In: 2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC), San Antonio, 3-7 October 2021
DOI:10.1109/DASC52595.2021.9594329

Multi-Agent Deep Reinforcement Learning for Solving Large-scale Air Traffic Flow Management Problem: A Time-Step Sequential Decision Approach

1st Yifan Tang
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedford, United Kingdom
yifan.tang@cranfield.ac.uk

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

***Abstract*—In this paper, we focus on the demand-capacity balancing (DCB) problem in air traffic flow management, which is considered as a fully cooperative multi-agent learning task. First, a rule-based time-step environment is designed to mimic the DCB process. In this environment, each agent 'flight' decides its action at valid time steps. Three different rules are defined, based on the remaining capacity and the number of cooperative flights in each sector, to ease the learning process. Second, a multi-agent reinforcement learning framework, built on the proximal policy optimization (MAPPO), is proposed by using the parameter sharing mechanism and the mean-field approximation method, where an inherent feature of all other agents is extracted to address the credit assignment problem. Moreover, a supervisor integrated MAPPO framework is proposed, where a supervisor is designed to generate supervised actions, in such a way to further improve the learning performance. In the experiments, two performance indices, *Search Capability* and *Generalization Capability*, are considered. Both indices are assessed with the evaluation of two toy cases and a real-world case study. Results suggest that, the supervisor integrated MAPPO with supervised actions achieves the best performance across the different cases; other proposed methods also show some promising *Search Capability*, but only prove an acceptable *Generalization Capability* in simpler cases than the training cases.**

***Keywords*—Air Traffic Flow Management; Demand-capacity Balance; Multi-agent Reinforcement Learning; Proximal Policy Optimization**

I. Introduction

With the fast growing of air transportation in the last decades, the imbalance of traffic demand and airspace capacity has been one of the bottlenecks of today's air traffic management (ATM) systems. The airspace has become more congested, and as a consequence recent years often saw record-breaking flight delays across the world. It was reported that in 2018 and 2019, 30.2 and 33.0 billion extra costs of flight delay were incurred in the US [1]. Also in 2019, an average departure delay of 13.1 min and 14 min were observed in Europe [2] and China [3] respectively.

In response to this issue, the demand-capacity balancing (DCB) problem has been widely studied for air traffic flow management (ATFM) [4]. Following the pioneering work in [5], different methodologies have been proposed such as the single/multi-airport ground-holding problem and the rerouting problem [6]. Bertsimas et al. [7] built a novel IP model to solve the large-scale ATFM problem, seeking the optimal combination of different strategies such as ground-holing, rerouting, speed control. Based on this model, Xu et al. [8] proposed a collaborative ATFM framework to complete the traffic flow optimization and the airspace configuration scheduling simultaneously.

In addition to the above exact search methods, different heuristic and meta-heuristic methods have been explored [9]. Taylor et al. [10] considered the design variable as a chromosome, and proposed an improved multi-objective genetic algorithm for automatic design of ATFM strategies. Marina [11] developed an airspace sectorization framework with genetic algorithm and k-means clustering algorithm, aimed to reduce the congestion of the airspace systems. Xiao et al. [12] proposed a hybridized indirect and direct encoding genetic algorithm for producing the ATFM solution.

Different from the above works formulating the DCB problem as an optimization problem, several studies have reformulated the problem as a learning task. Based on the air traffic simulator FACET [13], Kagan et al. [14] transferred the ATFM problem as a multi-agent reinforcement learning (MARL) task, where each individual ground location is defined as an agent, and the action is the separation between aircraft. The ϵ-greedy Q learning [15] method was utilized, and the trained model was compared with the Monte Carlo estimation in two simulated scenarios. Besides, a research team of the DART project [16] [17] reformulated the DCB problem with ground-holding strategy to a Markov Decision Process, where each flight is defined as an agent, and the action is whether to impose delay to a flight. Based on this formulation, different MARL frameworks, i.e., the edge-based MARL method and the agent-based MARL in [16], the hierarchical MARL framework in [17], have been tested and evaluated in real-world large-scale ATFM scenarios.

While numerous trails have been conducted using MARL methods to solve the DCB problems, up to the best of our knowledge, few discussion has been made towards the generalization issue of those attempts. In other words, it is not clear to us if the models trained with a specific ATFM scenario can be applied to handle a completely new scenario. As a result, in this paper, we aim to solve the DCB problem

This work is 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.

with MARL method such that some of the complexity of the problem can be addressed offline, i.e., in the learning phase., and also discuss the generalization issue in detail. The main contributions are summarized as follows.

- The DCB problem is transformed into a rule-based time-step environment, where each flight is to chronologically select action "*Holding*" or "*Departure*" at each valid time step. Three different rules are devised to filter the candidate flights to ease the learning task.
- An MARL framework and its variants are proposed based on the proximal policy optimization method. The ideas of supervised learning are also integrated to further improve the learning performance.
- The *Generalization Capability* and the *Search Capability* of the proposed approach are evaluated, via using the trained model/policy to directly tackle two toy cases with different flight population size, as well as a new real-world scenario.

The remainder of this paper is structured as follows: Section II describes the rule-based time-step environment and the formulated Markov Decision Process. Section III gives a brief introduction to reinforcement learning background considered in this work. Section IV presents the proposed MARL frameworks, including the baseline MAPPO method, and supervisor integrated MAPPO methods. The learning performances are assessed and compared with standard solutions in Section V. Section VI summarizes the present work and envisions our future work.

II. ATFM PROBLEM REFORMULATION

In this section, a rule-based time-step environment is introduced to mimic the conventional ATFM decision making process, which is further formulated as a partially observable Markov Decision Process (MDP).

A. Time-step Environment

Given a set of flights $f \in \mathcal{F}$, a set of sectors $j \in \mathcal{J}$, a set of time steps $t \in \mathcal{T}$, a set of time periods $\tau \in \Gamma$, and sector capacity at each time period c_j^τ, the initial flight plan FP_f of flight f can be represented as a combination of crossing sectors and their corresponding scheduled arrival times. The objective of the considered ATFM problem (with ground-holding strategy) is to minimize the total ground delay while resolving any hotspot where the traffic demand is higher than the sector capacity during any period of time.

This particular problem can be reformulated in a way that, at every single time step elapsed from the beginning to the end of the time horizon, each valid flight that has been planned to take off, needs to decide chronologically whether to hold or depart now. Thus, the objective is to have as few hold actions as possible for every flight, whereas no hotspots will be incurred.

A framework for building such environment is shown in Fig. 1. The Flight Plan Database and the whole airspace sectors are initialized using the original flight plans and an empty airspace respectively. At time step t, each flight f on the ground with a scheduled departure time $d_f \leq t+1$ is defined as a candidate flight. Then, a rule-based decision mechanism, including *Departure*, *Holding* and *Cooperation*, is designed to select the action of that candidate flight.

- *Holding*: When it foresees that its departure must cause a hotspot by reviewing the remaining available capacity of each sector appearing in its flight plan, the flight prefers to hold, which means that the ground delay increases by one time step;
- *Departure*: When the remaining capacity of each sector appearing in the flight plan is larger than the number of cooperative flights (i.e., all candidate flights planned to enter the same sector in the same time period), the flight prefers to depart.
- *Cooperation*: When the remaining capacity of any sector appearing in the flight plan is less than the number of cooperative flights, the *Cooperation* rule is activated, meaning that the associated flights need to cooperate to decide their joint actions.

