Proceedings of the 2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC), 3-7 October, San Antonio, USA
DOI:10.1109/DASC52595.2021.9594343

Demand and Capacity Balancing Technology Based on Multi-agent Reinforcement Learning

Yutong Chen[a,b], Yan Xu[a], Minghua Hu[b], Lei Yang[b]
[a] School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, United Kingdom
[b] College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing, China
{yutong.chen, yanxu}@cranfield.ac.uk, {minghuahu, laneyang}@nuaa.edu.cn

***Abstract*—To effectively solve Demand and Capacity Balancing (DCB) in large-scale and high-density scenarios through the Ground Delay Program (GDP) in the pre-tactical stage, a sequential decision-making framework based on a time window is proposed. On this basis, the problem is transformed into Markov Decision Process (MDP) based on local observation, and then Multi-Agent Reinforcement Learning (MARL) method is adopted. Each flight is regarded as an independent agent to decide whether to implement GDP according to its local state observation. By designing the reward function in multiple combinations, a Mixed Competition and Cooperation (MCC) mode considering fairness is formed among agents. To improve the efficiency of MARL, we use the double Q-Learning Network (DQN), experience replay technology, adaptive ϵ-greedy strategy and Decentralized Training with Decentralized Execution (DTDE) framework. The experimental results show that the training process of the MARL method is convergent, efficient and stable. Compared with the Computer-Assisted Slot Allocation (CASA) method used in the actual operation, the number of flight delays and the average delay time is reduced by 33.7% and 36.7% respectively.**

***Index Terms*—demand and capacity balancing, ground delay program, multi-agent reinforcement learning, double Q-learning network, experience replay, adaptive ϵ-greedy strategy, decentralized training with decentralized execution**

I. Introduction

At present, the principal contradiction in the development of civil aviation in the world is the contradiction between the increasing total transport volume and the increasingly saturated airspace capacity, that is, the contradiction between demand and capacity. Therefore, how to balance the relationship between demand and capacity has gradually become a concern in civil aviation. In 2005, International Civil Aviation Organization (ICAO) listed Demand and Capacity Balancing (DCB) as one of the seven operational concepts of Air Traffic Management (ATM) [1].

At present, air traffic usually is based on route operation, usually in the form of traffic flow. Therefore, at present, the main implementation form of DCB is Air Traffic Flow Management (ATFM) or Air Traffic Flow and Capacity Management (ATFCM) [2], [3]. The operation means of DCB include ground delay program (GDP), rerouting, speed regulation and their combination [4], [5].The planning method is a commonly used method to solve the DCB problem, and its advantage is that it can get the optimal solution under certain conditions. A new integer programming model was used to find the optimal combination of ground waiting and rerouting strategies. An ATFM optimization model based on mixed-integer linear programming implemented in the pre tactical stage has proven effective in reducing delays [6]. However, for large-scale problems, the planning method is often very time-consuming, even can not get the solution in a limited time, so the more rapid and efficient heuristic algorithm was also widely used in the study of DCB problem [7], although it can not guarantee the optimal solution. An improved multi-objective genetic algorithm was designed to generate the ATFM strategy to get an excellent combination of strategy [8]. A hybrid genetic algorithm with indirect coding and direct coding has proven more effective in dealing with the ATFM problem [9]. An approximate algorithm of stage solution was applied to solve the multi-objective optimization model for large-scale ATFM problems under emergencies, which has more advantages than the optimization algorithm [10]. Generally speaking, for the traditional ATFM problem, the solution steps can be summarized as follows: first, establish the optimization model according to the problem, and try best to get the optimal solution through the optimization method; second, if the optimization method can not get the optimal solution quickly, then consider the heuristic method to get the acceptable suboptimal solution in a limited time.

With the rise and rapid development of Artificial Intelligence (AI), the traditional framework of "modelling and solving" is no longer the only way to solve the DCB problem. A problem-solving model based on "learning" is gradually showing its potential in solving the DCB problem. Some scholars have tried to transform the DCB problem into a Markov decision process (MDP) through some clever ways and then use Multi-Agent Reinforcement Learning (MARL) method to solve it. For example, in the air traffic simulator FACET [11], some specific location points in the two-dimensional space were taken as agents, training which to decide the safety separation between the passing aircraft [12], or each aircraft was used as an agent to train itself to allocate an appropriate delay based on GDP. In this mode, scholars have explored many MARL frameworks, such as edge-based, agent-based, hierarchical MARL framework [13], [14].

Although there have been some attempts to apply the MARL technology to the DCB problem, no one has proposed a

This work is funded by the Postgraduate Research Innovation Program of Jiangsu (KYCX20_0213) and the Interdisciplinary Innovation Foundation for Postgraduates of NUAA (KXKCXJJ202001).

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

complete technical system. Especially in the framework of MARL, agents are either independent egoists or fully cooperative collectivists. They have not considered the current actual ATM operation mode that is the centralized management mode with limited fairness. Therefore, this paper proposes a DCB technology based on MARL that considers fairness. The main work of this paper is as follows:

- For the DCB problem based on GPD in the pre tactical stage, a sequential decision framework based on time window is proposed. By this framework, the DCB problem is transformed into MDP based on local observation so that the MARL method can be used;
- Each flight is regarded as an independent agent to determine the next action, i.e. take-off or holding, through local observation, i.e. the demand and capacity of the sector it passes through and the number of times it currently holds;
- The agent's action reward consists of three parts: local reward, social reward and global reward. They are used to feedback the holding times of the flight itself, the overload of sector capacity and the holding times of all flights. The social reward acts as the constraint of sector capacity, while the agent achieves limited fairness between flights by balancing local reward and global reward. Thus, a Mixed Competition and Cooperation (MCC) model considering fairness is formed among agents;
- Several Reinforcement Learning (RL) technologies are applied to improve learning efficiency, including double Deep Q-learning Network (DQN), experience replay, adaptive ϵ-greedy strategy and Decentralized Training with Decentralized Execution (DTDE) framework. The aim of using double DQN is to reduce the impact of overestimation caused by bootstrapping. The experience replay is an efficient method to improve the utilization rate of experience and break the correlation between experiences. An adaptive ϵ-greedy strategy is designed to balance the relationship between exploration and utilization. The MARL framework of DTDE is adopted to distinguish the characteristics of different flights.

