MARL_DP was much less than MARL_HC and MARL_AS. The increase for MARL_HC was greater than MARL_AS, in the result that the average delay time for MARL_HC was greater than MARL_AS despite the situation for the models trained by training dataset FR being the opposite. MARL_HC identified sectors by their numbers, but sector numbers had no mathematical meaning. That is probably the reason for its significant increase in the average delay time. Although the observation matrix of MARL_AS was constructed based on all Spanish and French sectors when trained by training dataset ES, the part of the neural network related to the French sectors was not effectively trained. Thus, its increase in average delay time would be justifiable. Based on this experiment, there is not enough evidence to show why the increase for MARL_HC would be greater than MARL_AS. We suspect it is most likely because the action supervisors of MARL_AS prevented many unnecessary delays. Besides, MARL_HC and MARL_AS trained by training dataset ES performed with more uncertainty than training dataset FR.

Fig. 13(b) shows the results in the number of delayed flights. The results show a similar trend as the average delay time, but there is a significant difference. The increases in the number of delayed flights (the mean for the three test scenarios) comparing the models trained by training dataset ES with training dataset FR for MARL_DP, MARL_HC and MARL_AS were 2.86%, 55.41% and 30.62%, respectively. The increases for MARL_HC were still greater than MARL_AS, like in average delay time. Differently, the number of delayed flights for MARL_HC was less than MARL_AS regardless of the training dataset.

Fig. 13(c) shows the results in the computing time. In the experimental results of this paper, the parameters of computing time are all clock time. Based on the mean for three test scenarios, CASA computed the fastest within 3.95 s. For MARL methods trained by training dataset FR, MARL_DP (25.74 s) outperformed other methods. MARL_CN (27.09 s) and MARL_AS (36.34 s) were close to MARL_DP. However, MARL_HC (173.64 s) is much slower than other methods, probably because the agent assigns a short delay time (e.g. less than 1 min) to the flight in each iteration, thus requiring more iterations. The increases in computing time (the mean for the three test scenarios) comparing the models trained by training dataset ES with training dataset FR for MARL_DP, MARL_HC, and MARL_AS were 1.02%, 61.45%, and 144.75%, respectively. The increase for MARL_DP was insignificant because MARL_DP could solve an unseen DCB problem well and did not lead to an exceptionally high number of additional iterations. However, the situations for MARL_HC and MARL_AS are the opposite.

Although training time is not our primary performance indicator for MARL-based DCB methods, we still give their training time for readers' reference (as shown in Fig. 14). In this study, the training and solution of reinforcement learning were operated on the CPU without parallel computation (refer to Section 5.6 for the proposed method's parallel computation potential). Each model trained 2000 episodes on the corresponding training dataset in this comparative experiment. The results show that the proposed MARL method is competitive in terms of training time.

Based on the above analysis, we can infer that the proposed method MARL_DP has potential advantages in optimisation performance, computational performance and generalisation compared with the three state-of-the-art MARL-based DCB methods (i.e., MARL_HC, MARL_CN and MARL_AS). The proposed method performed much better than CASA in the optimisation performance. Although the computing time for MARL_DP may be several times as CASA, in the experiments of this study, its computing times were less than half a minute. Additionally, MARL_DP's computing times did not increase significantly due to the reduced similarity between training and test scenarios. Therefore, the computing time of the proposed method is usually acceptable in practice.

Table 9
Nomenclature for the optimisation model.

Symbols	Types	Meanings
x_{id}	Variable	Flight i is allocated delay d (min)
$i \in I$	Set	Flights of interest
$i \in I_j$	Set	Flights that pass through sector j
$j \in J$	Set	Sectors of interest
$d \in D_i$	Set	Delay time (min) that flight i is allowed $D_i = \{0, \ldots, d_i^{\max}\}$ where $d_i^{\max}$ it flight i's maximum delay time allowed
$w \in W_j$	Set	Time windows that sector j opens
d	Parameter	Delay time (min)
C_{jw}	Parameter	Capacity of sector j in time window w
ψ_{id}^{jw}	Parameter	Indicator (refer to Eq. (26))

Table 10
Comparison results in test scenario FR_0727_P with reduced capacity by 10%.