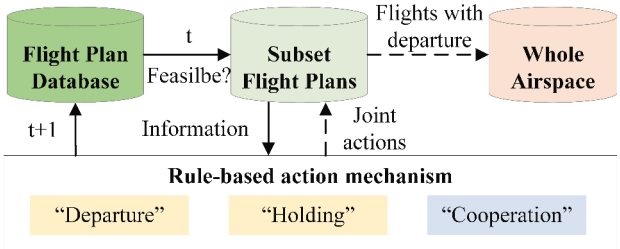


Fig. 1: Framework of rule-based time-step environment.

After performing the aforementioned mechanism, all candidate flights selecting *Departure* at time step t will enter their first sector at the next time step $t+1$. Their corresponding flight plans are added to the whole airspace to update the *Entry Count* and the remaining capacity of each sector at different time periods. Besides, the candidate flights with action *Holding* still maintain as the candidates for the next time step, and their flight plans stored in the Flight Plan Database are updated by adding one time step to the scheduled arrival time of each sector.

B. Partially Observable MDP

The above environment can be modelled as a partially observable Markov Decision Process $G = (N, S, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where N is the number of agents, S is the set of states, $\mathcal{A}$ is the joint action space, $\mathcal{P}$ is the transformation probability, $\mathcal{R}$ is the reward functions, and γ is the reward discount. The definitions of these elements in this environment are summarized as follows.

- *Agent*: To ease the problem's complexity, only candidate flights performing the *Cooperation* rule at time step t are defined as agents $\boldsymbol{Ag}_t = \{Ag_t^i | i = 1, ..., N_t\}$. Considering the varying size of candidate flights at different time steps, the number of agents N_t is also changing over time. Also, the agents in *Cooperation* are considered to be homogeneous, aiming to experience the least delay while solving the hotspots cooperatively.
- *State*: The local state s_t^i of each agent Ag_t^i at time step t includes the corresponding flight plan stored in the Flight

Plan Database, the number of ground delay the flight has taken, the remaining capacity of each sector, and the number of cooperative flights in each sector. Meanwhile, the joint state of all other agents at the same time step is denoted as $\boldsymbol{s}_t^{-i} = \{s_t^j | j = 1, ..., i-1, i+1, ..., N_t\}$. With a partial observed MDP, each agent has no direct observation of other agents. The structure of state s_t^i is depicted as shown in Fig. 2, where N_j is the maximum number of sectors. Only the crossed sectors by the agent have values subject to update, whereas the values for other sectors are fixed to 0.

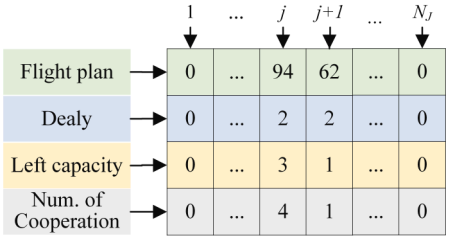


Fig. 2: Structure of the state matrix.

- *Action*: At time step t, each agent Ag_t^i has two action choices $a_t^i \in \{0, 1\}$. If $a_t^i = 0$, the agent takes the *Holding* action, and one time step is added to its ground delay as well as the scheduled arrival time of each sector in its flight plan. If $a_t^i = 1$, the agent chooses departure action, and the corresponding flight plan is activated to update the whole airspace. The joint action of all agents at current time step is represented as $\boldsymbol{a}_t = \{a_t^i | i = 1, ..., N_t\}$.
- *Reward*: The reward definition consists of two parts, an instant reward and a common reward. At each time step t, agent Ag_t^i gets an instant reward $r_{ins}^{t,i}$ with regard to its action a_t^i and the current joint action $\boldsymbol{a}_t$.

$$r_{ins}^{t,i} = \underbrace{\alpha_1 \times (1 - a_t^i)}_{\text{delay reward}} + \underbrace{\alpha_2 \times a_t^i \times Overload(\boldsymbol{a}_t)}_{\text{overload reward}} \quad (1)$$

where α_1 and α_2 are two negative coefficients. The delay reward depends on its own action, while the overload reward is calculated based on the overload increase caused by the joint action. The overload during the whole periods is defined as the summary of the excess part when the *Entry Count* is larger than the capacity of sector j during time period τ. Once all the flights have departed, a final common reward r_{com} is calculated as:

$$r_{com} = \alpha_3 \times Overload_{final} \times e^{Delay_{average}} \quad (2)$$

where α_3 is a negative coefficient; the average delay is the value of total delay divided by the number of all flights. The common reward is added to all agents stored. Thus, the final reward of agent Ag_t^i at time step t is computed as the sum of $r_{ins}^{t,i} + r_{com}$.
- *State transition probability*: The state transition probability $p(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ for joint state $\boldsymbol{s}_t$ is deterministic assuming no stochastic events. In other words, given a joint action $\boldsymbol{a}_t$, the joint state $\boldsymbol{s}_{t+1}$ transited from the previous state $\boldsymbol{s}_t$ is known in the current environment. However, for the local state of each agent, the transition probability is stochastic considering that each agent has no information of other agents' actions.

Based on the above description, the objective of minimising *Ground Delay* and the constraint of *Hotspot Resolution* for the ATFM problem will be reformulated as the partially observable MDP, aimed to achieve the maximum total reward shown as

$$max \sum_t \sum_i (r_{ins}^{t,i} + r_{com}) \quad (3)$$

which is expected to be solved by the reinforcement learning methods in this paper.

III. Reinforcement Learning Background

This section gives a brief introduction to the proximal policy optimization (PPO) method, and then reviews some relevant approaches of multi-agent reinforcement learning used in this paper.

A. Proximal Policy Optimization

The policy gradient methods update the policy parameters by solving an estimated objective function with the stochastic gradient ascend algorithm. The general formulation of the estimated function is:

$$\hat{g} = \hat{\mathbb{E}}_t[\bigtriangledown log \pi_\theta(a_t | s_t) \hat{A}_t] \quad (4)$$

where, $\hat{\mathbb{E}}_t[.]$ means the average value over a batch of samples; $\pi_\theta(a_t|s_t)$ is the policy; $\hat{A}_t$ is the estimated advantage value; s_t, a_t are the state and action sampled from the memory;. To limit the divergence between the new policy and the old policy, a clipped surrogate objective function is proposed in the PPO method [18],

$$L_\theta^{CLIP} = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, \mathcal{F}^{CLIP}(r_t(\theta), \epsilon)\hat{A}_t)] \quad (5)$$

where, $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$ is the ratio of action probability obtained by the new policy and the old policy; θ and θ_{old} are the trainable hyperparameters for the new and old policy respectively; $\epsilon \in [0, 1]$ is the clipping parameter; the clipping function in PPO is defined as Eq. (6).

$$\mathcal{F}^{CLIP}(r_t(\theta), \epsilon) = \begin{cases} 1 - \epsilon & r_t(\theta) < 1 - \epsilon \\ 1 + \epsilon & r_t(\theta) > 1 + \epsilon \\ r_t(\theta) & otherwise \end{cases} \quad (6)$$