II. Problem Description

In the pre-tactical phase, we can obtain the departure time of each flight and the time of each flight entering each sector according to the flight plan. For a time window of a sector, if the number of flights entering the sector is more than the sector capacity, the sector is recorded as a Hot Spot in the time window. The problem to be solved in this paper is that for large-scale and high-density flight operation scenarios, by implementing GDP on flights, the flight departure time and the time entering each sector are moved back synchronously to eliminate the hot spots and reduce flight delays as much as possible.

III. Demand and Capacity Balancing Sequence Decision Framework Based on Time Window

According to the initial flight plan of a day $\mathcal{P}^{initial}$, the flight set $f \in \mathcal{F}$, sector set $e \in \mathcal{E}$ and time window set $t \in \mathcal{T}$ are given. If a day is divided into n time windows, the length of each time window is $1440/n$ minutes. For flight f, its initial flight plan can be represented by an N_f dimensional tuple, that is,

$$p^f_{initial} = \left\{ \left(t^f_1, e^f_1\right), \left(t^f_2, e^f_2\right), \cdots, \left(t^f_{N_f}, e^f_{N_f}\right) \right\} \quad (1)$$

where $p^f_{initial} \in \mathcal{P}^{initial}$, N_f is the number of sectors that flight f passes through, $e^f_{N_f}$ is the N_fth sector of flight f, $t^f_{N_f}$ is the time window where flight f passes through the N_fth sector, tuple $\left(t^f_{N_f}, e^f_{N_f}\right)$ indicates that flight f enters sector $e^f_{N_f}$ in time window $t^f_{N_f}$ and at this time, the departure time window of flight f is expressed as $t^f_{dep} = t^f_1$. In the process of implementing GPD, we assume that a holding time of a flight is a time window width. For example, when a holding is imposed on flight f, its flight plan will change to:

$$p_f = \left\{ \left(t^f_1 + 1, e^f_1\right), \left(t^f_2 + 1, e^f_2\right), \cdots, \left(t^f_{N_f} + 1, e^f_{N_f}\right) \right\} \quad (2)$$

where $t+1$ represents the next time window of time window t and at this time, the departure time window of flight f is expressed as $t^f_{dep} = t^f_1 + 1$. The capacity of sector e in time window t is expressed as $c_{t,e}$. Use $k^f_{t,e}$ to indicate whether flight f enters sector e within the time window t. If yes, take 1; otherwise, take 0. If $\sum_f k^f_{t,e} \geq c_{t,e}$, then sector e is a hot spot in time window t. $\sigma_{t,e}$ is used to represent the overload of sector e in time window t. the overload refers to the value of sector demand exceeding capacity in a certain time window, that is,

$$\sigma_{t,e} = R_{eLU}\left(\sum_f k^f_{t,e} - c_{t,e}\right) \quad (3)$$

where R_{eLU} is the Rectified Linear Unit (ReLU), that is,

$$R_{eLU}(x) = \max(x, 0) \quad (4)$$

Therefore, (3) shows that when the demand exceeds the capacity, the overload takes the difference between them, if not, takes 0. Obviously, with other conditions unchanged, the smaller the time window width is, the more strict the matching requirements between capacity and demand are, and the more accurate the optimization is.

Based on the above environment, we build a sequential decision framework based on time window (see Fig. 1).

In this framework, firstly, according to the initial flight plan $p^f \in \mathcal{P}^{initial}$, all flights $f \in \mathcal{F}$ are allocated to the corresponding decision set $\mathcal{D}_t$ according to the time window t^f_{dep} of their initial departure time, that is,

