Proceedings of the 2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC), 3-7 October, San Antonio, TX, USA
DOI: 10.1109/DASC52595.2021.9594397

Integrated Frameworks of Unsupervised, Supervised and Reinforcement Learning for Solving Air Traffic Flow Management Problem

1st Cheng Huang
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedford, UK
cheng-huang.huang@cranfield.ac.uk

2nd Yan Xu
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedford, UK
yanxu@cranfield.ac.uk

***Abstract*—This paper studies the demand-capacity balancing (DCB) problem in air traffic flow management (ATFM) with collaborative multi-agent reinforcement learning (MARL). To attempt the proper ground delay for resolving airspace hotspots, a multi-agent asynchronous advantage actor-critic (MAA3C) framework is firstly constructed with the long short-term memory network (LSTM) for the observations, in which the number of agents varies across training steps. The unsupervised learning and supervised learning are then introduced for better collaboration and learning among the agents. Experimental results demonstrate the scalability and generalization of the proposed frameworks, by means of applying the trained models to resolve different simulated and real-world DCB scenarios, with various flights number, sectors number and capacity settings.**

***Index Terms*—DCB; Multi-agent; Reinforcement Learning; Unsupervised Learning; Supervised Learning**

I. Introduction

The limited airspace capacity and the fast growing demand of flights nowadays have often led to airspace congestions. It is therefore needed to offload the airspace during the pre-flight phase, such that the downstream air traffic controllers (ACTOs) can be well protected when providing in-flight separation assurance. Air traffic flow management (ATFM) is one of these activities. In ATFM, a hotpot occurs when the predicted demand of airspace volume exceeds its capacity, which leads to the so-called demand-capacity balancing (DCB) problem to eliminate those hotspots.

One common resolution for the problem is to regulate the demand (or traffic flow). It can redistribute the flow temporally, such as ground delay and airborne delay, and can also redirect the flow spatially such as re-routing [1]. Another resolution is to reduce the traffic complexity and thus increase the airspace capacity [2] [3]. Conventionally, the DCB problem has been mathematically formulated with optimisation models, such as 0-1 Integer Programming [1], Eulerian-Lagrangian model [4], with ground holding, airborne holding and re-routing strategies involved.

With the successful application in many domains, machine learning methods have been also explored in the ATFM field. Specifically, multi-agent reinforcement learning (MARL) methods were adopted to tackle the DCB problem as the intelligent agent could learn a proper solution, through numerous trials and errors, without formulating complex hand-crafted models. The demand-capacity imbalance could be constructed as the interacted networks of flight trajectories, in which agents with interactions were defined as "peers" and the connection of "peers" neighbourhood promoted the information propagation. Independent reinforcement learning, edge-based multi-agent reinforcement learning and agent-based multi-agent learning were proposed according to the features of agents' coordination graph [5]. The hierarchical reinforcement learning frameworks were proposed based on the state-action abstraction and temporal action abstraction by taking advantage of the coordination of agents to handle real-world problems. Regardless of the agents' actions being independent or connected, the hierarchical frameworks could produce promising results [6].

There have been also similar congestion problems in other fields resolved by MARL methods. The fleet management for online ride-sharing reallocated the available vehicles to alleviate the huge amount of orders in the specific areas. The environment was usually divided into several grid areas and large-scale agents were involved. The contextual MARL utilized the geographic context and collaborative context to tackle the complex stochastic demand-supply issue [7]. The mean field MARL took advantage of mean field approximation to resolve the variable population size in the ride-sharing order dispatching [8]. MARL was also integrated with graph neural networks to solve the changing demand and supply issue [9].

In this paper, it is hypothesized that MARL can come

The work was conducted under the project ISOBAR (artificial Intelligence to forecaSt meteO-Based DCB imbAlances for netwoRk operations planning), which is supported by SESAR, under the EU H2020 programme. This research was also partially supported by grants from the Funds of China Scholarship Council (202008420248). We thankfully acknowledge the effort of EUROCONTROL for developing the NEST software and the access of using the huge amount of historical data.

into play to train a general DCB solver, where some of the computational complexity can be addressed offline, i.e., in the learning phase. Thus, the objective is, on one hand, providing solutions of high quality for different DCB scenarios, and on the other hand, enabling fast-time computations to meet the operational requirements. The MARL environment is constructed by a number of airspace sectors and flights with spatio-temporal information. Flights in hotspots are defined as the candidate agents to interact with the environment. The agents' number reaches hundreds even thousands, if compared with typical multi-agent systems [10], therefore increasing the difficulty of multi-agent learning process.

The contributions of the paper are summarised as follows:

- A multi-agent asynchronous advantage actor-critic (MAA3C) framework is proposed, with unsupervised learning and supervised learning further integrated.
- Only flights in hotspots are regarded as the candidate agents, and as a consequence the proposed frameworks can well deal with the varying number of agents.
- The scalability and generalization are assessed by evaluating the models trained on a simulated DCB scenario with real scenarios of different magnitudes.

The rest of the paper is organized as follows: Section II introduces the fundamental mathematics of the DCB problem and the reinforcement learning. Section III describes the proposed MAA3C frameworks, unsupervised and supervised learning methods. Section IV demonstrates and analyses the results on simulated and real cases. Section V gives the conclusion of the paper.

II. Problem Statement

In this section, the basic DCB problem and a general understanding of its modelling approach are elaborated via an illustrative example. The problem is then formulated as a Partially Observable Markov Decision Process (POMDP), in which the involved components are explained in detail.

A. DCB problem

The DCB problem involves two kernel objects: the flight information (demand) and the airspace sector configuration (capacity). A flight in this paper is represented by a spatio-temporal trajectory, in general, written as $f_i = \{(S_p, EntryTime_p), \cdots, (S_q, EntryTime_q)\}$, where i is the flight index, p and q are indexes of sectors, $p \neq \cdots \neq q$ and there is no strict order for them.