However, the work in [19] proved that the PPO method fails in restricting the probability ratio $r_t(\theta)$ within the clipped range $[1 - \epsilon, 1 + \epsilon]$ strictly for some tasks. To address this limitation, a clipping function with a rollback function is proposed in [19]:

$$\mathcal{F}^{RB}(r_t(\theta), \epsilon, \beta) = \begin{cases} -\beta r_t(\theta) + (1 + \beta)(1 - \epsilon) & r_t(\theta) \leq 1 - \epsilon \\ -\beta r_t(\theta) + (1 + \beta)(1 + \epsilon) & r_t(\theta) \geq 1 + \epsilon \\ r_t(\theta) & otherwise \end{cases} \quad (7)$$

where, β is a positive constant parameter determining the effect of the rollback. Different from the clipping function in Eq. (6), the rollback function generates a negative feedback when $r_t(\theta)$ locates outside the range $[1-\epsilon, 1+\epsilon]$, which is more promising

to weaken the feedback derived from the objective function L_θ^{CLIP}. The reformulated surrogate objective function L_θ^{RB} is considered in this paper.

$$L_\theta^{RB} = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, \mathcal{F}^{RB}(r_t(\theta), \epsilon, \beta)\hat{A}_t)] \tag{8}$$

B. Multi-agent Reinforcement Learning

Given the fact that the direct interactions between agents and the environment coexist with the potential interactions among agents, multi-agent reinforcement learning is faced with several challenges, such as non-stationary and credit assignment [20]. To this end, some approaches including communication [21], value function or task decomposition [22] and mean-filed regime [23], have been explored and applied in different fields and tasks.

The ideas of parameter sharing and mean-field regime are briefly introduced as follows.

- *Parameter Sharing*: In the parameter sharing approach, all homogeneous agents share a common policy, which allows the policy to be trained with the experience of all agents simultaneously [24]. For each agent, the shared policy provides the action according to its own observation, which maintains the exploration capability.
- *Mean-field Regime*: For a discrete action space, the dimension of joint action increases exponentially with regard to the number of agents, which makes the *Curse of Dimensionality* more serious. To tackle the scalability and the credit assignment issues caused by this problem, the mean-field regime mechanism simplifies the interactions of each agent with other agents by some mean-field quantities, such as the average action [25], the empirical distribution of other agents' states [26].

Following these ideas, the existing PPO method will be further extended to multi-agent PPO, as discussed below.

IV. Multi-agent Proximal Policy Optimization

To tackle the cooperative multi-agent task presented in section II, a baseline multi-agent proximal policy optimization (MAPPO) method is discussed. Then, the principle of supervised learning is incorporated to form a supervisor integrated MAPPO so as to improve the training performance.

A. Baseline MAPPO

Inspired from the parameter sharing and mean-field regime mechanisms, a baseline MAPPO is proposed by extending the conventional PPO framework to a multi-agent version, where several adaptations are included to fit the features of the environment.

Considering the varying agent size in the rule-based time-step environment, a common actor $\pi_\theta(\cdot)$ and a common critic $V_\omega(\cdot)$ for state-value are both shared across all agents, which makes the proposed method applicable to different ATFM scenarios with different flight plans. Besides, to ease the credit assignment problem among agents, a mean state feature $\phi(\boldsymbol{s}_t^{-i})$ is used to indicate the implicit interaction between the agent Ag_t^i and all other agents $\boldsymbol{Ag}_t^{-i} = \{Ag_t^j | j = 1, ..., i-1, i+1, ...N_t\}$ at the same time step. The mean state feature $\phi(\boldsymbol{s}_t^{-i})$ is extracted by a Recurrent Neural Network (RNN), and is integrated in the actor and the critic, whose structures are both depicted in Fig. 3.

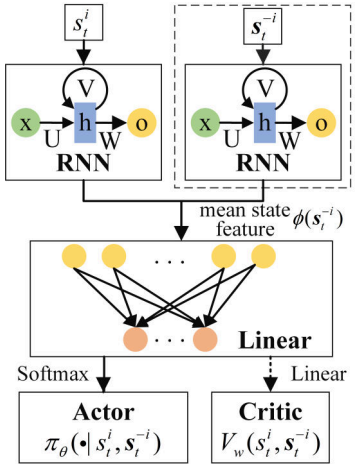


Fig. 3: Structure of the actor and critic network.

The only difference between the actor and the critic is the output layer, where the Soft-max layer is applied in the actor $\pi_\theta(\cdot|s_t^i, \boldsymbol{s}_t^{-i})$ to calculate the probability of each action (*Holding* and *Departure*), whereas a linear layer is used in critic $V_\omega(s_t^i, \boldsymbol{s}_t^{-i})$ to obtain the sate value. Using this structure, the state value and the action of agent Ag_t^i are produced taking into account the implicit interaction with other agents as expected. To enable the trade-off between exploration and exploitation, the action a_t^i of each agent is selected by the ε-greedy strategy:

$$a_t^i = \begin{cases} \sim \pi_\theta(\cdot|s_t^i, \boldsymbol{s}_t^{-i}) & \mu \geq \varepsilon + \dfrac{(1-\varepsilon)I_c}{I_m} \\ argmax(\pi_\theta(\cdot|s_t^i, \boldsymbol{s}_t^{-i})) & otherwise \end{cases} \tag{9}$$

where, $\mu \in [0, 1]$ is generated randomly for each agent at each episode and at each time step; ε is the initial greedy coefficient; $I_c \in [0, I_m]$ is the current iteration; I_m is the defined maximum iteration. When the random number is larger than the modified greedy coefficient $\varepsilon + \frac{(1-\varepsilon)I_c}{I_m}$, the action a_t^i of agent Ag_t^i is sampled based on the action probability $\pi_\theta(\cdot|s_t^i, \boldsymbol{s}_t^{-i})$; otherwise, the action is set as the one with the maximum probability.

For each agent Ag_t^i, the critic is parameterized as the function $V_\omega(s_t^i, \boldsymbol{s}_t^{-i})$, where ω is the trainable hyperparameter. Its advantage value $\hat{A}_t^i$ is calculated by the truncated version of generalized advantage estimation equation [27]:

$$\hat{A}_t^i = \delta_t^i + (\gamma\lambda)\delta_{t+1}^i + ... + (\gamma\lambda)^T\delta_{t+1}^i \approx \delta_t^i \tag{10}$$

where δ_t^i is calculated as $(r_{ins}^{t,i} + r_{com}) + \gamma V_{\omega_{old}}(s_{t+1}^i, \boldsymbol{s}_{t+1}^{-i}) - V_{\omega_{old}}(s_t^i, \boldsymbol{s}_t^{-i})$; ω_{old} is the parameter of the old critic. In practical implementation, all agents data are stored in an universal memory $\mathcal{M}$. Thus, the surrogate objective function calculated based on a sampled batch is expressed as:

$$\begin{aligned} L_\theta^{RB} &= \hat{\mathbb{E}}_t[min(r_t^i(\theta)\hat{A}_t^i, \mathcal{F}^{RB}(r_t^i(\theta), \epsilon, \beta)\hat{A}_t^i)] \\ &\approx \hat{\mathbb{E}}_t[min(r_t^i(\theta)\delta_t^i, \mathcal{F}^{RB}(r_t^i(\theta), \epsilon, \beta)\delta_t^i] \end{aligned} \tag{11}$$

where, $r_t^i(\theta) = \pi_\theta(a_t^i|s_t^i, \boldsymbol{s}_t^{-i})/\pi_{\theta_{old}}(a_t^i|s_t^i, \boldsymbol{s}_t^{-i})$. The common actor is updated by maximizing the surrogate objective function, as follows.