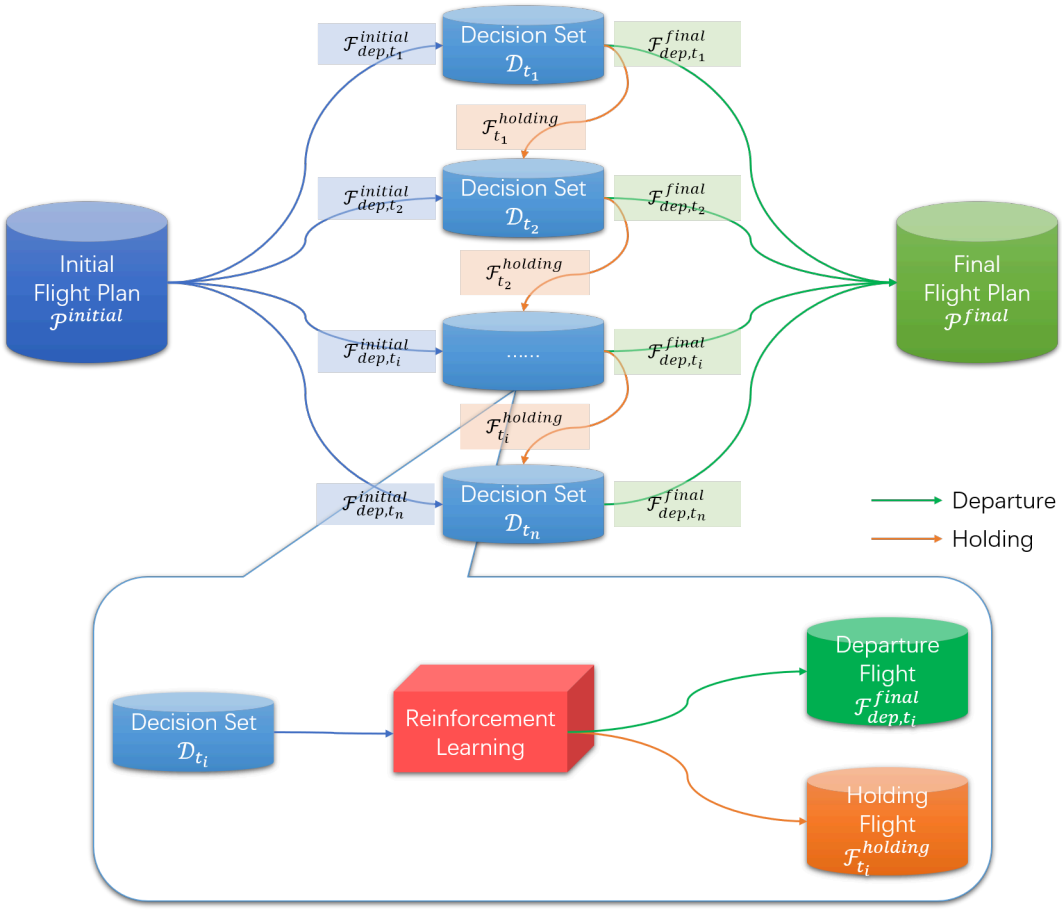


Fig. 1. DCB sequence decision framework based on time window.

$$\mathcal{D}_t = \mathcal{F}^{initial}_{dep,t} = \left\{ f | \, t^f_{dep} = t \right\} \tag{5}$$

where $\mathcal{F}^{initial}_{dep,t}$ is the set of all flights whose initial departure time window is t. Then, according to the order of the time window of the decision set, the flights in each decision set make decisions in turn, that is, take-off or holding. Taking decision set $\mathcal{D}_t$ as an example, for all flights $f \in \mathcal{F}^{initial}_{dep,t}$ in decision set $\mathcal{D}_t$, RL algorithm is used to decide whether to take off or hold. Among them, the flights that decide to take off constitute the departure flight $f \in \mathcal{F}^{final}_{dep,t}$, and its current flight plan p^f is added to the final flight plan $\mathcal{P}^{final}$. $\mathcal{P}^{final}_t$ is used to represent the set of final flight plans that have been generated after all flights in decision set $\mathcal{D}_t$ have made the decision and $\mathcal{F}^{departure}_t$ is the set of all flights contained in $\mathcal{P}^{final}_t$, that is,

$$\mathcal{F}^{departure}_t = \bigcup_{i=1,2,\ldots,t} \mathcal{F}^{final}_{dep,i} \tag{6}$$

The flight that decides to hold is composed of the holding flight $\mathcal{F}^{holding}_t$, which is added to the subsequent decision set $\mathcal{D}_{t+1}$. Therefore, before making a decision on a flight in the decision set $\mathcal{D}_t$, if the flight delayed from the previous day is not taken into account, we can get:

$$\mathcal{D}_t = \begin{cases} \mathcal{F}^{initial}_{dep,t} & t = 1 \\ \mathcal{F}^{initial}_{dep,t} \bigcup \mathcal{F}^{holding}_t & t > 1 \end{cases} \tag{7}$$

Complete the flight decisions in the decision set in turn until all flights $f \in \mathcal{F}$ have generated the final flight plan, that is, $\forall p^f \in \mathcal{P}^{final}$, and the framework is completed. The detailed framework flow is shown in Algorithm 1.

IV. Markov Decision Process Based on Local Observation

In the above sequential decision framework based on time window, we can find that the decision of flights in decision set $\mathcal{D}_t$ is only related to the departure flight in the previous decision set of decision set $\mathcal{D}_t$, that is, $\mathcal{F}^{departure}_{t-1}$, and has nothing to do with the flights in decision set $\mathcal{D}_t$ and its subsequent decision set.In addition, the scale of DCB problem is often very large. Assuming that there are 3000 flights, 16 sectors and 72 time windows, if the 0-1 sparse matrix is used to represent the global information, that is, flight plans of all flights at present, the dimension of the matrix will be $3000 \times 16 \times 72$. It can be seen that the agent state matrix under global observation will be very large, and it is difficult for the agent to learn effective information efficiently. Therefore, we transform the DCB problem into an MDP based on local observations. The

Algorithm 1 Framework execution process

Input: $f \in \mathcal{F}$, $t \in \mathcal{T}$, $p^f \in \mathcal{P}^{initial}$

$\mathcal{P}^{final} \leftarrow \varnothing$
for t from 1 to n **do**
  $\mathcal{D}_t \leftarrow \varnothing$
  for f in $\mathcal{F}$ **do**
    if $t_{dep}^f = t$ **then**
      add f to $\mathcal{D}_t$
    end if
  end for
end for
for t from 1 to n **do**
  for f in $\mathcal{D}_t$ **do**
    flight f decides to take off or hold by RL
    if flight f decided to take off **then**
      add p^f to $\mathcal{P}^{final}$
    else
      $t_{dep}^f \leftarrow t_{dep}^f + 1$
      update p^f
      add f to $\mathcal{D}_{t+1}$
    end if
  end for
end for

MDP is expressed as $\boldsymbol{G} = (N, \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \gamma)$, where N is the number of agents, $\boldsymbol{S}$ is the set of States, $\boldsymbol{A}$ is the action space, $\boldsymbol{P}$ is the state transition probability, $\boldsymbol{R}$ is the reward function, and γ is the discount rate of reward. The specific definition of the above elements is as follows.The specific definition of the above elements is as follows:

A. Agent

Each flight is regarded as an independent agent, denoted as Ag^f. It should be noted that in this RL environment, only the agent representing the flight in the current decision set will be activated. Therefore, the set of activated agents in decision set $\mathcal{D}_t$ can be expressed as $\boldsymbol{Ag}_t = \left\{ Ag^f \middle| f \in \mathcal{D}_t \right\}$. Each agent is regarded as heterogeneous, in order to learn its own unique environment, and then make the most suitable decision.