Taking a toy model as an example, there are 16 sectors S_0-S_{15} created and constructed as 4×4 squares. A sample flight $f_1 = \{(S_4, 10{:}22), (S_5, 10{:}32), (S_1, 10{:}52)\}$ is then simulated which means that the flight f_1 enters the sector S_4 at 10:22, sector S_5 at 10:32 and sector S_1 at 10:52. The matrix expression of the flight is $F_{1\times M}$, where M is the number of sectors, then N-flights plan is $F_{N\times M}$. In each sector, the flights inside also carry the information about other sectors. For instance, when considering f_1 in S_4, it also includes its information in sector S_5 and S_1. Hence, it is a compromise process to achieve the global observation about all interested sectors. The capacity of sectors depicts the maximum number of flights that can enter the sector within a specific period of time. In this paper, the capacity of each sector $C_{S,T}$ (S is the sector index, and T is the time period) is given in 20-minutes period.

An illustrative example used to understand the toy model is shown in Fig. 1. The time horizon (24 hours) are equally divided into 72 time periods, each combined with the configured 4×4 sectors, leading to 72 snapshots. Each snapshot contains the traffic flow of all sectors within 20 minutes. To clearly illustrate the scheme, an example of flight plan is given in (1):

$$\begin{cases} f_1 = & \{(S_4, 10{:}22), (S_5, 10{:}32), (S_1, 10{:}52)\} \\ f_2 = & \{(S_4, 10{:}05), (S_5, 10{:}25), (S_{10}, 10{:}45), (S_{11}, 11{:}05)\} \\ f_3 = & \{(S_{13}, 09{:}55), (S_9, 10{:}15), (S_5, 10{:}35), (S_6, 10{:}55), \\ & (S_2, 11{:}10\} \end{cases} \tag{1}$$

Above are three flights randomly generated with different number of traversed sectors. All flights are mapped into the snapshots. Assuming that the capacity of all sectors is 2, we can observe from Fig. 1 that sector S_5 in period T_{31} becomes a hotspot as there are three flights entering the sector during 10:00-10:20. The three flights can constitute a multi-agent system, and then some learning algorithm can be applied to the system to select which flight to delay and how long. In this paper, only the ground delay strategy is considered. Thus, if the learning algorithm outputs a delay result, such as $[10min, 0min, 0min]$, it means that only f_1 is delayed for 10 minutes. This delay result is then used to update the flight plan, where f_1 is updated to $f_1 = \{(S_4, 10{:}32), (S_5, 10{:}42), (S_1, 11{:}02)\}$, with f_2 and f_3 unchanged. This process is repeated until all hotspots are eliminated.

B. POMDP

Following the previous discussion, the DCB problem can be formulated as a Partially Observable Markov Decision Process (POMDP) for N agents [11], and defined by a tuple $\left(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, \gamma, \{\mathcal{O}^i\}_{i\in\mathcal{N}}\right)$, where $\mathcal{N} = \{1, \cdots, N\}$ is the agent set, $\mathcal{S}$ is the state set of all agents, $\mathcal{A}^i$ is the action space of agent i, and $\{\mathcal{A}^i\}_{i\in\mathcal{N}}$ means the joint action space. Let $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$, then the transition probability becomes $\mathcal{P} : \mathcal{S} \times \mathcal{A} \longrightarrow \mathcal{S}$. $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathcal{R}$ is the reward obtained from agent i, with discount factor $\gamma \in [0, 1]$. $\mathcal{O}^i$ is the observation set of agent i. Specifically,

a) Agent: Only flights in hotspots are regarded as the candidate agents. Since the flight number in each sector is different, it will change over training steps and the agent number is an unfixed value among steps. Agents are homogeneous in the designed multi-agent system, with a common reward function.

b) State, Observation: Agent i observes local information $o^i_{st} \in \mathcal{O}^i$ at each training step st from the global

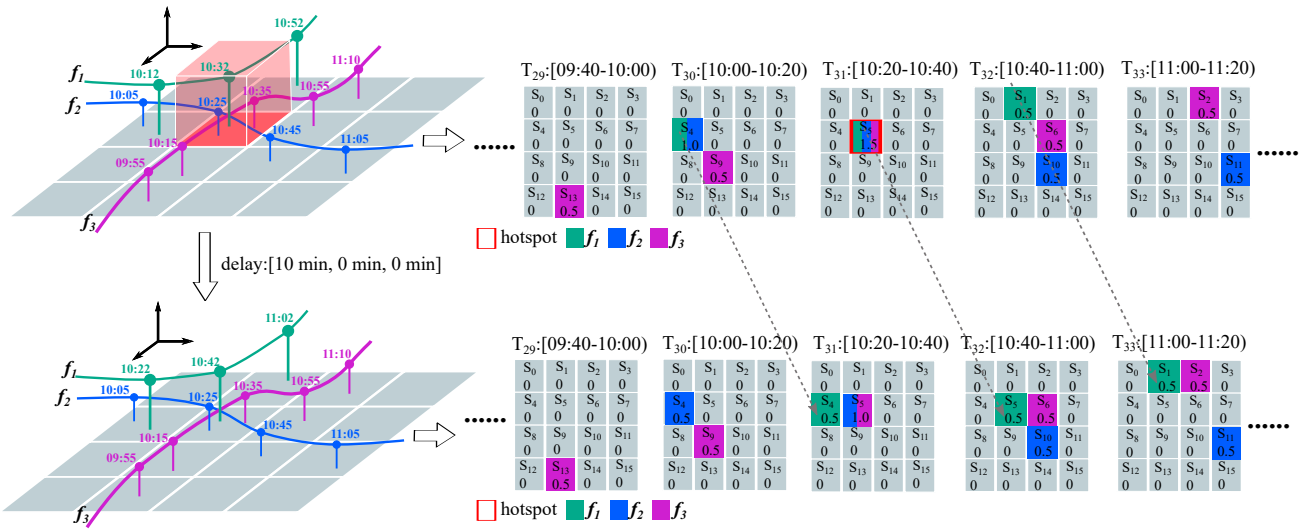

Fig. 1: Effects of delay on spatio-temporal trajectory information associated with traversed sectors.