$$\Delta\theta = \bigtriangledown_\theta L_\theta^{RB} \tag{12}$$

Algorithm 1: Baseline MAPPO

Initialize common critic π_θ and common actor V_ω ;
Initialize the old critic $\pi_{\theta_{old}} \leftarrow \pi_\theta$, and the old critic $V_{\omega_{old}} \leftarrow V_\omega$;
Initialize a memory buffer $\mathcal{M}$;
for *iteration* $I_c = 1, ..., I_m$ **do**
  Reset the environment;
  Get the initial state of each agent;
  for *time step* $t = 1, ..., T$ *in an episode* **do**
    Get action $\{a_t^i\}^{i=1,..,N_t}$ by Eq. (9) with the old actor $\pi_{\theta_{old}}(\cdot|s_t^i, \boldsymbol{s}_t^{-i})$;
    Perform $\boldsymbol{a}_t$;
    Get $r_{ins}^{t,i}$, s_{t+1}^i, $\boldsymbol{s}_{t+1}^{-i}$;
    Store $(s_t^i, \boldsymbol{s}_t^i, a_t^i, r_{ins}^{t,i}, s_{t+1}^i, \boldsymbol{s}_{t+1}^i))$ in $\mathcal{M}$
  end
  Calculate the common reward r_{com} as Eq. (2);
  Store the common reward to $\mathcal{M}$;
  Get advantage value $\{\hat{A}_t^i\}_{t=1,...,T}^{i=1,...,N_t}$ by Eq. (10) ;
  Compute $\{y_t^i = \hat{A}_t^i + V_{\omega_{old}}(s_t^i, \boldsymbol{s}_t^{-i})\}_{t=1,...,T}^{i=1,...,N_t}$;
  for $k = 1, ..., K$ **do**
    for $j = 1, ..., J$ **do**
      Sample a minibatch from the memory $\mathcal{M}$;
      Calculate L_θ^{RB} by Eq. (11);
      Calculate L_ω by Eq. (13);
      Update θ using $\Delta\theta = \bigtriangledown_\theta L_\theta^{RB}$ by Adam;
      Update ω using $\Delta\omega = -\bigtriangledown_\omega L_\omega$ by Adam;
    end
  end
  Update $\pi_{\theta_{old}} \leftarrow \pi_\theta$, and $V_{\omega_{old}} \leftarrow V_\omega$;
  Empty the memory buffer $\mathcal{M}$;
end

Besides, the conventional citric update procedure in PPO is applied to modify the common critic in the baseline MAPPO method. The critic loss is also calculated based on the sampled batch from the common memory $\mathcal{M}$, shown as:

$$L_\omega = MSE(y_t^i, V_\omega(s_t^i, \boldsymbol{s}_t^{-i})) \tag{13}$$

where, MSE is the mean square error function; $y_t^i = \hat{A}_t^i + V_{\omega_{old}}(s_t^i, \boldsymbol{s}_t^{-i})$. The common critic is updated by minimizing the critic loss, where the critic parameter is update as below.

$$\Delta\omega = -\bigtriangledown_\omega L_\omega \tag{14}$$

The overall pseudo code of baseline MAPPO is detailed in Algorithm 1, and more details are presented as follows.

The algorithm starts from initialization including the common critic π_θ, the common actor V_ω, the old actor $\pi_{\theta_{old}}$, the old critic $V_{\omega_{old}}$, and an empty memory buffer. In the data collection procedure, the action a_t^i of each agent Ag_t^i is determined by the old actor $\pi_{\theta_{old}}(a_t^i|s_t^i, \boldsymbol{s}_t^{-i})$ and the ε-greedy strategy. After performing the current joint action $\boldsymbol{a}_t$, the environment is updated, and outputs the instant reward r_{ins}^i, and the state information (e.g., s_{t+1}^i, and $\boldsymbol{s}_{t+1}^{-i}$). The information of different agents is stored in the memory buffer $\mathcal{M}$ as a whole.

Based on the data stored in $\mathcal{M}$, the final common reward r_{com}, and the estimated advantage value $\hat{A}_t^i$ are calculated based on Eq. (2) and Eq. (10) respectively. Given the total epoch K, and the size of minibatch B, the max update iterations J in each epoch is determined by the total sample size $N_\mathcal{M}$ in the memory, i.e., $J = N_\mathcal{M}/B$. During the iteration, a mini-batch is sampled from the memory to calculate the surrogate objective L_θ^{RB} and the critic loss L_ω, which are both used for updating their hyperparameters by the Adaptive Moment Estimation (Adam) method [28]. After K epochs, the old actor $\pi_{\theta_{old}}$ and the old critic $V_{\omega_{old}}$ are updated to new π_θ and V_ω. After that, the memory buffer $\mathcal{M}$ is emptied for the next iteration.

B. Supervisor Integrated MAPPO

In the rule-based time-step environment, the *Departure* rule and *Holding* rule act as a *supervisor* for the relevant candidate flights. Different from these deterministic rules, the candidate flights in *Cooperation* need to decide their actions with the reinforcement learning method, where the joint action at time step t may cause a hotspot. Such joint action is not expected, and may cause a waste of time on searching the infeasible action space.

Inspired by the supervised reinforcement learning approach [29], a supervisor integrated MAPPO (S-MAPPO) is proposed, where a *supervisor* is designed for *Cooperation* to improve the scalability and the learning performance. In this paper, the *supervisor* aims to transfer the initial joint action $\boldsymbol{a}_t = \{a_t^i | i = 1, ..., N_t\}$ to the supervised one $\boldsymbol{sa}_t = \{sa_t^i | i = 1, ..., N_t\}$ by considering their effects on the future sectors.

The framework of the *supervisor* is shown in Fig. 4. At each time step, the *supervisor* gets the state value $V_\omega(s_t^i, \boldsymbol{s}_t^{-i})$ of each agent Ag_t^i, and rank all agents by the descend order of the state value. Then the action a_t^i of each agent is checked in sequence: if the action $a_t^i = 1$ causes a hotspot by considering the states of all agents, the action will be modified to *Holding*, denoted as $sa_t^i = 0$; otherwise, the supervised action remains constant as the initial action, i.e., $sa_t^i = s_t^i$. As a result, the supervised joint action is predicted to cause no hotspots.

The supervised action is applied from two aspects, a direct method (S-MAPPO-1) and an indirect method (S-MAPPO-2). Both algorithms are presented in Algorithm 2.