B. State

If agent Ag^f is activated in decision set $\mathcal{D}_t$, the state of agent at this time is expressed as s_t^f. s_t^f is a $3 \times N_f$ dimensional matrix and can be expressed as:

$$s_t^f = \begin{bmatrix} c^{remaining}_{t^f_1, e^f_1} & \cdots & c^{remaining}_{t^f_{N_f}, e^f_{N_f}} \\ d^{potential}_{t^f_1, e^f_1} & \cdots & d^{potential}_{t^f_{N_f}, e^f_{N_f}} \\ h_t^f & \cdots & h_t^f \end{bmatrix} \tag{8}$$

Firstly, let us read the matrix horizontally. Each element in the first row represents the current remaining capacity of flight f in each sector it passes through, that is, $c_{t,e}^{remaining}$ represents the remaining capacity of current sector e in time window t. The remaining capacity depends on the capacity of each sector and the flight plan $p^f \in \mathcal{P}_t^{final}$ of the departure flight $f \in \mathcal{F}_{t-1}^{departure}$, that is,

$$c_{t,e}^{remaining} = c_{t,e} - \sum_f k_{t,e}^f, f \in \mathcal{F}_{t-1}^{departure} \tag{9}$$

Each element in the second row represents the current potential demand of flight f in each sector it passes through, that is, $d_{t,e}^{potential}$ represents the potential demand of current sector e in time window t. This potential demand depends on the current flight plan of the flights in decision set $\mathcal{D}_t$, that is, assuming that all the flights in decision set $\mathcal{D}_t$ choose to take off in time window t, the number of new demands will be added to these sectors. It can be expressed as:

$$d_{t,e}^{potential} = \sum_f k_{t,e}^f, f \in \mathcal{D}_t \tag{10}$$

Each element in the third row is the same, which is used to represent the number of times flight f has been waiting in the decision set $\mathcal{D}_t$, so that the agent can learn the time of the specific environment, and then make a specific action in a specific time and environment.

Secondly, let us read the matrix vertically. The agent is expected to make appropriate actions by learning the relationship between the capacity and demand of each sector and its absolute quantity.

C. Action

When agent Ag^f is activated in decision set $\mathcal{D}_t$, there are two actions to choose, namely $a_t^f \in \{0, 1\}$. When $a_t^f = 0$, it means to hold; When $a_t^f = 1$, it means to choose to take off. It can be seen from the environment that in a learning round, when agent Ag^f chooses to take off, the agent will not appear in the subsequent decision set, that is, it will not be activated again. When agent Ag^f chooses to hold, it will continue to join the subsequent decision set and be activated until it chooses to take off.

D. Reward

For the action a_t^f of agent Ag^f, the reward function consists of three parts: local reward, social reward and global reward.

The local reward $r_{local,t}^f$ will be obtained after it completes the action, which is used to feedback the number of times a single agent holds. It can be expressed as:

$$r_{local,t}^f = \begin{cases} 0 & a_t^f = 1 \\ -\frac{1}{\lambda} \times h_t^f & a_t^f = 0 \end{cases} \tag{11}$$

where λ is the cooperation coefficient which is used to consider the fairness between flights. When the agent chooses to take off, the reward is 0. When the agent chooses to hold, it gets a negative feedback related to the number of times it has been holding. The larger λ is, the more likely a single flight is to increase its delay to achieve a greater reduction of overall delay, and vice versa. The purpose of local reward is to make agent take off as early as possible on the basis of cooperation and fairness, so as to reduce individual delay.

The social reward $r^f_{social,t}$ will be obtained after all the agents in the decision set $\mathcal{D}_t$ have finished their actions. It is used to feedback the overload of the sector that the agent passes through. The so-called society refers to that the flights entering the same sector in the same time window form a society. For example, all the flights entering sector e in time window t form a society about sector e in time window t. Obviously, if flight f passes through multiple sectors, it will exist in multiple societies respectively, and its social reward will be determined by the total number of overloads in each society. It can be expressed as:

$$r^f_{social,t} = \begin{cases} -M \times \sigma^f_t & a^f_t = 1 \\ 0 & a^f_t = 0 \end{cases} \tag{12}$$

where M is a very large constant, and σ^f_t is the sum of all sector overloads of the flight plan $p_f \in \mathcal{P}^{final}_t$ of all departure flights $f \in \mathcal{F}^{departure}_t$ after all flights in the decision set $\mathcal{D}_t$ complete the decision, if the flight f chooses to take off in the time window t. σ^f_t can be expressed as:

$$\begin{aligned} \sigma^f_t &= \sum_{(t',e)\in p^f} \sigma_{t',e} \\ &= \sum_{(t',e)\in p^f} R_{eLU}\left(\sum_{f'\in\mathcal{F}^{departure}_t} k^{f'}_{t',e} - c_{t',e}\right) \end{aligned} \tag{13}$$

(13) is also can be written as:

$$\begin{aligned} \sigma^f_t &= \sum_{(t',e)\in p^f} \sigma_{t',e} \\ &= \sum_{(t',e)\in p^f} R_{eLU}\left(\sum_{f'\in\mathcal{D}_t} k^{f'}_{t',e} a^{f'}_t - c^{remaining}_{t',e}\right) \end{aligned} \tag{14}$$

When agent Ag^f chooses to take off in decision set $\mathcal{D}_t$, if the sector it passes through is overloaded, it will get a very large negative feedback. In fact, limiting overload is the constraint of the DCB problem, and imposing a very large negative feedback on it is a common method in RL. If there is no overload or it chooses to hold, the reward is 0.