environment state $S_{st} \in \mathcal{S}$ based on the observation function $\mathcal{S} \times \mathcal{N} \longrightarrow \mathcal{O}$. Each agent observes the information regarding its entry time, demand, capacity, over-demand status and sector index of all traversed sectors from the global state. Flights may traverse different number of sectors. According to the definition of observation, different agents have specific dimensions of their observation matrix.

c) Action: Each agent selects an action $a_{st}^i \in \mathcal{A}^i\,(i = 1, \cdots, N_{st})$, namely the exact delay time, from the action set, which together forms the joint action $a_{st} = \{a_{st}^i\} \in \mathcal{A}$ at each step st. As all agents are considered to be homogeneous, the action spaces can be naturally viewed as $\mathcal{A}^1 = \cdots = \mathcal{A}^{N_{st}}$.

d) Reward function: The goodness of the executed action is assessed by a reward function. The goal of all agents is to minimize the average or total delay, and in the meanwhile, eliminate all the hotspots. With the cooperative setting, all agents share one joint reward function which is also formed by all local observations and actions as in (2):

$$\mathcal{R}_{st}^i\left(\{o_{st}^i\}, \{a_{st}^i\}, \{o_{st+1}^i\}\right) = \alpha \cdot r_1 + \beta \cdot r_2 \tag{2}$$

where r_1 denotes the average delay of all agents as in (3), and r_2 is the relative change of hotspot number after executing the joint action as in (4). α and β are the coefficients. Here, $\alpha = 1$ and $\beta = 10$.

$$r_1 = -\left(\sum_0^{N_{st}} a_{st}^i\right)/N_{st} \tag{3}$$

$$r_2 = \left[hotspot(\{o_{st+1}^i\}) - hotspot(\{o_{st}^i\})\right] / hotspot(\{o_{st=0}^i\}) \tag{4}$$

where $hostpot(\cdot)$ calculates the hotpsot number.

All agents share the same function and get the same reward value in each step as follows.

$$\mathcal{R}_{st}^1 = \mathcal{R}_{st}^1 = \cdots = \mathcal{R}_{st}^{N_{st}} \tag{5}$$

e) State transition: The state transition function represents the probability of converting state $\mathcal{S}_{st}$ to $\mathcal{S}_{st+1}$ after executing the joint action a_{st}.

f) Discounted factor: The total discounted return of agent i at step st is $G_{st}^i = \sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{st+k}^i$. All the agents aim at maximizing the expected discounted return $\mathbb{E}\left[G_{st}^i\right]$. According to (5), all the agents have the same objective.

g) Policy: Each agent selects its action based on the policy $\pi^i : \mathcal{O}^i \longrightarrow \mathcal{A}^i$, and then the joint policy is $\pi = \{\pi^i\}$. The state value function and state-action value function are expressed in (6) and (7) respectively.

$$V^{\pi}\left(\mathcal{S}_{st}\right) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{st+k}^i \mid \mathcal{S}_0 = \mathcal{S}_{st}\right] \tag{6}$$

$$Q^{\pi}\left(\mathcal{S}_{st}, a_{st}\right) = \mathbb{E}[\mathcal{R}\left(\mathcal{S}_{st}, a_{st}, \mathcal{S}_{st+1}\right) + \gamma V^{\pi}\left(\mathcal{S}_{st+1}\right) \mid \mathcal{S}_0 = \mathcal{S}_{st}, a_0 = a_{st}] \tag{7}$$

The state value function $V^{\pi}\left(\mathcal{S}_{st}\right)$ is the expectation of $Q^{\pi}\left(\mathcal{S}_{st}, a_{st}\right)$ regarding action a_{st}. It has $V^{\pi}\left(\mathcal{S}_{st}\right) = \mathbb{E}_{a_{st}}[Q^{\pi}\left(\mathcal{S}_{st}, a_{st}\right)]$, and the Q-function can be rewritten as (6):

$$Q^{\pi}\left(\mathcal{S}_{st}, a_{st}\right) = \mathbb{E}_{\mathcal{S}_{st+1}}[\mathcal{R}\left(\mathcal{S}_{st}, a_{st}, \mathcal{S}_{st+1}\right) + \gamma \mathbb{E}_{a_{st+1}}[Q^{\pi}\left(\mathcal{S}_{st+1}, a_{st+1}\right)]] \tag{8}$$

If an action a^* makes $Q^{\pi}\left(\mathcal{S}_{st}, a^*\right) > V^{\pi}\left(\mathcal{S}_{st}\right)$, the return of executing a^* is higher than current policy π, and in turn the policy are supposed to be revised to increase the propability $\mathcal{P}(a^* \mid \mathcal{S}_{st})$.

III. Multi-agent Reinforcement Learning Frameworks

The proposed multi-agent reinforcement learning frameworks and their relevant specifications are presented in this section. The basic multi-agent asynchronous advantage actor-critic (MAA3C) framework is implemented at first. To improve its learning performance, unsupervised and supervised

learning are introduced and further integrated, resulting to enhanced MAA3C frameworks.

A. Multi-agent A3C Algorithm

The single-agent asynchronous advantage actor-critic algorithm (A3C) [12] is extended to a multi-agent version (MAA3C) in this paper. The advantage function is still estimated by $A^{\pi}(\mathcal{S}_{st}, a_{st}) = Q^{\pi}(\mathcal{S}, a) - V^{\pi}(\mathcal{S}) = \sum_{j=0}^{k-1} \gamma^{j} \mathcal{R}^{j}_{st+k} + \gamma^{k} V^{\pi}(\mathcal{S}_{st+k}) - V^{\pi}(\mathcal{S}_{st})$, which indicates the better performance of action a_{st} compared with the average level. Specifically, $A^{\pi} > 0$ means the action is better than the average.