- *S-MAPPO-1*: In the direct method, the supervised action is performed instead of the initial action in the environment, i.e., $a_t^i \leftarrow sa_t^i$. Consequently, no hotspots exist, and the reward is only associated with the ground delay. Besides, the training becomes easier, due to a narrowed action space for searching compared with the action space in the baseline MAPPO.
- *S-MAPPO-2*: In the indirect method, the initial action is performed in the environment. During the training process, and a new surrogate objective function L_θ^{new} is designed [30]:

$$L_\theta^{new} = (1 - \delta)L_\theta^{RB} + (-\delta)L_{ce}(\pi_\theta, \boldsymbol{sa}_t) \tag{15}$$

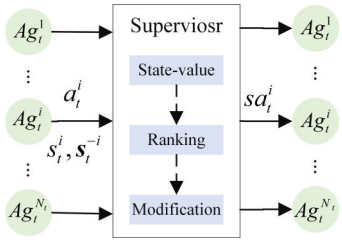


Fig. 4: Schematic of the role of Supervisor.

where $L_{ce}(\pi_\theta, \boldsymbol{sa}_t)$ is the cross entropy loss (shown as Eq. (IV-B)) of the supervised learning task, which aims to minimize the difference between the initial actions and the supervised actions; δ is the weight parameter to trade off the reinforcement learning and the supervised learning.

$$L_{ce}(\pi_\theta, \boldsymbol{sa}_t) = \hat{\mathbb{E}}[-sa_t^i \cdot \log(\pi_\theta) + (1 - sa_t^i) \cdot \log(1 - \pi_\theta)] \tag{16}$$

As a result, the actor is updated as below.

$$\Delta\theta = \bigtriangledown_\theta L_\theta^{new} \tag{17}$$

V. Experiments and Discussions

This section presents the experiments in which training and evaluation are performed using the proposed approach, with respect to a group of toy cases and a real-world large-scale case study. Results are analysed in terms of the search capability and generalization capability.

A. Experimental setup

Three categories of experimental settings are given below:

Hyperparamter: During training and evaluation, several hyperparameters are set as constant values, including the delay reward coefficient $\alpha_1 = -0.5$, the overload reward coefficient $\alpha_2 = -0.5$, the clipping parameter $\epsilon = 0.2$, the rollback coefficient $\beta = 0.3$, the initial greedy coefficient $\varepsilon = 0.1$, the weight $\delta = 0.5$, the reward discount $\gamma = 0.95$, and the size of minibatch $B = 20$. The common reward coefficient α_3 is set dynamically according to the buffer size N_b, i.e., $\alpha_3 = -\frac{1}{N_b}$. For the toy case with 300 flights, the maximum training iteration I_m is set as 5000 for all three methods. Besides, for the toy case with 3000 flights and the real-world case study, the baseline MAPPO and S-MAPPO-2 are performed with $I_m = 2000$, considering their large action space; and S-MAPPO-1 works with $I_m = 200$ due to a narrowed action space.

Performance: The performance is assessed from two aspects:

- *Search Capability*: the capability of the proposed methods to handle different DCB scenarios.
- *Generalization Capability*: the capability of a pre-trained policy to handle a new DCB scenario.

Evaluation Framework: To assess the above two performance indices, two different evaluation frameworks are proposed as shown in Fig. 5, where the solid line represents framework 1 (*EF1*), and the dashed line is framework 2 (*EF2*).

- In *EF1*, the new scenario is evaluated by the pre-trained policy with action $a_t^i = argmax(\pi_\theta(\cdot|s_t^i, \boldsymbol{s}_t^{-i}))$. The results are collected regardless of hotspots resolved or not. Considering the deterministic action performed, the statistic results, i.e., the average delay (*AveD*), the remaining overload (*LefO*), and the number of flights delayed (*NumFD*), are obtained after preforming *EF1* for only one time.
- In *EF2*, the action a_t^i is sampled based on the action probabilities $\pi_\theta(\cdot|s_t^i, \boldsymbol{s}_t^{-i})$. If the hotspot is not solved, or the iteration count is smaller than the threshold (10), the environment is updated with the solved scenario (i.e., the updated flight plans), and the evaluation is performed once again. Considering the stochastic actions in this framework, the average statistic results (i.e., *AveD*, *LefO*, and *NumFD*) are obtained after performing *EF2* by 10 runs for each scenario.

Algorithm 2: Supervisor Integrated MAPPO

```
Initialize common critic π_θ and common actor V_ω ;
Initialize the old critic π_{θ_old} ← π_θ, and the old critic
  V_{ω_old} ← V_ω ;
Initialize a memory buffer M;
for iteration I_c = 1, ..., I_m do
    Reset the environment;
    Get the initial state of each agent;
    for time step t = 1, ..., T in an episode do
        Get action {a_t^i}^{i=1,..,N_t} by Eq. (9) with the old
          actor π_{θ_old}(·|s_t^i, s_t^{-i});
        Get supervised action sa_t;
        if S-MAPPO-1 then
            a_t ← sa_t
        end
        Perform a_t;
        Get r_ins^{t,i}, s_{t+1}^i, s_{t+1}^{-i};
        Store (s_t^i, s_t^i, a_t^i, sa_t^i, r_ins^{t,i}, s_{t+1}^i, s_{t+1}^i) in M
    end
    Calculate the common reward r_com as Eq. (2);
    Store the common reward to M;
    Get advantage value {Â_t^i}_{t=1,...,T}^{i=1,...,N_t} by Eq. (10) ;
    Compute {y_t^i = Â_t^i + V_{ω_old}(s_t^i, s_t^{-i})}_{t=1,...,T}^{i=1,...,N_t};
    for k = 1, ..., K do
        for j = 1, ..., J do
            Sample a minibatch from the memory M;
            Get critic loss L_ω of minibatch by Eq. (13);
            if S-MAPPO-1 then
                Δθ = ▽_θ L_θ^{RB};
            else if S-MAPPO-2 then
                Δθ = ▽_θ L_θ^{new};
            Update θ using Δθ by Adam;
            Update ω using Δω = − ▽_ω L_ω by Adam;
        end
    end
    Update π_{θ_old} ← π_θ, and V_{ω_old} ← V_ω ;
    Empty the memory buffer M
end
```

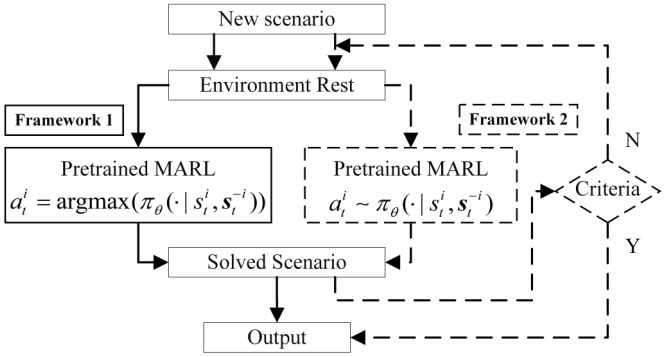


Fig. 5: Two evaluation frameworks: *EF1* and *EF2*.

TABLE I: Evaluation results on toy cases with 300 flights.