The global reward $r^f_{global,t}$ will be obtained after all agents have completed the take-off action, and it is used to feedback the total number of holding times of all agents. It can be expressed as:

$$r^f_{global,t} = \sum_{f\in\mathcal{F}} h^f_{final} \tag{15}$$

where h^f_{final} is the final holding times of flight f.

To sum up, for the action a^f_t of agent Ag^f, the final reward r^f_t is:

$$r^f_t = r^f_{local,t} + r^f_{social,t} + r^f_{global,t} \tag{16}$$

Since the global reward $r^f_{global,t}$ is only obtained after all agents have completed the take-off action, the final reward of action a^f_t of agent Ag^f is also obtained at this time. In reward r^f_t, agents achieve limited fairness between flights by balancing local reward $r^f_{local,t}$ and global reward $r^f_{global,t}$, thus forming an MCC mode considering fairness between agents. It should be noted that when agent Ag^f chooses to take off in decision set $\mathcal{D}_t$, agent Ag^f will not be activated in the decision set after decision set $\mathcal{D}_t$. To enable agent to effectively identify the state of taking off, we set state s^f_{t+1} artificially, that is,

$$s^f_{t+1} = \begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{bmatrix} \tag{17}$$

where the dimension of s^f_{t+1} is also $3 \times N_f$.

The process of obtaining reward r^f_t is shown in Algorithm 2.

Algorithm 2 Getting rewards

$A^f \leftarrow \varnothing$
for t from 1 to n **do**
  for Ag^f in $\boldsymbol{Ag}_t$ **do**
    if A^f is activated for the first time **then**
      observe the current state of Ag^f and get s^f_t
    end if
    decide a^f_t takes 1 or 0 by RL
    add a^f_t to A^f
    get $r^f_{local,t}$ by calculation
  end for
  for Ag^f in $\boldsymbol{Ag}_t$ **do**
    observe the current state of Ag^f and get s^f_{t+1}
    get $r^f_{social,t}$ by calculation
  end for
end for
for a^f_t in A^f **do**
  get $r^f_{global,t}$ by calculation
  $r^f_t \leftarrow r^f_{local,t} + r^f_{social,t} + r^f_{global,t}$
  generate the experience $\left(s^f_t, a^f_t, r^f_t, s^f_{t+1}\right)$
end for

V. Reinforcement Learning

A. Deep Q-learning Network

DQN is a reinforcement learning algorithm based on action value function learning [15], that is, the action value network $Q(a, s; \mathbf{w})$ learned by Neural Network (NN) is used to approximate the optimal action value function $Q^*(s, a)$ and the core learning method of DQN is Temporal Difference (TD) learning. Among them, $\mathbf{w}$ is a neural network, and its structure is shown in Fig. 2. The NN consists of three layers: input layer, hidden layer and output layer. There are $3 \times N_f$ neurons in the input layer, and each neuron represents an element of the observed state s^f_t of agent Ag^f in the decision set $\mathcal{D}_t$; There are N_{hidden} neurons in the hidden layer, and N_{hidden} is a hyper-parameter; There are two neurons in the output layer, which represent the actions selected by the agent, namely take-off and holding.

Bootstrapping is a common problem in DQN, that is, one estimation model is used to update the same estimation model, which leads to overestimation. In order to eliminate this

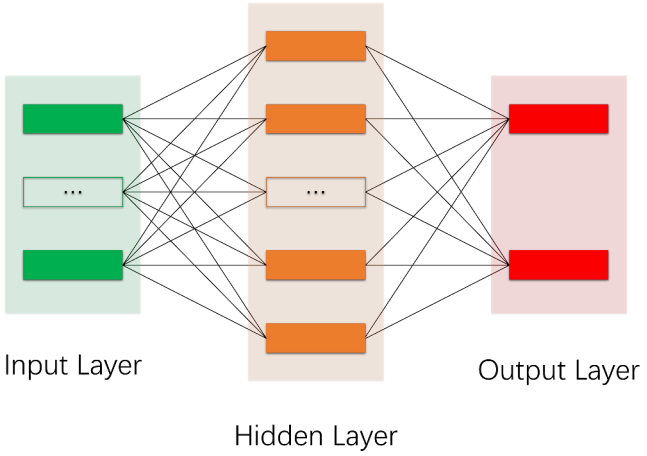


Fig. 2. Schematic diagram of neural network structure.

negative effect as much as possible, this paper adopts double DQN technology [16], which trains two NNs with the same structure but different parameters. One is used to predict the action value, and the other is used to calculate the TD target value. In the DCB problem of this paper, the parameter update of agent Ag^f can be divided into the following steps in an iteration.

- Step 1: observe state s_t^f and choose action a_t^f.
- Step 2: predict the action value q_t^f by NN, that is,

$$q_t^f = Q\left(s_t^f, a_t^f; \mathbf{w}_t^f\right) \tag{18}$$

where $\mathbf{w}_t^f$ is the NN to predict action value.