Models	Average delay time (min)	Number of delayed flights	Computing time (min)
MARL_DP_ES	6.32	3421	1.32
CBC	7.04	2841	240

5.6. Comparison with an optimisation method

In this part, we compare the proposed method with an optimisation method to show the difference between the proposed method's optimisation performance and the DCB instance's mathematical boundary for readers' reference. We refer to the state-of-the-art DCB research (Dalmau et al., 2022) to build an integer linear programming (ILP) model based on this study's DCB instance. Eq. (25) formalises the optimisation model of the DCB instance, where the nomenclature refers to Table 9.

$$\min_{x_{id}} \quad \sum_{i \in I} \sum_{d \in D_i} x_{id} d$$

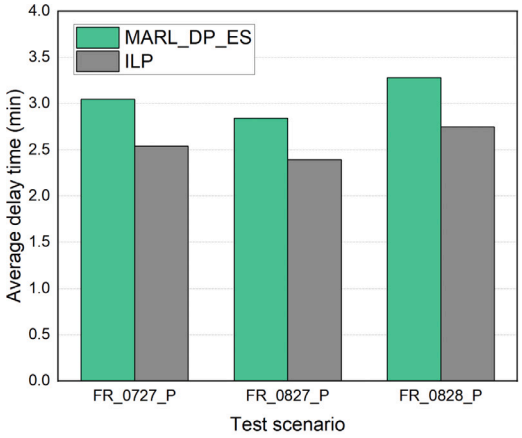
$$\text{s.t.} \begin{cases} \sum_{d \in D_i} x_{id} = 1 & i \in I \\ \sum_{i \in I_j} \sum_{d \in D_i} x_{id} \psi_{id}^{jw} \le C_{jw} & j \in J, w \in W_j \\ x_{id} \in \{0,1\} & i \in I, d \in D_i \end{cases} \tag{25}$$

$$\psi_{id}^{jw} = \begin{cases} 1 & \text{flight } i \text{ enters sector } j \text{ in time window } w \\ & \text{if it is delayed for } d \text{ minutes} \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

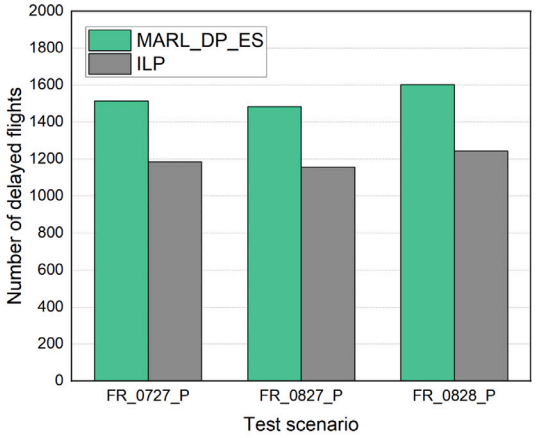
The optimisation model was implemented using PuLP 2.7.0, a Python toolkit for linear programming. To solve the problem, we employed two representative optimisation solvers: CBC 2.10.8 (an open-source solver) and Gurobi 10.0.1 (a commercial solver). The optimisation method's upper limit of computing time is set to 4 h (assuming that the solver runs at 20:00 on the D-1 day). The computing time of optimisation methods consists of model generation time and problem-solving time. The term 'model generation time' refers to the duration required for constructing the optimisation model, encompassing the optimisation objective and constraints. The term 'problem-solving time' encompasses the duration involved in invoking the solver, performing the optimisation calculations, and outputting the solution. In contrast, the computing time of reinforcement learning methods includes only problem-solving time because the neural network has been trained in advance. Model MARL_DP_ES was used to represent the proposed reinforcement learning method. Dataset FR_P was used as the test scenarios.

Fig. 15 shows the comparison results between the proposed reinforcement learning method and the optimisation method. The optimisation method obtained the optimal solution within the computing time limitation in the three test scenarios. In the average delay time, the ILP reduced by 16.6%, 15.8% and 16.3%, respectively, compared with MARL_DP_ES. In the number of delayed flights, the ILP reduced by 21.7%, 22.1% and 22.5%, respectively, compared with MARL_DP_ES. However, in the computing time, MARL_DP_ES is only 0.62%, 0.58% and 0.74% of the CBC, and 0.79%, 0.72% and 0.87% of the Gurobi.