Case	Method	CASA	MAPPO		S-MAPPO-1		S-MAPPO-2	
			EF1	EF2	EF1	EF2	EF1	EF2
C-3-84	*AveD*	1.783	0.507	1.422	**0.52**	**0.52**	4.877	4.078
	LefO	0	18	5.5	**0**	**0**	15	13.2
	NumFD	255	92	137.9	**87**	**87**	79	81.2
C-5-8	*AveD*	0.128	0.04	0.043	**0.033**	**0.037**	0.037	0.037
	LefO	0	1	0	**0**	**0**	0	0
	NumFD	73	10	11.4	**8**	**8**	11	11
C-4-14	*AveD*	0.24	0.04	0.069	**0.06**	**0.06**	0.043	0.047
	LefO	0	7	1.3	**0**	**0**	6	5.9
	NumFD	100	10	16.3	**16**	**16**	9	9.5
C-4-27	*AveD*	0.473	0.137	0.186	**0.11**	**0.11**	0.197	0.197
	LefO	0	0	0.3	**0**	**0**	0	0
	NumFD	178	33	39.8	**27**	**27**	33	33

B. Toy Case

1) Case with 300 flights: This case generates flight plans within 16 sectors, during the 72 time periods. The time step (minute) size is also 72, and an identical capacity is shared by all sectors during the whole time periods.

Firstly, the baseline MAPPO, and two supervisor integrated MAPPO methods are pre-trained on the training case, where a set of initial flight plans FP_A is generated randomly with 27 initial overloads and an identical capacity 4. The overall return during the training process is shown in Fig. 6. The training results indicate that, all three MAPPO methods can converge to a stable overall return within a given number of training episodes. S-MAPPO-1 outperforms the baseline MAPPO and S-MAPPO-2 in terms of the final return.

Four evaluation cases are designed to test the performance of the three pre-trained policies. The first three evaluation cases are to test the *Generalization Capability*, while the last one is to access the *Search Capability*. Specifically,

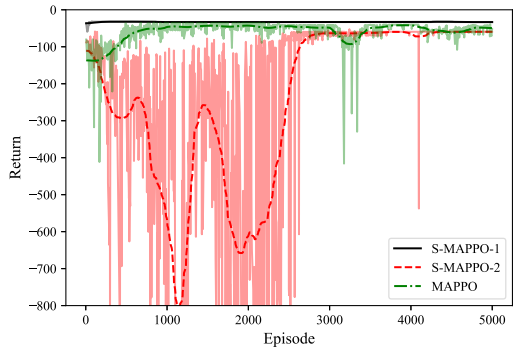


Fig. 6: Overall return during training on case *C-4-47*.

- *C-3-84*: 84 initial overloads exist in the initial flight plans FP_A with an identical capacity 3.
- *C-5-8*: 8 initial overloads occur in FP_A with capacity 5;
- *C-4-14*: A new set of initial flight plans FP_B is generated, leading to 14 initial overloads with capacity 4;
- *C-4-27*: The training case is applied.

The evaluation results of the pre-trained MARL policies are summarized in Table I. The results of CASA are also included for comparison. The bold results are the best one among the four methods. For all cases, S-MAPPO-1 obtains the best resolutions with a smaller average delay and a smaller number of flights delayed.

By comparing the results of the training case *C-4-27*, all methods successfully resolve the hotspots with both evaluation frameworks after training in most times, which indicates a reasonable *Search Capability*.

For the case *C-3-84* with *EF1*, the ratio (around 20%) of the remaining overload to the initial overload in the baseline MAPPO and S-MAPPO-2 is larger than that (about 10%) in case *C-5-8*. Also, both methods solve the new scenario *C-4-14*, remaining about 50% of the initial overload. These comparisons demonstrate that, the pre-trained MAPPO and S-MAPPO-2 have a better *Generalization Capability* in the same flight plans with a simpler DCB problem. However, S-MAPPO-1 can solve the former three evaluation cases with all hotspots eliminated and a smaller average delay. Thus, the supervisor integrated MAPPO performing the supervised action has the best *Generalization Capability* among all three MARL methods. Another important fact is that, all methods have a better generalization under *EF2* than *EF1* in most times, in terms of the average delay and the left overload.

Figs. 8.(a)-(d) summarize the ratio of demand to capacity (*D2R*) in each sector before and after running each method. In these figures, all ratio values are shown in the ascending order, and the value on x-axis indicates the relative location of each ratio among all ratio values. Comparing the initial *D2R* values, all methods can reduce the number of sectors whose $D2R > 1$, which proves the effectiveness in hotspot resolution.

2) Case with 3000 flights: A more complex toy case is designed to test the performance in large-scale DCB problem. In this case, 3000 flight plans are generated with 1440 time steps (minutes), 72 time periods, and a constant capacity 23.

A set of flight plans are generated with initial overload 177 for training, denoted as case *C-23-177*. The training curves shown in Fig. 7 indicate that S-MAPPO-1 still has the best training performance. Though the overall returns obtained by MAPPO and S-MAPPO-2 are both increasing, the fluctuation is larger than that of S-MAPPO-1. Besides, another evaluation case *C-23-153* is defined with a new set of flight plans causing 153 initial overloads.

The comparison results in Table II show that, S-MAPPO-

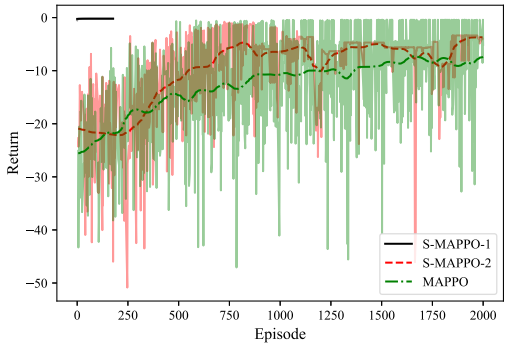


Fig. 7: Overall return during training on case *C-23-177*.

1 has the best evaluation results among the four methods, considering the remaining overload and the average delay. In *C-23-177*, the *Search Capability* is proved, as all methods can resolve the hotspots in most times (shown as 8 (e)), and have a better resolution with fewer flights delayed and a smaller average delay, both about 10% of those obtained by CASA. However, the pre-trained MAAPO and S-MAPPO-2 have a poor *Generalization Capability*, as they fail to solve *C-23-153* in both evaluation frameworks, as shown with the *D2R* values in Fig. 8 (f).

TABLE II: Evaluation results on toy case with 3000 flights.

Case	Method	CASA	MAPPO		S-MAPPO-1		S-MAPPO-2	
			EF1	EF2	EF1	EF2	EF1	EF2
C-23-177	*AveD*	13.514	1.273	1.280	**1.257**	**1.257**	1.423	1.423
	LefO	0	0	0.9	**0**	**0**	1	1
	NumFD	2122	212	214	**212**	**212**	213	213
C-23-153	*AveD*	5.708	1.129	1.250	**1.327**	**1.327**	1.147	1.223
	LefO	0	19	8.8	**0**	**0**	9	9.4
	NumFD	1877	199	212.8	**222**	**222**	208	213.2

To sum up, the comparison results in both toy cases indicate that all methods have promising *Search Capability* in different cases. S-MAPPO-1 has the best *Generalization Capability* in all tested cases, while MAPPO and S-MAPPO-2 only have an acceptable *Generalization Capability* in a simpler case with the same flight plans as used in training case.