- Step 3: derive the NN $\mathbf{w}_t^f$ by back propagation to obtain its gradient $\mathbf{d}_t^f$, that is,

$$\mathbf{d}_t^f = \frac{\partial Q\left(s_t^f, a_t^f; \mathbf{w}^f\right)}{\partial \mathbf{w}^f}\Big|\mathbf{w}^f = \mathbf{w}_t^f \tag{19}$$

- Step 4: get the reward r_t^f and the new state s_{t+1}^f, and obtain the TD target value y_t^f, that is,

$$y_t^f = r_t^f + \gamma \cdot Q\left(s_{t+1}^f, a^*; \mathbf{w}_t^{f\text{-}}\right) \tag{20}$$

where γ is the discount rate of reward, and $\mathbf{w}_t^{f\text{-}}$ is an NN with the same structure as $\mathbf{w}_t^f$, which is used to calculate the TD target value.

- Step 5: use Mean Square Error (MSE) to express the loss function l_t^f, that is,

$$l_t^f = \frac{1}{2}\left(q_t^f - y_t^f\right)^2 \tag{21}$$

Then, update the NN $\mathbf{w}_t^f$ by gradient descent, that is,

$$\mathbf{w}_{t+1}^f = \mathbf{w}_t^f - \alpha \cdot \frac{\partial l_t^f}{\partial \mathbf{w}_t^f} \tag{22}$$

(22) can also be written as:

$$\mathbf{w}_{t+1}^f = \mathbf{w}_t^f - \alpha \cdot \left(q_t^f - y_t^f\right)\mathbf{d}_t^f \tag{23}$$

where α is the learning rate.

- Step 6: update the NN $\mathbf{w}_t^{f\text{-}}$ by assigning the parameters of $\mathbf{w}_t^f$ to $\mathbf{w}_t^{f\text{-}}$, that is,

$$w_t^{f\text{-}} \leftarrow w_t^f \tag{24}$$

It should be noted that the NN $\mathbf{w}_t^f$ used to predict the action value is updated every iteration, while the NN $\mathbf{w}_t^{f\text{-}}$ used to calculate the TD target value is updated every certain number of iterations. This number of iterations is a hyper-parameter, which can be expressed by N_{TD}.

In order to reuse, avoid the waste of experience, break up the learning sequence and eliminate the correlation of experience, this paper uses the experience replay technology in the DQN [17]. In RL, a four-dimensional tuple (s_t, a_t, r_t, s_{t+1}) is called an experience. Experience replay is to store a certain amount of experience in the Replay Buffer, and randomly extract a certain Batch Size of experience from the replay buffer for learning each time. Let us use N_{buffer} and N_{batch} to represent replay buffer size and batch size respectively, both of which are hyper-parameters.

B. Adaptive ϵ-greedy Strategy

In the aspect of exploration and exploitation, in order to balance the relationship between them and improve learning efficiency, this paper adopts adaptive ϵ-greedy strategy. The action selection model of agent Ag^f is:

$$a_t^f = \begin{cases} \arg\max\limits_a Q\left(s_t^f, a; w_t^f\right) & \mu \le \varepsilon^{adaptive} \\ random\left(A_t^f\right) & otherwise \end{cases} \tag{25}$$

Among (25), $random\left(A_t^f\right)$ means to randomly select any possible action of the agent Ag^f in the decision set $\mathcal{D}_t$, that is, to take off or to hold; μ is a random number between 0 and 1, that is, $\mu \in [0,1]$; $\varepsilon^{adaptive}$ is the adaptive greedy rate, that is,

$$\varepsilon^{adaptive} = \begin{cases} \varepsilon_0 + (1-\varepsilon_0)\left(1 - \frac{\sigma_{\min}^{global}}{\sigma_0^{global}}\right) & \sigma_t^{global} < \sigma_0^{global} \\ \varepsilon_0 & otherwise \end{cases} \tag{26}$$

Among (26), ε_0 is the benchmark greedy rate, which is a hyper-parameter and $\varepsilon_0 \in [0,1)$; σ_0^{global} is the initial global overload, that is, the total overload generated by all flights flying according to the initial flight plan $\mathcal{P}^{initial}$, and it can be expressed as:

$$\sigma_0^{global} = \sum_{t\in\mathcal{T}, e\in\mathcal{E}} \sigma_{t,e}\left|p^f \in \mathcal{P}^{initial}, f \in \mathcal{F}\right. \tag{27}$$

$\sigma_{\min}^{global}$ is the minimum overload of the existing solutions, and σ_r^{global} is used to represent the total overload in the r-th round, that is,

$$\sigma_r^{global} = \sum_{t\in\mathcal{T}, e\in\mathcal{E}} \sigma_{t,e}\left|p^f \in \mathcal{P}^{final}, f \in \mathcal{F}\right. \tag{28}$$

$$\sigma_{\min}^{global} = \min \sigma_r^{global}, r \in R_{done} \tag{29}$$

where R_{done} is the set of rounds that have been learned at present.

C. Multi-Agent Reinforcement Learning Framework

In the DCB problem, each agent is heterogeneous due to the differences in the aircraft type, flight plan and airspace environment. In addition, each agent can make action selection independently according to its local observation. Therefore, this paper adopts the MARL Framework of DTDE (see Fig. 3), and each agent does not share parameters.