The parameter of policy network in MAA3C is denoted as ϕ and of critic network is ϕ_v. The loss function consists of three parts: policy loss $J(\pi_{\phi'})$, value loss $L(\phi_v)$ and entropy regularization $-H(\pi_{\phi'}(a_{st} \mid \mathcal{S}_{st}))$, as below:

$$J(\pi_{\phi'}) = \mathbb{E}\left[\sum_{st=0}^{T} log\pi_{\phi'}(a_{st} \mid \mathcal{S}_{st}) A^{\pi}(\mathcal{S}_{st}, a_{st}; \phi, \phi_v)\right] \tag{9}$$

$$L(\phi_v) = \mathbb{E}\left[\frac{1}{2}\sum_{st=0}^{T}(R - V(\mathcal{S}_{st}; \phi_v))^2\right] \tag{10}$$

As a result, the loss function of MAA3C is denoted as in (11).

$$loss = J(\pi_{\phi'}) + L(\phi_v) + (-H(\pi_{\phi'}(a_{st} \mid \mathcal{S}_{st}))) \tag{11}$$

Some important features of the MAA3C framework include:

1) Recurrent structure: The policy network is constructed to estimate the action distribution, more specifically, the μ and σ values of normal distribution. All continuous actions are sampled from the distribution. The long short-term memory (LSTM) [13] is used as the basic layer of the neural networks to flexibly deal with the different number of traversed sectors of stacked observations. As depicted in Fig. 2, the first layer in MAA3C is shared, and three branches are formed on the second layer to approximate the policy parameters μ, σ and the value function $V(\mathcal{S}_{st}; \phi_v)$ respectively.

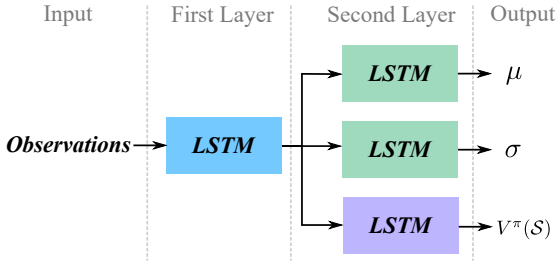

Fig. 2: Network structure in MAA3C framework.

The agent number varies with training steps, the batch size of the networks' input accordingly becomes dynamic as the agent number equals to the batch size. To some degree, a dynamic batch size can promote the learning process [14]. The structure of MAA3C, on the one hand, can be viewed as centralized if all agents' observations are stacked to be a batch. On the other hand, as the batch calculation can be also regarded as the loop operation of single agent's observation, the learning process can also be viewed as decentralized.

2) Parameter Sharing: Instead of initializing an individual policy for every agent, all agents share a single policy network and can then share the experience [15]. From the perspective of the network, it is also a single meta-agent network. Although the parameters are shared, the agent will get its unique output as each agent receives different observations [16]. With parameter sharing, the storage cost and training process can be optimized.

3) Stationarity Analysis: In POMDP, from agents' perspective, the multi-agent environment is non-stationary as other agents interacts with the environment. If the learning process is centralized , the concatenation of the observations of all agents is taken as the input and the critic can access the information from all agents. With the centralized critic, regardless of the actor being centralized or decentralized, the environment can be stationary [17].

4) Scalability: The observations of agents are stacked as the input matrix, and the dimension of this matrix will change as the number of agent varies. Since there is a single policy network, it can allow dynamic batch size, or in other words, any number of flights. More importantly, the scalability and generalization of the application process can be guaranteed by the state definition as well as the shared parameters.

The above features apply to the proposed MAA3C framework and also the following variations.

B. Unsupervised Learning

We define those flights in hotspots as candidate agents. It is not wise to have all of them equally treated, which will rise the number of delayed flights and costs. Thus, the next step is to find which flights are most critical to be delayed. From the perspective of cooperation and prioritization, two unsupervised learning methods including clustering and ranking are introduced to select the critical flights for delay.

1) Clustering: A hierarchical clustering mechanism is designed for agents' cooperation instead of constructing a structured network [18]. The unsupervised communication scheme is illustrated in Algorithm 1.

As also depicted in Fig. 3, the scheme consists of three steps:

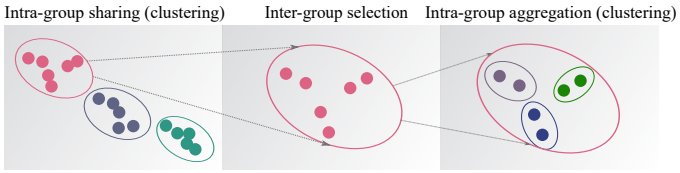

Fig. 3: Schematic of hierarchical clustering.

- **Intra-group sharing**. At first, the dimension of the observation matrix is reduced to 1×2 with principal component analysis (PCA) [19] for the 2-dimensional

Algorithm 1: Hierarchical Clustering

for $S \in hotspots$ **do**
  Reduce the dimension of observation matrix O_S: $M = PCA(O_S)$;
  First-level clustering: $\{C_1, C_2, \cdots, C_N\} = DBSCAN(M)$;
  Choose one cluster $C_i, i \in 1, \cdots, N$;
  Second-level clustering: $\{c_1, c_2, \cdots, c_k\} = DBSCAN(C_i)$;
  Get agent list L_S in cluster $c_1, \cdots, c_j (j \leq k)$;
  L.append(L_S)
Get observations O.

Algorithm 2: Unsupervised Ranking

for $f \in$ *flights in hotspots* **do**
  $f = \{(S_p, EntryTime_p), \cdots, (S_q, EntryTime_q)\}$;
  Delay the flight with $t \in \{1, 2, 3, 4, 5\}$ minutes and get f^t;
  Reset global score $sc^t \longleftarrow 0$;
  for $S \in [S_p, \cdots, S_q]$ **do**
    The influence of delaying the flight for S in period $T(EntryTime_S)$: $V^t_{S_1} = hotspot(S, T(EntryTime_S), f) - hotspot(S, T(EntryTime_S), f^N)$;
    The influence of delaying the flight for S in period $T(EntryTime_S) + 1$: $V^t_{S_2} = hotspot(S, T(EntryTime_S) + 1, f^N) - hotspot(S, T(EntryTime_S) + 1, f)$;
    Score of delay t minutes: $sc^t += V^N_{S_1} - 2 \times V^N_{S_2}$;
  Global Score of delaying flight f: $score_f = \sum_{t=1}^{5} sc^t$;
Sort all $score_f$ of all flights in the hotspots and choose the flights with higher scores.

clustering. The first-level clustering is conducted with the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method [20], as DBSCAN can find arbitrarily-shaped clusters without specifying the number of clusters in advance. Low-level agents can aggregate for high-level information based on the similarity of features.