C. Real-world Case Study

A real-world large-scale case study is performed based on the 24-hour traffic data on a typical day in February 2017, collected from the EUROCONTROL Demand and Data Repository v2 (DDR2) database. The scenario is focused on the French and Spain airspace, where 8153 flights are selected with 356 sectors in total. In this study, each sector is assumed to have a constant capacity during the 1440 time steps. The capacities are defined by narrowing the historical capacity in the database with coefficient $\psi \in [0, 1]$, due to the fact that the selected flights are a subset of all flights crossing these sectors.

Three different cases are designed by using different narrowing coefficients to generate different capacities.

- *C-74*: 74 initial overloads exists by using a medium coefficient, i.e., $\psi = 1/1.5$. This case is applied for both pre-training and evaluation;
- *C-35*: A simpler evaluation case with 35 initial overloads is designed by using a larger narrowing coefficient.
- *C-145*: With a smaller coefficient, less capacity is defined for each sector, resulting in a more complex evaluation case with 145 initial overload.

TABLE III: Evaluation results on real-world case study.

Case	Method	CASA	MAPPO		S-MAPPO-1		S-MAPPO-2	
			EF1	EF2	EF1	EF2	EF1	EF2
C-74	*AveD*	0.810	0.125	0.126	**0.124**	**0.124**	0.148	0.132
	LefO	0	0	0	**0**	**0**	0	0.02
	NumFD	473	129	129	**127**	**127**	138	132.4
C-35	*AveD*	0.188	0.036	0.038	**0.040**	**0.040**	0.040	0.040
	LefO	0	4	2.3	**0**	**0**	1	0.5
	NumFD	270	54	57.5	**57**	**57**	59	59.6
C-145	*AveD*	2.986	0.341	0.431	**0.450**	**0.450**	7.336	0.474
	LefO	0	16	14.9	**0**	**0**	5	2.3
	NumFD	917	221	242.6	**244**	**244**	490	243.5

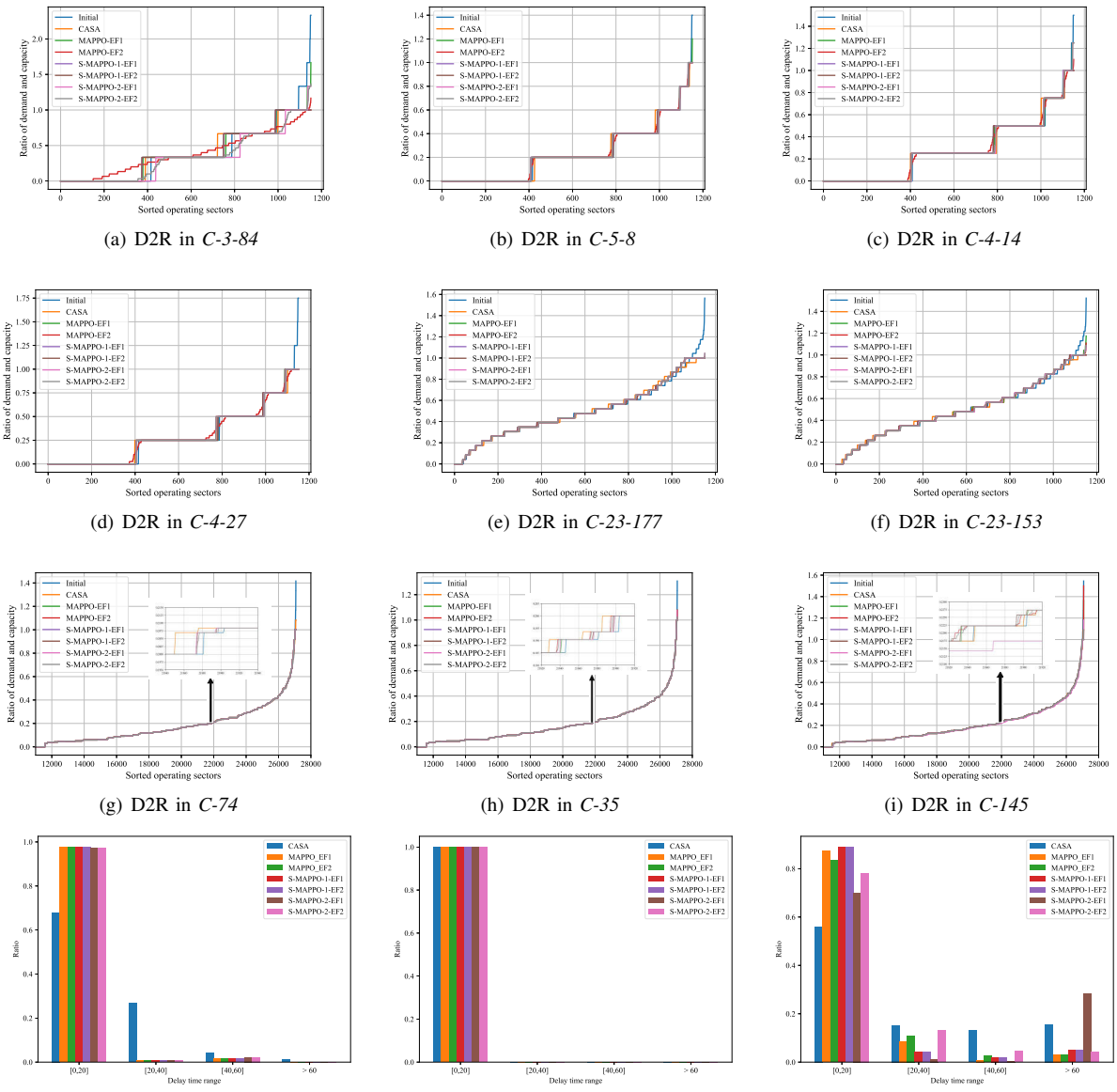
Fig. 9 shows the training process of overall return by three MARL methods. The evaluation results are compared in Table III. Compared with CASA, MPPO, S-MAPPO-1, and S-MAPPO-2 obtain better results in all cases, with a smaller average delay and a smaller number of flights delayed. Besides, the *Search Capacity* is proved in the training case, as MAPPO and S-MAPPO-2 successfully solve the case *C-74* under both evaluation frameworks, and S-MAPPO-1 can solve the same case with fewer average delay and number of flights delayed. Meanwhile, S-MAPPO-1 has the best *Generalization Capability* in the simpler case *C-35* and the more complex case *C-145*. Even though MAPPO and S-MAPPO-2 have better results in *EF2* in *C-35* and *C-145*, they fail to resolve all the hotspots under both evaluation frameworks.

In Fig. 8 (j)-(l), the ratios of flights with delays $0 \sim 20$ min, $20 \sim 40$ min, $40 \sim 60$ min, and over 60 min, to the size of all delayed flights are shown respectively for different methods. After running methods under different evaluation frameworks, over 60% of flights in *C-145* and *C-145*, and all flights in *C-35*, take only delays less than 20 min. For the most complex case *C-145*, around 20% of delayed flights have over 60 min delays in S-MAPPO-2 under *EF2*, larger than other MARL methods.