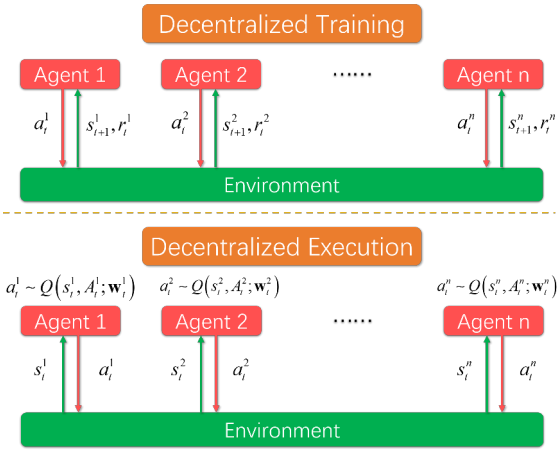


Fig. 3. Schematic diagram of decentralized training with decentralized execution.

In the framework of DTDE, all agents are independent individuals. There is no communication between agents. Each agent interacts with the environment independently. In decentralized training, each agent Ag^f independently obtains the observed state s^f and reward r^f from the environment, and each agent Ag^f has its independent action value network $\mathbf{w}^f$. They are trained independently and do not share parameters. Since the double DQN technology is used in the RL, each agent Ag^f has two independent NNs, namely $\mathbf{w}^f$ and $\mathbf{w}^{f\text{-}}$, so there are $2N$ NNs in the whole framework. At the end of training, each agent Ag^f uses its action value network $\mathbf{w}^f$ to make decisions independently, that is, decentralized execution. The observed state s^f is input into the action value network $\mathbf{w}^f$ which calculates the value of each action. Action a^f is made by the action selection model based on adaptive ϵ-greedy strategy, and then action a^f is executed.

VI. Experiments and Analyses

A. Scenario construction

This paper constructs an abstract airspace scenario to simplify the problem and facilitate the discussion (see Fig. 4). The airspace is composed of 16 grids, each of which represents a sector. Number each sector in order by 1-16. We divide a day into 72-time windows, and then the width of each time window is 20 minutes. Three thousand flight plans in a day generate randomly. Each flight includes the departure time of the flight and the sector it passes through. For example, as shown by the green route with arrow in Fig. 4, the flight successively passes through sectors 8, 7, 11, 15 and 14. Suppose that the average flight time of a flight in each sector is 20 minutes, and the capacity of a sector in each time window is 20 sorties, that is, 20 flights/ 20 minutes.

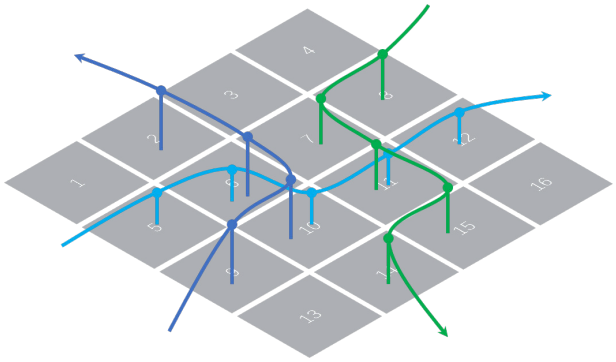


Fig. 4. Schematic diagram of experimental scenario.

B. Parameters setting

In Python environment, the double DQN is constructed based on the Pytorch framework, and its network parameters are updated by Adam optimization algorithm. For experience replay, the replay buffer size N_{buffer} is 200 and the batch size N_{batch} is 20. The interval number of iteration N_{TD}, which is used to decide when to update the NN $\mathbf{w}^{f\text{-}}$, is 10. The benchmark greedy rate in the adaptive ϵ-greedy strategy ε_0 is 0.9. The discount rate of reward γ is 0.9 and the learning rate α is 0.9. The empirical values of the parameters required in the training process are shown in Table I.

TABLE I
Parameter Value Table

Parameters	Empirical Values	Parameters	Empirical Values
N_{buffer}	200	γ	0.9
N_{batch}	20	α	0.9
N_{TD}	10	λ	0.5
ε_0	0.9	M	10000

C. Analysis of training results

In this experiment, we set the number of training rounds to 1000 and evaluate the strategy with average reward. Average reward refers to the quotient of the sum of reward and the number of experience in a complete training round. The average reward can be expressed as:

$$r_i^{average} = \frac{\sum_{r \in R_i^{experience}} r}{n_i^{experience}} \tag{30}$$

where $r_i^{average}$ is the average reward of the i-th round, $R_i^{experience}$ is the set of reward of all experiences in the i-th round, r is the element in $R_i^{experience}$, and $n_i^{experience}$ is the number of experiences in the i-th round.

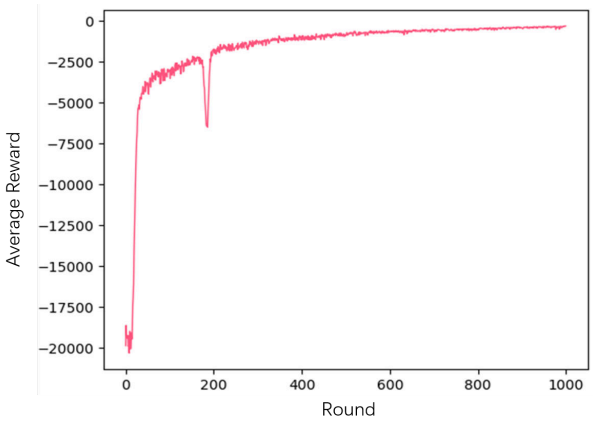


Fig. 5. Average reward during training.

From the process of training (see Fig. 5), the average reward of the first 50 rounds increased rapidly, then increased steadily, and the growth rate gradually slowed down, and basically stabilized around 800 rounds. It should be noted that there is a shortfall near the 200th round, which is due to the experience replay, and the replay buffer size N_{buffer} is 200. In detail, while in the 200th round, the experience replay begins to work. The experience in the replay buffer is randomly selected from the first 200th round for learning. However, the average rewards of the first 50 rounds in the replay buffer is small. That is the reason for the shortfall.