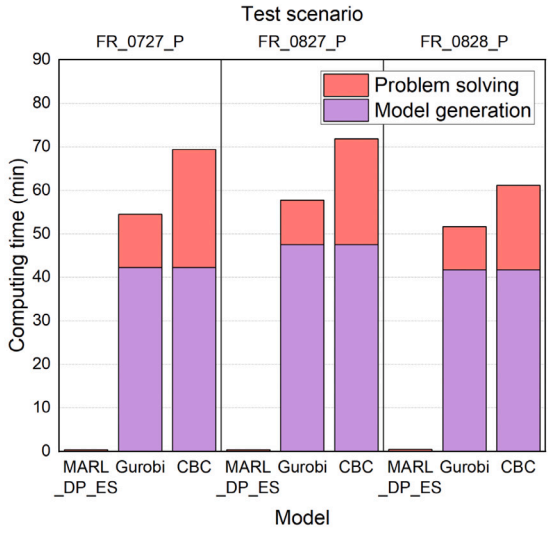
In some cases, particularly when there is high airspace traffic density, the optimisation method may only yield a sub-optimal solution within the designated time frame and this solution may not be as effective as the one obtained through the use of MARL (refer to Table 10 for an example). The mathematical model presented in this section serves as an excellent representation of the flights-to-delay (ATFM slot) allocation problem encountered in the real world and it is commonly utilised in academic literature (Castelli et al., 2011). However, it is worth noting that if the ILP method is employed to solve the DCB problem while considering uncertainty, the size and complexity of the model could potentially increase further (Sandamali et al., 2021; Chen et al., 2017). Hence, the primary challenges for the optimisation method in addressing the actual DCB problem lie in the instability of computing time and the difficulties associated with handling uncertainty. In contrast, the proposed MARL method offers the advantage of reducing computing time through parallel computation (e.g., lines 4–6 in Algorithm 1, and lines 8–11 and 13–17 in



(a) Average delay time



(b) Number of delayed flights



(c) Computing time

Fig. 15. Comparison with the optimisation method in test scenarios FR_P.

Algorithm 2) and quickly iterating to respond to uncertainties in operations (refer to Section 6.1 for more detailed information). Furthermore, although there are some outstanding heuristic algorithm-based DCB methods that can strike a balance between optimality and computing time (Khassiba and Delahaye, 2022), the proposed RL method holds the potential to enhance the solution's optimality compared to conventional heuristic methods. This is because the RL approach has the capability to offload a substantial computational burden from the online phase to the offline learning phase.

6. Discussion

6.1. Uncertainty

As this paper focuses on verifying the method's feasibility, uncertainty was not considered in the proposed model. However, from the point of view of practical application, the uncertainty issue may need to be taken carefully. In real-world DCB problems, there are probably two main types of uncertainty.

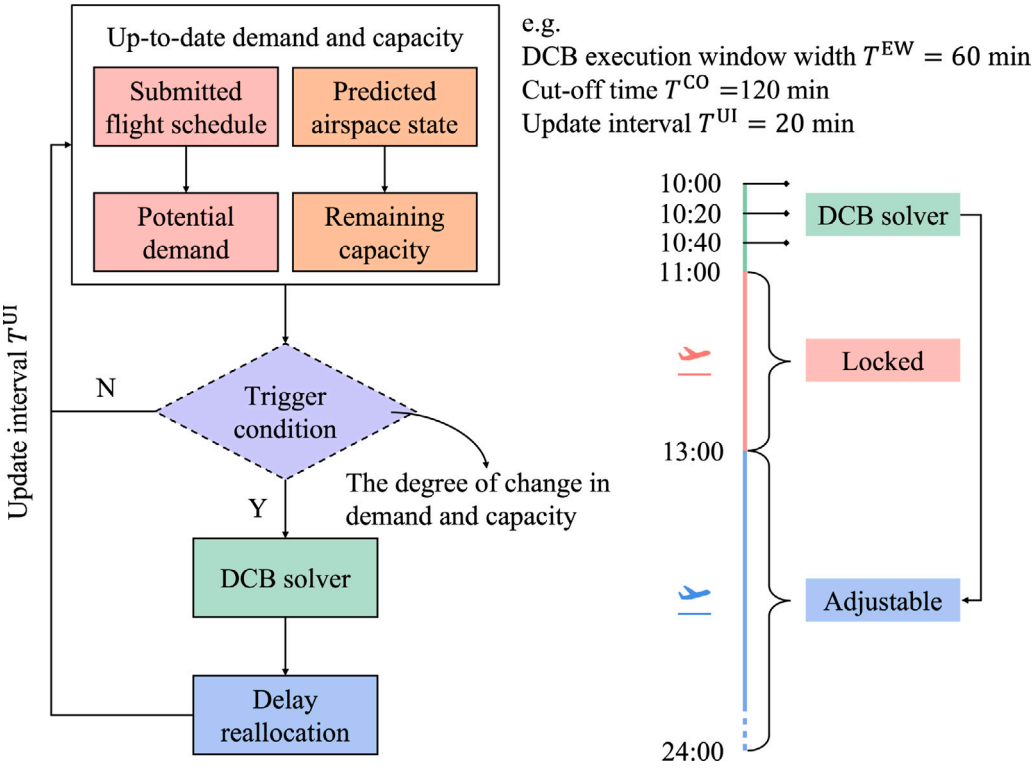


Fig. 16. Potential operational framework that can deal with uncertainty on the operation day (D0 day).