- **Inter-group selection**. A typical cluster is selected as the candidate agents group.
- **Intra-group aggregation**. The second-level clustering is then conducted in the selected cluster using DBSCAN for internal refined information. Final agents list is selected from the sub-clusters.

In the clustering process, it can be regarded as the unintentional cooperation because all flights in the hotspots move into the same cluster only based on the density of the feature distributions. The selected flights of clustering are the agents. Their observations O are finally stacked as the input of MAA3C networks.

2) Ranking: To obtain the prioritization of all flights in the hotspots, ranking method is also taken into consideration. As conventional ranking algorithms are supervised [21] and the priority label can not be accessed, it is not feasible to directly apply the supervised ranking in our MAA3C framework. Hence, we transfer the supervised ranking to the unsupervised version and define a function as the evaluation metric.

- The metric definition: *the influence (the change of hotspot numbers) of delaying this flight*

All flights in the hotspots will be assessed with this metric, leading to the final score list. The flights with higher score are the better selections, because delaying this flight, to the maximum extent, can relieve the current congestions and avoid generating new hotspots. The unsupervised ranking scheme is presented in Algorithm 2 in detail. Each flight in the hotspot will try to delay itself of 1 to 5 minutes. As the flight contains the entry time of several sectors, when delaying the flight, the entry time for all traversed sectors will also change. For each traversed sector S, the hotspot status is calculated by $hotspot(\cdot)$ in the current period $T(EntryTime_S)$ and the adjacent period $T(EntryTime_S) + 1$.

Taking an example to explain the algorithm, as demonstrated in Fig. 4, suppose that sector S_p is a hotspot during the entry period $T(EntryTime_p)$ for flight f, the hotspot status is denoted by $hotspot(S_p, T(EntryTime_p), f) = 1$ for the entry period , and $hotspot(S_p, T(EntryTime_p) + 1, f) = 0$ for the next period $T(EntryTime_p) + 1$. If the flight is delayed by t ($t \in \{1, 2, 3, 4, 5\}$) minutes, the $hotspot(S_p, T(EntryTime_p), f) = 0$ as the current hotspot is eliminated after this flight is delayed. Let $hotspot(S_p, T(EntryTime_p) + 1, f) = 1$, which means that the delayed flight causes new hotspot in $T(EntryTime_p) + 1$ period. Hence, for the entry time period and its adjacent period, the hotspot status changes before and after the delay, which can reveal the influence of delaying this flight. Those flights, who can eliminate hotspots and not generate new hotspots if delayed, will have the higher priority to be delayed.

C. Supervised Learning

Supervised learning aims at narrowing the gap between the networks output and the trusted label, and improving the learning ability of the neural networks. If the supervised learning is involved, the initialization point of the reinforcement learning might be better [22].

In order to improve the training process, a supervised loss based on Computer-assisted Slot Allocation (CASA) solution [23] is introduced to update the network. The practical CASA algorithm in [23] is referred and adapted as shown in Algorithm 3. Some conditions have been simplified, keeping only the key principles for equivalent comparison with respect to the proposed methods.

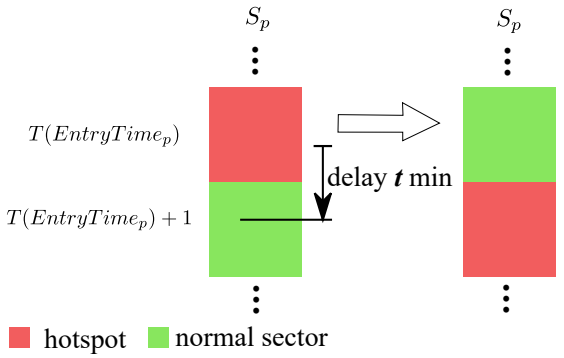

Fig. 4: Unsupervised ranking mechanism.

Algorithm 3: Adapted Computer-assisted Slot Allocation (CASA) Algorithm

Initialize computational round: $r = 0$;
repeat
  Get hotspots list $\{H\}$ based on the demand $D_{S,T}$ and capacity $C_{S,T}$;
  for *each sector* $S \in hotspots\{H\}$ **do**
    Get current period $t = T$ of S;
    Reset the accumulated capacity $AC = C_{S,t}$;
    Reset the accumulated demand $AD = D_{S,t}$;
    Reset the counter: $count = 1$;
    while $AC < AD$ **do**
      Include next period: $t = t + 1$;
      Accumulated capacity: $AC = AC + C_{S,t}$;
      Accumulated demand: $AD = AD + D_{S,t}$;
      $count+ = 1$;
    Rank all flights from period T to $T + count$ in the sector S;
    Divide the accumulated capacity AC into equal slots according the accumulated demand AD;
    Map all ranked flights into slots and get the delay result list DL_S in this sector;
  Compare delay results of all sectors and apply the maximum delay for each regulated flight.;
  $r = r + 1$;
until *Hotspots list* $\{H\} \in \oslash$ *or* $r > 100$;
Output delay result a_{casa} of all flights.