D. Method comparison

To verify the effectiveness of the proposed MARL method, we compare it with the Computer-Assisted Slot Allocation (CASA). CASA is a commonly used DCB method in the actual operation of Europe, which realizes the DCB by identifying sectors with hot spots and implementing GDP for flights passing through hot spots [18]. CASA transforms complex large-scale optimization problems into scheduling problems based on time priority and obtains feasible solutions through repeated iterations. The CASA framework is simple and easy to implement in practical operation. In the constructed experimental scenario, we generate 3000 flight plans randomly, and the results are optimized by the method of MARL and CASA (see Fig. 6).

The number of flight delays refers to the total number of flights that did not take off on time according to the initial flight plan, and the average delay time refers to the quotient of the total delay time of all flights and the total number of flights. Compared with CASA, the optimization results of the MARL method in this paper reduce the number of flight delays and the average delay time by 33.7% and 36.7%, respectively. Therefore, we believe that the MARL method proposed in this paper has great potential in solving the DCB problem in the pre-tactical stage. Note that in this example, the absolute values of the number of flight delay and average delay time have no practical reference significance because the simulation is carried out based on abstract high-density scenarios. Readers only need to pay attention to the relative values of these two indexes in both the MARL and CASA method.

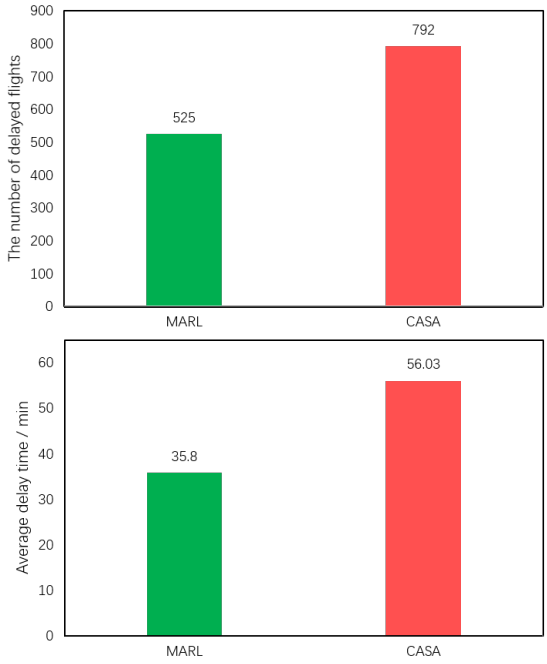


Fig. 6. Comparison of experimental results between MARL and CASA.

VII. Conclusion

In this paper, for the DCB problem in the large-scale and high-density scenario in the pre-tactical stage, we propose a set of MARL method based on GDP, in which we use double DQN, experience replay, adaptive ϵ-greedy strategy and DTDE framework. Compared with CASA, this MARL method can effectively reduce the number of flight delays and the average delay time. Therefore, it is potential for future application in the actual operation. Since this paper only considers GDP as a single means to solve the DCB problem, we can add the strategy of rerouting and speed control into it in the future. In addition, the experimental environment of this paper is relatively simple and ideal, which can improve the realness of simulation, and take the uncertainty factors into account, so that it can adapt to the application of the real world.

References

[1] ICAO, "Global Air Traffic Management Operational Concept," Montreal: ICAO, 2005.
[2] A. R. Odoni, "The flow management problem in air traffic control," in Flow control of congested networks. Springer, 1987, pp. 269–288.
[3] P. Choroba, L Hoorn, "Towards a more harmo-nised and wider use of short-term ATFCM measures (STAM)," Daejeon: ICAS, 2016.
[4] S. Ruiz, H. Kadour, P. Choroba, "An Innovative Safety-neutral Slot Overloading Technique to Im-prove Airspace Capacity Utilisation," Brussels: SESAR, 2019.
[5] A. Agustin, A. Alonso-Ayuso, L. F. Escudero, C. Pizarro, "Mathematical optimization models for air traffic flow management: A review," 2010.
[6] 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.

[7] 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.
[8] C. Taylor, T. Masek, and C. Wanke, "Designing traffic flow manageament strategies using multiobjective genetic algorithms," Journal of Guidance, Control, and Dynamics, vol. 38, no. 10, pp. 1922–1934, 2015.
[9] 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.
[10] L. WANG, H. WANG, "Combined optimization method for large-scale air traffic flow management under emergencies," Acta Aeronauticaet Astronautica Sinica, 2019, 40(08): 228-240.
[11] M. Sergeeva, "Automated airspace sectorization by genetic algorithm," Ph.D. dissertation, Universite Paul Sabatier (Toulouse 3), 2017.
[12] 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.
[13] 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.
[14] 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.
[15] V. Mnih, K. Kavukcuoglu, D. Silver, and A. Graves, "Playing atari with deep reinforcement learning," arXiv:1312.5602, 2013.
[16] V. Hasselt, Hado, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30. No. 1. 2016.
[17] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," arXiv:1511.05952, 2015.
[18] EUROCONTROL, "ATFCM operations manual: net-work manager," Brussels: SESAR, 2020.



https://dspace.org/

Demand and capacity balancing technology based on multi-agent reinforcement learning

Chen, Yutong; Xu, Yan; Hu, Minghua; Yang, Lei

2021-11-15
IEEE
https://doi.org/10.1109/DASC52595.2021.9594343
Chen Y, Xu Y, Hu M, Yang L (2021) Demand and capacity balancing technology based on multi-agent reinforcement learning. In: 2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC), 3-7 October 2021, San Antonio, USA

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