1. Uncertainty of plans: Getting all flight plans on the D-1 day is challenging (Eurocontrol, 2021b). Information on non-scheduled (e.g. business aviation and charter) flights are not easily available and known, and airspace users rarely file their flight plans that early (thus revealing their flight intentions).
2. Uncertainty of operation: Some flights may not be able to depart on time as allocated (later or even earlier) due to non-ATFM factors (e.g., airline factors) (Eurocontrol, 2021b).

Fortunately, the proposed method holds promising potential for addressing the two uncertainties. To enhance readers' comprehension of the proposed method, we present a potential operational framework in Fig. 16 that effectively handles uncertainty on the D0 day.

On the D-1 day, the proposed DCB solver is executed based on the submitted flight schedules and the predicted airspace state. On the D0 day, the DCB solver is periodically executed with an event-based trigger condition. Specifically, we assume that the update interval is denoted as T^{UI}. Every T^{UI}, the potential demand and remaining capacity information of the airspace of interest are updated based on the most recent submitted flight schedules and predicted airspace state, respectively. Then, the program determines whether the difference between the current demand and capacity and that of the last DCB solver execution reaches the set threshold. For example, this threshold can be defined as the number of additionally submitted flight schedules reaching 200 or a local reduction of 5% in predicted sector capacity. If the threshold is reached, the DCB solver will be executed to reallocate delay, where the allocated delay is updated instead of accumulated. The program then proceeds to the next round. If the trigger condition is not met, the program will directly proceed to the next round. It should be noted that the purpose of setting trigger conditions is to avoid frequent adjustment of delay allocations when there are no significant changes in the airspace environment. In fact, the trigger condition is not necessary for the proposed framework and its inclusion can be based on practical requirements.

We set the DCB execution window width T^{EW} and cut-off time T^{CO} to provide a clearer understanding of the execution method employed by the DCB solver. In the same DCB execution window, the DCB solver processes flights that fall within the same departure time range. It is important to note that flights with a current departure time earlier than the specified cut-off time are not included in the DCB solver's execution. To illustrate, if we assume a DCB execution window width T^{EW} of 60 min, a cut-off time of T^{CO} 120 min and a update interval T^{UI} of 20 min, AUs must submit their flight schedules at least 200 min prior to the scheduled departure time (i.e., $T^{EW} + T^{CO}+T^{UI}$). For example, during the period of 10:00–11:00 (a DCB execution window), the DCB solver can be executed at 10:00, 10:20, and 10:40 (if the trigger condition is met), with the solved flight's departure time falling between 13:00 and 24:00.

However, flights with a departure time between 11:00 and 13:00 are not considered within the solution range of the DCB solver. The DCB execution window moves forward in increments of T^{EW} until the end of the operational day (D0).

In addition to the operational framework mentioned above, we can also enhance the reinforcement learning model. For example, we can expand the time range of the agent's observations to enhance its foresight and add a robustness index to the reward function to reduce the possibility of a hotspot surge due to the uncertainty of operation.

6.2. Optimality

In this study, we have compared the proposed method with a real-word method (CASA), state-of-the-art MARL methods (MARL_HC, MARL_CN and MARL_AS) and an optimisation method (ILP) through the test scenarios FR_P. The proposed method is more competitive than the real-world method and the three state-of-the-art MARL methods in terms of optimality. Compared with the mathematical boundary of the DCB instances in this study, the proposed MARL method could still be further optimally enhanced. In some cases, the optimisation method is challenging to obtain the optimal solution within a limited time or the computing time is too long. Therefore, it is not fully competent for real-world DCB problems (e.g., dealing with uncertainty through the proposed operational framework in Fig. 16). In the real world, in addition to using the proposed MARL method alone, parallel computation with two solvers (MARL method and optimisation method) is also a possible way. Within the preset computing time limit: (1) if the optimisation method can obtain the optimal solution, the optimal solution shall be used; (2) otherwise, the solution with better optimality between the optimisation method (i.e., its existing sub-optimal solution) and the MARL method shall be used. Moreover, the solution of the MARL method can also be used as the initial solution of the optimisation method to shorten its computing time.