At each training step, the updated flight plan is fed into CASA, resulting to result a_{casa}. The estimated joint action a of MAA3C is compared with CASA result as the additional loss as in (12), which can also be regarded as the average difference value. Then, the overall loss function is written as in (13) with the CASA loss included.

$$C(\phi, \phi_v) = \sum (a - a_{casa})/N \tag{12}$$

$$loss = J(\pi_{\phi'}) + L(\phi_v) + (-H(\pi_{\phi'}(a_{st} \mid \mathcal{S}_{st}))) + C(\phi, \phi_v) \tag{13}$$

Finally, the proposed MAA3C frameworks with unsupervised learning and supervised learning involved are presented in Algorithm 4. To wrap up, four algorithms are constructed and will be compared:

Algorithm 4: MAA3C Framework

Initialize global parameters ϕ and ϕ_v, thread parameter $\phi^{'}$ and $\phi_v^{'}$, thread step counter $st \longleftarrow 1$;
repeat
  $d\phi \longleftarrow 0$ and $d\phi_v \longleftarrow 0$;
  Synchronize parameters in each thread with global parameters $\phi^{'} = \phi$ and $\phi_v^{'} = \phi_v$;
  $st_{start} = st$;
  Get all agents' observations O in hotspots with unsupervised learning methods;
  repeat
    Execute joint action a_{st} according to policy π;
    Get reward $\mathcal{R}_{st}$ and new state s_{st+1};
    $st \longleftarrow st + 1$;
  until *terminal* s_{st} *or* $st - st_{start} == st_{max}$;
  U=$\begin{cases} 0 & \text{for terminal } s_{st} \\ V(s_{st}, \phi_v^{'}) & \text{for non-terminal } s_{st} \end{cases}$;
  for $i \in \{st-1, \cdots, st_{start}\}$ **do**
    $U \longleftarrow \mathcal{R}_i + \gamma U$;
    Accumulate gradients wrt $\phi^{'}$: $d\phi \longleftarrow d\phi + \nabla_{\phi'} log\pi\left(a_i \mid s_i; \phi^{'}\right)\left(U - V\left(s_i; \phi_v^{'}\right)\right) + (\partial C(\phi^{'}, \phi_v^{'})/\partial \phi^{'})$;
    Accumulate gradients wrt $\phi_v^{'}$: $d\phi_v \longleftarrow d\phi_v + \partial\left(U - V\left(s_i; \phi_v^{'}\right)\right)^2/\partial \phi_v^{'} + (\partial C(\phi^{'}, \phi_v^{'})/\partial \phi_v^{'})$;
  Perform asynchronous update of ϕ using $d\phi$ and of ϕ_v using $d\phi_v$;
until $MaxSteps$;

- MAA3C + Clustering (MAA3C-C)
- MAA3C + Clustering + CASA Loss (MAA3C-C-C)
- MAA3C + Ranking (MAA3C-R)
- MAA3C + Ranking + CASA Loss (MAA3C-R-C)

IV. Experimental Analysis

To evaluate the proposed MAA3C frameworks, several simulated and real-world case studies are performed. We will evaluate the models trained on simulated small-scale scenarios with real large-scale scenarios in such a way to demonstrate the generalisation capability,

A. Simulated Case Study

A DCB simulator is developed to perform and visualize the training process. As shown in Fig. 5, 4×4 sectors are created and randomly generated flights are projected into the 72 time periods. With the capacity pre-defined, the ratio numbers of demand/capacity can be obtained and depicted in the boxes.

Fig. 5: DCB simulator developed to perform and visualize the training process.

Key information can be noticed from the interface, such as the training steps, hotspots information and capacity information.

The training flight plan includes 3000 simulated flights, as shown with Case 1 in TABLE I, the number of initial hotspots is 38 in spatio-temporal $4 \times 4 \times 72$ sectors, of which the capacity in any 20-minutes period is 23 constant, and there are 823 flights flying across at least one hotspots. All the proposed frameworks are trained with this plan to obtain models that can eliminate all the hotspots.

TABLE I. Cases of the study.

	Sectors Num	Flights Num	Capacity (/20 min)	Initial Hotspots	Initial Flights in Hotspots
Case 1	16	3000	23	38/1152	823/3000
Case 2	16	3000	23	49/1152	1074/3000
Case 3	16	5000	40	15/1152	625/5000
Case 4	376	8153	C_S	31/27072	293/8153

C_S: capacity varies with sectors.

Training return curves of all frameworks are shown in Fig. 6, the MAA3C+Ranking (+CASA Loss) frameworks have the higher return values in the training process compared with MAA3C+Clustering frameworks, regardless of the supervised loss is involved or not. Agents can learn to resolve the hotspots in the training process as depicted in Fig. 7, and the MAA3C+ Clustering (+CASA Loss) frameworks are stable to resolve all hotspots. The results will be assessed by the following metrics:

- Average delayed flights number
- Percentage of delayed flights number
- Average delay time of all flights
- Average delay time of delayed flights
- Percentage of resolved hotspots
- Flights delayed to the next day
- Computational cost
- Distribution of delays

To evaluate the effectiveness of the trained models, brand-new flight plans are generated or collected. Case 2 uses a simulated flight plan, in which the number of flights and that of sectors as well as the capacity value remain the same as used in the training, but the hotspots number is higher than Case 1. The test results are listed in TABLE II. Specially, the actor network in the MAA3C is to approximate the two parameters of the normal distribution, thus a random normal distribution is included to replace the network for comparison. MAA3C+Clustering+Random (MAA3C-C-R) and MAA3C+Ranking+Random (MAA3C-R-R) are also included for comparisons.

We can observe that the multi-agent reinforcement learning methods are better than CASA in average delay flights number, percentage of delayed flights, and average delayed time. The MAA3C+Ranking frameworks can reduce the delayed flights number significantly. In the MAA3C+Clustering and variations, the results achieve the best after the supervised CASA loss is included, but MAA3C+Ranking frameworks are not the same case.

Combined with Fig. 8, over 90% of delayed flights are delayed less than 20 minutes. There is no flight delayed to the next day and all hotspots can be eliminated except for the MAA3C-R-C. We can see that the supervised CASA loss is much more effective for MAA3C-C-C than for MAA3C-R-C. The computational time of multi-agent reinforcement learning algorithms is the average time of several test episodes and can be acceptable even if CASA can output its result in a very

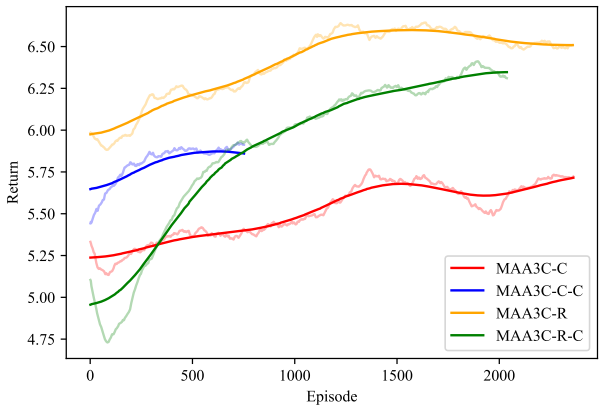

Fig. 6: Training return curves for different frameworks.