In conclusion, S-MAPPO-1 has a better performance in real-world large-scale case study, which is probably attributed to the supervised actions performed. MAPPO and S-MAPPO-2 have an acceptable *Search Capability*, while the *Generalization Capability* is poor, as they are unable to resolve the hotspots in new cases.

VI. Conclusion

In this study, we introduced MARL frameworks to solve the DCB problem with ground-delay strategy. The DCB problem is first simulated in a rule-based time-step environment, where three rules, *Holding*, *Departure*, and *Cooperation*, are designed to select a subset of flights to decide their actions cooperatively,



(a) D2R in *C-3-84*

(b) D2R in *C-5-8*

(c) D2R in *C-4-14*

(d) D2R in *C-4-27*

(e) D2R in *C-23-177*

(f) D2R in *C-23-153*

(g) D2R in *C-74*

(h) D2R in *C-35*

(i) D2R in *C-145*

(j) Delay in *C-74*

(k) Delay in *C-35*

(l) Delay in *C-145*

Fig. 8: Evaluation results for different cases, in terms of demand-capacity ratio before and after running the proposed MARL methods, along with the distribution of required flight delays.

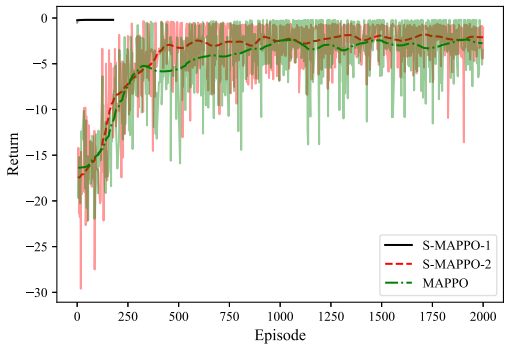


Fig. 9: Overall return during training on real-world case.

in order to minimize the final average delay and to solve the hotspots. Based on the PPO framework, the baseline MAPPO and supervisor integrated MAPPO methods (S-MAPP-1, S-MAPPO-2) are developed by parameter sharing, mean-filed regime, and supervised learning. The evaluation results in two toy cases and a large-scale real-world ATFM case demonstrate that, all proposed MARL methods have great *Search Capacity*, which means they can handle different DCB scenarios after being trained with a specific one. Besides, S-MAPPO-1 has a better *Generalization Capacity*, as the pre-trained policy can be applied directly to solve different cases. However, MAPPO and S-MAPPO-2 only have an acceptable *Generalization Capacity* in a simpler case.

Some further research are envisioned in our future work.

The sparse state will be reshaped to enable the pre-trained policy to be used directly in different DCB scenarios with different sector numbers. Rerouting and dynamic airspace sectorisation will be explored. More real-world large-scale DCB scenarios can be designed to justify the *Search Capacity* and the *Generalization Capacity* of the proposed approach. Some promising methods in MARL such as communication and credit assignment will be also investigated.

REFERENCES

[1] FAA, "Cost of delay estimates," Jul. 2020, accessed April 16, 2021. [Online]. Available: https://www.faa.gov/data_research/aviation_data_statistics/media/cost_delay_estimates.pdf

[2] C. Eurocontrol, "Coda digest all-causes delay and cancellations to air transport in europe annual report for 2019," 2019.

[3] Civil Aviation Administration of China, "Statistical Bulletin of Civil Aviation Industry Development in 2019," *China Civil Aviation Annual Report*, pp. 35–45, 2019.

[4] 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.

[5] A. R. Odoni, "The flow management problem in air traffic control," in *Flow control of congested networks*. Springer, 1987, pp. 269–288.

[6] A. Agustín, A. Alonso-Ayuso, L. F. Escudero, C. Pizarro *et al.*, "Mathematical optimization models for air traffic flow management: A review," 2010.

[7] D. Bertsimas, G. Lulli, and A. Odoni, "An integer optimization approach to large-scale air traffic flow management," *Operations research*, vol. 59, no. 1, pp. 211–227, 2011.

[8] Y. Xu, X. Prats, and D. Delahaye, "Synchronised demand-capacity balancing in collaborative air traffic flow management," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 359–376, 2020.

[9] K. Ng, C. K. Lee, F. T. Chan, and Y. Lv, "Review on meta-heuristics approaches for airside operation research," *Applied Soft Computing*, vol. 66, pp. 104–133, 2018.

[10] C. Taylor, T. Masek, and C. Wanke, "Designing traffic flow management strategies using multiobjective genetic algorithms," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 10, pp. 1922–1934, 2015.

[11] M. Sergeeva, "Automated airspace sectorization by genetic algorithm," Ph.D. dissertation, Université Paul Sabatier (Toulouse 3), 2017.

[12] M. Xiao, K. Cai, and H. A. Abbass, "Hybridized encoding for evolutionary multi-objective optimization of air traffic network flow: A case study on china," *Transportation Research Part E: Logistics and Transportation Review*, vol. 115, pp. 35–55, 2018.

[13] K. D. Bilimoria, B. Sridhar, S. R. Grabbe, G. B. Chatterji, and K. S. Sheth, "Facet: Future atm concepts evaluation tool," *Air Traffic Control Quarterly*, vol. 9, no. 1, pp. 1–20, 2001.

[14] K. Tumer and A. Agogino, "Distributed agent-based air traffic flow management," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007, pp. 1–8.

[15] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[16] 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.

[17] 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.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[19] Y. Wang, H. He, and X. Tan, "Truly proximal policy optimization," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 113–122.

[20] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[21] S. Omidshafiei, D.-K. Kim, M. Liu, G. Tesauro, M. Riemer, C. Amato, M. Campbell, and J. P. How, "Learning to teach in cooperative multiagent reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6128–6136.

[22] J. Yang, A. Nakhaei, D. Isele, K. Fujimura, and H. Zha, "Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning," *arXiv preprint arXiv:1809.05188*, 2018.

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

[24] 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.

[25] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5571–5580.

[26] J. Arabneydi and A. Mahajan, "Team optimal control of coupled subsystems with mean-field sharing," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1669–1674.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] D.-C. Juan and D. Marculescu, "Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, 2012, pp. 97–102.

[30] L. Wang, W. Zhang, X. He, and H. Zha, "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2447–2456.

AUTHOR BIOGRAPHIES

Mr Yifan Tang is currently a PhD candidate in Aerospace at Cranfield University. He received his M.Sc. and B.Eng. in Aeronautical & Astronautical Science & Technology from Beijing Institute of Technology. His main research interests include air traffic flow management, meta-heuristic optimization integrated with artificial intelligence. Email address: yifan.tang@cranfield.ac.uk

Dr Yan Xu is a Lecturer in ATM/CNS with the Centre for Autonomous and Cyber-Physical Systems in the School of Aerospace, Transport and Manufacturing at Cranfield University. He received his Ph.D. in Aerospace Science and Technology from the Technical University of Catalonia, and received his M.Sc. and B.Eng. in Traffic Engineering from Nanjing University of Aeronautics and Astronautics. His main research interests include air traffic flow and capacity management, ATM/UTM and Urban Air Mobility. Email address: yanxu@cranfield.ac.uk