7. Conclusions

This paper presented a general multi-agent reinforcement learning method that integrates a heuristic-based delay priority strategy for demand and capacity balancing. The proposed model was trained and tested based on real Spanish and French air traffic data (in addition to a toy case for comparison). The results of the various experiments were analysed and discussed in detail. The key conclusions are briefly summarised as follows:

1. The proposed method is fully scalable. Even models trained on toy cases can solve arbitrary large-scale DCB instances, not necessarily limited by training scenarios and problem dimensions.
2. Despite above, as the similarity between the training scenario and the test (application) scenario increases, the optimisation performance of the model can be also improved.
3. The delay priority strategy with the following two features is most suitable for the proposed MARL method: it (a) assesses not only the time window of flight entering but also the next one, and (b) considers the importance of each hotspot as inconsistency.
4. The proposed method's computing time is only around half a minute (although this may increase depending on the problem scale), which is typically acceptable for practical application.
5. The proposed method has potential advantages in optimisation performance, computational performance and generalisation compared with the three state-of-the-art MARL-based DCB methods.

Nevertheless, we acknowledge that the situation in practice could be more complicated than we have considered in this study. In future work, we will further explore the advantages of RL-based DCB methods and refine their functionality. For example, we will systematically test the performance of the proposed method in uncertain scenarios. We hope to explore adding a rerouting strategy into the reinforcement learning framework to form a DCB optimisation method integrating rerouting and delay allocation. The delay priority strategy will be enhanced to improve performance further. Moreover, although the delay priority strategy reduces the potential learning difficulty for agents, it also narrows the solution space of the DCB problem. Therefore, the mechanism of the impact of heuristic algorithms on reinforcement learning deserves further exploration to obtain a balance between learning efficiency and solution space. Systematic parameter studies should be conducted to reduce the optimality gaps. Moreover, improving the fairness and transparency of the RL-based DCB method (e.g., by incorporating fairness reward and/or explainable artificial intelligence) can enhance its practical acceptance.

CRediT authorship contribution statement

Yutong Chen: Methodology, Validation, Writing – original draft. **Yan Xu:** Methodology, Supervision, Writing – review & editing. **Minghua Hu:** Supervision, Writing – review & editing.

Acknowledgements

This work was funded by the SESAR Joint Undertaking under grant agreement No. 891965, as part of the European Unions Horizon 2020 research and innovation programme: ISOBAR (artificial Intelligence to forecaSt meteO-Based DCB imbAlances for netwoRk operations planning). The opinions expressed herein reflect the authors' view only. Under no circumstances shall the SESAR Joint Undertaking be responsible for any use that may be made of the information contained herein.

Table A.11
Nomenclature for the proposed MARL method.