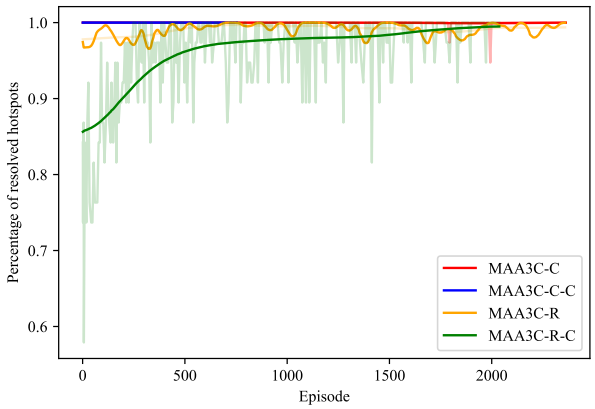

Fig. 7: Resolved hotspots ratios for different frameworks.

TABLE II. Test results on Case 2.

		MAA3C+Clustering			**MAA3C+Ranking**		
	CASA	**MAA3C-C**	**MAA3C-C-C**	**MAA3C-C-R**	**MAA3C-R**	**MAA3C-R-C**	**MAA3C-R-R**
Avg delayed flights num	1600	639	613	637	447	527	408
% of delayed flights	53.33%	21.31%	20.44%	21.23%	14.91%	17.58%	13.59%
Avg delay time (/ all flights) [min]	3.90	1.43	1.25	1.88	1.34	1.84	1.33
Avg delay time (/ delayed flights) [min]	7.31	5.65	6.40	8.67	8.08	9.74	10.33
% of resolved hotspots	100%	100%	100%	100%	100%	96.7%	100%
Flights delayed to the next day	0	0	0	0	0	0	0
Total computational time [s]	0.2	14.9	25.1	8.0	19.8	20.3	5.7

short time. It should be noted that the exact computational time of each method varies with the computational platform, but the minor difference can be ignored.

The scalability of the proposed algorithms is evaluated by larger-scale flight plans, as with Case 3 shown in TABLE I. The test number of flights is expanded from 3000 in the training case to 5000 for evaluation, but the hotspot ratio is decreased to 1.3% from 3.3%. As demonstrated in TABLE III, multi-agent reinforcement learning methods achieve better results than CASA, and, in particular, the delayed flights number is reduced to around $1/3 - 1/6$ of what CASA needs. If the total delay is divided by total flights number, multi-agent reinforcement learning methods can get the small average delay. In contrast, if it is divided by delayed flights number, each delayed flight is delayed for shorter time when using CASA, which means that multi-agent reinforcement learning methods output more delay time for each delayed flight.

The hotspots can be resolved by most of the methods, and only CASA delays one flight to the next day. The computational time of transferring trained models to the larger-scale are also reasonable. The delay distribution of the test case is shown in Fig. 9, nearly all flights can be delayed in less than 30 minutes no matter which algorithm is applied. All analyses above prove the feasibility and availability of directly using trained models on different flight population scales. Sometimes the random results are better since the randomly generated normal distribution is better than the network parameters.

B. Real World Case Study

The real world scenario consists of a large amount of sectors and flights. As a result, it is computationally challenging to train with the real data. Thus, it is valuable to train with the simulated data and then apply it to the real scenario, due

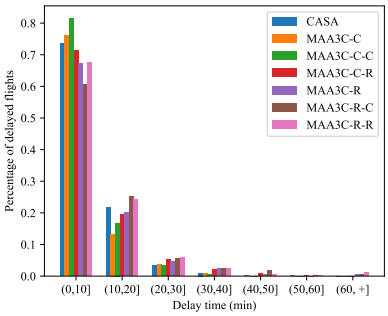

Fig. 8: Delay distribution of Case 2.

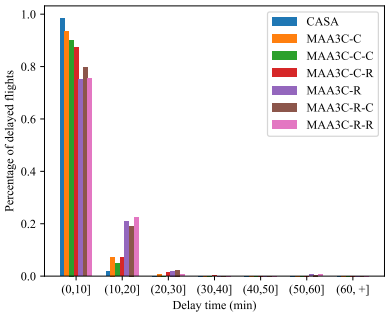

Fig. 9: Delay distribution of Case 3.

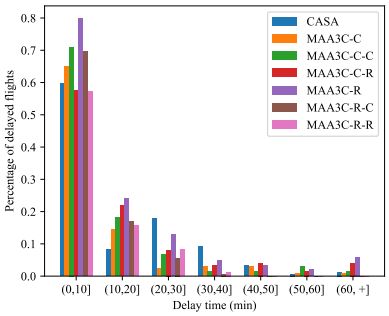

Fig. 10: Delay distribution of Case 4.

TABLE III. Test results on Case 3.

		MAA3C+Clustering			**MAA3C+Ranking**		
	CASA	**MAA3C-C**	**MAA3C-C-C**	**MAA3C-C-R**	**MAA3C-R**	**MAA3C-R-C**	**MAA3C-R-R**
Avg delayed flights num	1315	358	358	387	201	214	183
% of delayed flights	26.30%	7.16%	7.16 %	7.75%	4.02%	4.29%	3.66%
Avg delay time (/ all flights) [min]	0.82	0.27	0.26	0.44	0.28	0.31	0.27
Avg delay time (/ delayed flights) [min]	3.10	4.08	3.44	5.12	7.00	7.07	7.23
% of resolved hotspots	100%	100%	100%	100%	100%	98.7%	100%
Flights delayed to the next day	1	0	0	0	0	0	0
Total computational time [s]	0.4	19.7	25.4	17.6	15.4	17.5	18.5

to the fact that similar experience can be learned with lower computational cost. Hence, the models trained on Case 1 are tested on a real case, as shown with Case 4 in TABLE I. Data are collected from the EUROCONTROL Demand Data Repository v2 (DDR2).