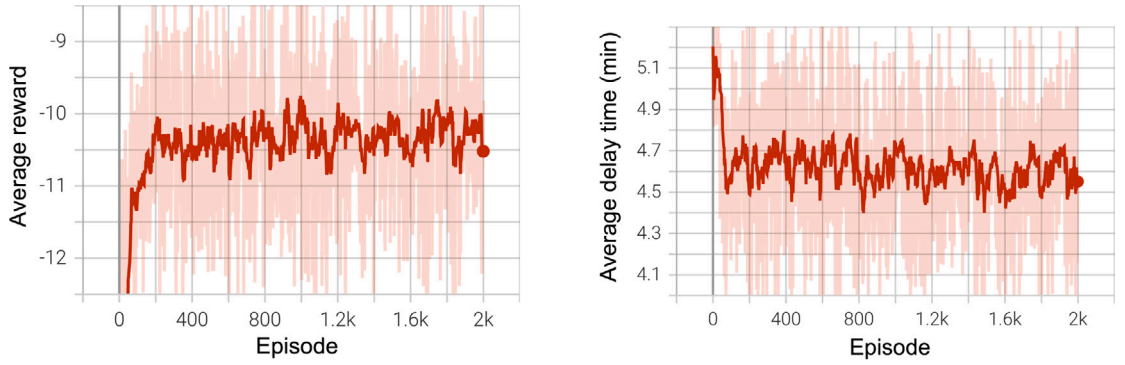
Symbols	Definitions
$\mathbb{F}$	Traffic demand
$\mathbb{F}_{\text{Initial}}$	Initial traffic demand
$\mathbb{S}$	Sector opening scheme
I	Set of flights
I^t_{Selected}	Set of selected flights at step t
I^t_{jw}	Set of flightswhose entry time of sector j is in time window w at step t
J	Set of sectors
J_i	Set of sectors through which flight i is scheduled to pass
W	Set of time windows
Π_Θ	Set of policies, where Θ is the parameter
$\mathbf{A}$	Action space
$\mathbf{A}_t$	Joint action at step t
$\boldsymbol{\Phi}^t_{ij}$	Partial observation (inputted state for the agent)of sector j about flight i at step t
f_i	Flight sequence of the ith flight
e_{ij}	j-th sector through which the ith flight is scheduled to pass
t_{ij}	Time when the ith flight enters into the j-th sector
w^t_{ij}	Time window when flight i will enter sector $j$if it departs as scheduled at step t
D_{jw}	Demand of the jth sector in the wth time window
$d^t_{jw^t_{ij}}$	Sector j's demand in time window w^t_{ij} at step t
C_{jw}	Capacity of the jth sector in the wth time window
$c^t_{jw^t_{ij}}$	Sector j's capacity in time window w^t_{ij} at step t
Δt_i	Delay time allocated for the ith flight
h^i_t	Delay time of flight i at step t
t^{Last}_p	The last step of episode p
$p^t_{jw^t_{ij}}$	Entry timestamp
$u^t_{jw^t_{ij}}$	Number of overloads of sector j in time window w^t_{ij}
ς^t_i	Score of flight i at step t
$\left(\varsigma^t_i\right)_*$	The scores of flight i at step t by *Priority assessment model* *
$k^t_{jw^t_{ij}}$	Indicator: whether time window w^t_{ij} of sector $j$is a hotspot at step t
a^i_t	Action of the ith agent at the tth step
$\mathbf{s}^i_t$	Global state of the ith agent at the tth step
$\mathbf{o}^i_t$	Partial observation of the ith agent at the tth step
r^i_t	Agent i's reward at step t
$\left(r^{\text{Overload}}\right)^i_t$	Agent i's reward of overload at step t
$\left(r^{\text{Delay}}\right)^i_t$	Agent i's reward of delay at step t
r^{Average}_p	Average reward of episode p
$\boldsymbol{\pi}^i_{\theta_i}$	Policy of the ith agent with policy parameter θ_i
θ_i	Current policy parameter of the ith agent
M_i	Number of sectorsthrough which the ith flight is scheduled to pass
N	Number of flights
N^{E}	Maximum number of episodes
n^t_{Selected}	Number of selected flights at step t of episode p
T_{W}	Width of a time window
n_{BC}	Capacity of buffer
n_{BS}	Number of transitions sampled from buffer
n_{TN}	Target network update cycle
l	Loss function
$\wp$	Buffer for storing transitions
ε	Greedy rate
γ	Discount rate
α	Learning rate

Appendix A. Nomenclature

See Table A.11

Appendix B. Training curves for model r

See Fig. B.17



(a) Average reward (b) Average delay time

Fig. B.17. Training curves for model R (random strategy model).

References

Agogino, A.K., Tumer, K., 2012. A multiagent approach to managing air traffic flow. Auton. Agents Multi-Agent Syst. 24 (1), 1–25.

Agustín, A., Alonso-Ayuso, A., Escudero, L.F., Pizarro, C., et al., 2010. Mathematical optimization models for air traffic flow management: A review. URL https://burjcdigital.urjc.es/bitstream/handle/10115/3405/ATFM_SOTA.pdf?sequence=1&isAllowed=y.

Castelli, L., Pesenti, R., Ranieri, A., 2011. The design of a market mechanism to allocate air traffic flow management slots. Transp. Res. C 19 (5), 931–943.

Chen, J., Chen, L., Sun, D., 2017. Air traffic flow management under uncertainty using chance-constrained optimization. Transp. Res. B 102, 124–141.

Chen, Y., Hu, M., Xu, Y., Yang, L., 2023. Locally generalised multi-agent reinforcement learning for demand and capacity balancing with customised neural networks. Chin. J. Aeronaut. 36 (4), 338–353.