The sector number is much increased compared with simulated cases. there are 376 sectors considered in French and Spanish airspace, and 8153 flights in total. Even more, the capacity is different for each sector rather than a fixed value used in simulated plans. Effective results are listed in TABLE IV where we can observe the feasibility of the trained models used for the real case. All proposed methods can achieve less delayed flights and average delay time than CASA, with all computational costs acceptable. Hotspots can be effectively eliminated. In Fig. 10, most of the delays produced by the proposed methods are within 20 minutes and better than CASA results.

It is an encouraging result that such operation, namely applying the models trained on small-scale simulated data to tackle real world large-scale DCB scenarios, can prove effective and cost-efficient. With the performance permitted, the proposed frameworks can be suitable for any different flight plans, sector configurations and capacity settings.

V. Conclusion

In this paper, the DCB problem is formulated as a multi-agent system where hundreds or thousands agents collaboratively work to minimize the average delay and eliminate airspace hotspots. The flight is represented by spatio-temporal trajectory and only flights in hotspots are regarded as the candidate agents. The asynchronous advantage actor-critic reinforcement learning algorithm is extended to multi-agent version (MAA3C). To promote the cooperation and collaboration of agents, the unsupervised learning methods including clustering and ranking are then introduced. Supervised CASA loss is also incorporated and expected to improve the learning results. Experiments on the simulated and real case studies show the effectiveness of the proposed frameworks. Trained models can easily deal with dynamic flight plans including the change of flight number, sector number and capacity.

It is not quite clear to conclude the effects of introducing supervised CASA loss, as sometimes the results are better

TABLE IV. Test results on Case 4.

		MAA3C+Clustering			MAA3C+Ranking		
	CASA	**MAA3C-C**	**MAA3C-C-C**	**MAA3C-C-R**	**MAA3C-R**	**MAA3C-R-C**	**MAA3C-R-R**
Avg delayed flights num	473	131	132	124	187	159	159
% of delayed flights	5.80%	1.60%	1.62%	1.52%	2.29%	1.96%	1.95%
Avg delay time (/ all flights) [min]	0.81	0.17	0.16	0.20	0.24	0.16	0.21
Avg delay time (/ delayed flights) [min]	13.96	8.03	10.84	14.92	19.22	6.73	7.35
% of resolved hotspots	100%	100%	100%	100%	100%	100%	100%
Flights delayed to the next day	80	80	80	80	80	80	80
Total computational time [s]	12.1	39.4	43.5	34.2	29.8	31.2	34.6

when this loss is involved in MAA3C+Clustering framework, but sometimes better in MAA3C+Ranking framework. Further investigation in this regard is required in our future work. Besides, we will take into account the aspects of rerouting and dynamic airspace structures. In addition, it is also necessary to improve the fineness of the actions for more practical applications.

REFERENCES

[1] D. Bertsimas and S. S. Patterson, "The air traffic flow management problem with enroute capacities," *Operations research*, vol. 46, no. 3, pp. 406–422, 1998.

[2] P. L. de Frutos, R. R. Rodríguez, D. Z. Zhang, S. Zheng, J. J. Cañas, and E. Muñoz-de Escalona, "Cometa: An air traffic controller's mental workload model for calculating and predicting demand and capacity balancing," in *International Symposium on Human Mental Workload: Models and Applications*. Springer, 2019, pp. 85–104.

[3] D. Gianazza, "Forecasting workload and airspace configuration with neural networks and tree search methods," *Artificial intelligence*, vol. 174, no. 7-8, pp. 530–549, 2010.

[4] Y. Zhang, R. Su, Q. Li, C. G. Cassandras, and L. Xie, "Distributed flight routing and scheduling for air traffic flow management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2681–2692, 2017.

[5] C. Spatharis, T. Kravaris, G. A. Vouros, K. Blekas, G. Chalkiadakis, J. M. C. Garcia, and E. C. Fernandez, "Multiagent reinforcement learning methods to resolve demand capacity balance problems," in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, 2018, pp. 1–9.

[6] C. Spatharis, A. Bastas, T. Kravaris, K. Blekas, G. A. Vouros, and J. M. Cordero, "Hierarchical multiagent reinforcement learning schemes for air traffic management," *Neural Computing and Applications*, pp. 1–13, 2021.

[7] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1774–1783.

[8] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *The World Wide Web Conference*, 2019, pp. 983–994.

[9] J. Kim, "Optimizing large-scale fleet management on a road network using multi-agent deep reinforcement learning with graph neural network," *arXiv preprint arXiv:2011.06175*, 2020.

[10] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[11] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Q. Ye, Y. Zhou, M. Shi, Y. Sun, and J. Lv, "Dbs: Dynamic batch size for distributed deep neural network training," *arXiv preprint arXiv:2007.11831*, 2020.

[15] J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *arXiv preprint arXiv:1605.06676*, 2016.

[16] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[17] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," *arXiv preprint arXiv:1906.04737*, 2019.

[18] J. Sheng, X. Wang, B. Jin, J. Yan, W. Li, T.-H. Chang, J. Wang, and H. Zha, "Learning structured communication for multi-agent reinforcement learning," *arXiv preprint arXiv:2002.04235*, 2020.

[19] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Density-based spatial clustering of applications with noise," in *Int. Conf. Knowledge Discovery and Data Mining*, vol. 240, 1996, p. 6.

[21] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for list-wise recommendations," *arXiv preprint arXiv:1801.00209*, 2017.

[22] P.-H. Su, P. Budzianowski, S. Ultes, M. Gasic, and S. Young, "Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management," *arXiv preprint arXiv:1707.00130*, 2017.

[23] EUROCONTROL, "Atfcm operations manual - network manager."