Chen, Y., Xu, Y., Hu, M., Yang, L., 2021. Demand and capacity balancing technology based on multi-agent reinforcement learning. In: 2021 IEEE/AIAA 40th Digital Avionics Systems Conference. DASC, IEEE, pp. 1–9.

Crespo, A.M.F., Weigang, L., de Barros, A.G., 2012. Reinforcement learning agents to tactical air traffic flow management. Int. J. Aviat. Manag. 1 (3), 145–161.

Dabney, W., Ostrovski, G., Barreto, A., 2020. Temporally-extended *epsilon*-greedy exploration. arXiv preprint arXiv:2006.01782.

Dalmau, R., Gawinowski, G., Anoraud, C., 2022. Comparison of various temporal air traffic flow management models in critical scenarios. J. Air Transp. Manag. 105, 102284.

Diao, X., Chen, C.-H., 2018. A sequence model for air traffic flow management rerouting problem. Transp. Res. E Logist. Transp. Rev. 110, 15–30.

Duong, T., Todi, K.K., Chaudhary, U., Truong, H.-L., 2019. Decentralizing air traffic flow management with blockchain-based reinforcement learning. In: 2019 IEEE 17th International Conference on Industrial Informatics, vol. 1. INDIN, IEEE, pp. 1795–1800.

Eurocontrol, 2020. Technical note on en route capacity - A perspective on all aspects of en route capacity. Tech. rep., EUROCONTROL, URL https://ansperformance.eu/library/tn_prc_2020_12_enroute_capacity.pdf.

Eurocontrol, 2021a. ATFCM Operations Manual - Network Manager. Tech. rep., EUROCONTROL, URL https://www.eurocontrol.int/sites/default/files/2021-01/eurocontrol-atfcm-operations-manual-24-1-18012021.pdf.

Eurocontrol, 2021b. All-causes delay and cancellations to Air Transport in Europe for 2020. Tech. rep., EUROCONTROL, URL https://www.eurocontrol.int/publication/all-causes-delay-and-cancellations-air-transport-europe-2020.

Eurocontrol, 2022. Performance review report 2021 - An assessment of air traffic management in Europe. Tech. rep., EUROCONTROL, URL https://www.eurocontrol.int/sites/default/files/2022-06/eurocontrol-prr-2021_0.pdf.

Guo, Y., Hu, M., Zou, B., Hansen, M., Zhang, Y., Xie, H., 2022. Air traffic flow management integrating separation management and ground holding: An efficiency-equity Bi-objective perspective. Transp. Res. B 155, 394–423.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Huang, C., Xu, Y., 2021. Integrated frameworks of unsupervised, supervised and reinforcement learning for solving air traffic flow management problem. In: 2021 IEEE/AIAA 40th Digital Avionics Systems Conference. DASC, IEEE, pp. 1–10.

Huang, J., Xu, Q., Yan, Y., Ding, H., Tian, J., 2019. Generalized method of modeling minute-in-trail strategy for air traffic flow management. Math. Probl. Eng. 2019.

ICAO, 2008. Manual on Air Traffic Management System Requirements (Doc 9882). Tech. rep., ICAO, URL https://www.icao.int/airnavigation/IMP/Documents/Doc%209882%20-%20Manual%20on%20ATM%20Requirements.pdf#search=doc%209882.

ICAO, 2016. Procedures for Air Navigation Services (PANS) - Air Traffic Management (Doc 4444). Tech. rep., ICAO.

Ivanov, N., Netjasov, F., Jovanović, R., Starita, S., Strauss, A., 2017. Air traffic flow management slot allocation to minimize propagated delay and improve airport slot adherence. Transp. Res. A Policy Pract. 95, 183–197.

Khassiba, A., Delahaye, D., 2022. Simulated-annealing hyper-heuristic for demand-capacity balancing in air traffic flow management. In: SESAR Innovation Days. pp. 1–9.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kistan, T., Gardi, A., Sabatini, R., Ramasamy, S., Batuwangala, E., 2017. An evolutionary outlook of air traffic flow management techniques. Prog. Aerosp. Sci. 88, 15–42.

Kravaris, T., Spatharis, C., Blekas, K., Vouros, G.A., Garcia, J.M.C., 2018. Multiagent reinforcement learning methods for resolving demand-capacity imbalances. In: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference. DASC, IEEE, pp. 1–10.

Kravaris, T., Vouros, G.A., Spatharis, C., Blekas, K., Chalkiadakis, G., Garcia, J.M.C., 2017. Learning policies for resolving demand-capacity imbalances during pre-tactical air traffic management. In: German Conference on Multiagent System Technologies. Springer, pp. 238–255.

Liu, Y., Liu, Y., Hansen, M., Pozdnukhov, A., Zhang, D., 2019. Using machine learning to analyze air traffic management actions: Ground delay program case study. Transp. Res. E Logist. Transp. Rev. 131, 80–95.

Mannino, C., Nakkerud, A., Sartor, G., 2021. Air traffic flow management with layered workload constraints. Comput. Oper. Res. 127, 105159.

Montlaur, A., Delgado, L., 2020. Flight and passenger efficiency-fairness trade-off for ATFM delay assignment. J. Air Transp. Manag. 83, 101758.
Murca, M.C.R., 2018. Collaborative air traffic flow management: Incorporating airline preferences in rerouting decisions. J. Air Transp. Manag. 71, 97–107.
Sandamali, G.G.N., Su, R., Sudheera, K.L.K., Zhang, Y., 2021. A safety-aware real-time air traffic flow management model under demand and capacity uncertainties. IEEE Trans. Intell. Transp. Syst..
Schaul, T., Quan, J., Antonoglou, I., Silver, D., 2015. Prioritized experience replay. arXiv preprint arXiv:1511.05952.
SESAR, 2021. ISOBAR operational framework. URL https://isobar-project.eu/wp-content/uploads/2021/06/ISOBAR-OSED-D1.1-v00.01.01_approved.pdf.
Shone, R., Glazebrook, K., Zografos, K.G., 2021. Applications of stochastic modeling in air traffic management: Methods, challenges and opportunities for solving air traffic problems under uncertainty. European J. Oper. Res. 292 (1), 1–26.
Spatharis, C., Bastas, A., Kravaris, T., Blekas, K., Vouros, G.A., Cordero, J.M., 2021. Hierarchical multiagent reinforcement learning schemes for air traffic management. Neural Comput. Appl. 1–13.
Spatharis, C., Blekas, K., Bastas, A., Kravaris, T., Vouros, G.A., 2019. Collaborative multiagent reinforcement learning schemes for air traffic management. In: 2019 10th International Conference on Information, Intelligence, Systems and Applications. IISA, IEEE, pp. 1–8.
Spatharis, C., Kravaris, T., Vouros, G.A., Blekas, K., Chalkiadakis, G., Garcia, J.M.C., Fernandez, E.C., 2018. Multiagent reinforcement learning methods to resolve demand capacity balance problems. In: Proceedings of the 10th Hellenic Conference on Artificial Intelligence. pp. 1–9.
Tang, Y., Xu, Y., 2021. Multi-agent deep reinforcement learning for solving large-scale air traffic flow management problem: A time-step sequential decision approach. In: 2021 IEEE/AIAA 40th Digital Avionics Systems Conference. DASC, IEEE, pp. 1–10.
Torres, S., 2012. Swarm theory applied to air traffic flow management. Procedia Comput. Sci. 12, 463–470.
Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30. pp. 1–7.
Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N., 2016. Dueling network architectures for deep reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 1995–2003.
Xiao, M., Cai, K., Abbass, H.A., 2018. Hybridized encoding for evolutionary multi-objective optimization of air traffic network flow: A case study on China. Transp. Res. E Logist. Transp. Rev. 115, 35–55.
Xu, Y., Dalmau, R., Melgosa, M., Montlaur, A., Prats, X., 2020a. 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 134, 229–255.
Xu, Y., Prats, X., Delahaye, D., 2020b. Synchronised demand-capacity balancing in collaborative air traffic flow management. Transp. Res. C 114, 359–376.
Zhang, Y., Xu, D., Guoye, C., Wen, T., Yinge, A., 2019. Research on performance analysis of air traffic flow management combined strategy impacted by uncertainty. In: 4th International Conference on Humanities Science, Management and Education Technology (HSMET 2019). Atlantis Press, pp. 238–244.



https://dspace.org/

General multi-agent reinforcement learning integrating heuristic-based delay priority strategy for demand and capacity balancing

Chen, Yutong; Xu, Yan; Hu, Minghua

2023-08-01
Elsevier
https://doi.org/10.1016/j.trc.2023.104218
Chen Y, Xu Y, Hu M. (2023) General multi-agent reinforcement learning integrating heuristic-based delay priority strategy for demand and capacity balancing. Transportation Research Part C: Emerging Technologies, Volume 153, August 2023, Article number 104218

Downloaded from a DSpace repository. https://dspace.org/