Generation of Optimized Trajectories for Congestion Mitigation in Fukuoka Approach Control Area Using Deep Reinforcement Learning

Yota Iwatsuki [1], Yasutaka Kawamoto[1,2] and Shin-Ichiro Higashino[2]

[1]Department of Aeronautics and Astronautics, Kyushu University, Fukuoka, Japan
[2]B787 Flight Crew Department, Japan Air Lines, Tokyo, Japan

Abstract. Fukuoka airport is the busiest single runway airport in Japan and the congestion has been increasing year by year. the COVID-19 pandemic has temporarily eased the congestion, but it is expected to increase again after the pandemic. Excessive radar vector by air traffic controllers to maintain aircraft separation during congestion causes economic and environmental losses due to increased flight distances and times. We attempted to generate optimal trajectories in the approach control area of Fukuoka airport using a deep reinforcement learning method based on a centralized Deep Q Network (DQN). Wind information was considered in the trajectory optimization using Mesoscale Model (MSM) data from the Japan Meteorological Agency. As a result, the optimized trajectory was found to be useful because a radar vector reduction of 23.7% in distance and 33.6% in flight time were attained compared with the recorded radar data provided as CARATS open data. The trajectories have the characteristics that all of the Ground speed (GS) of aircraft are made slower while directed straight to the intermediate fix (IF) after entering the approach control area.

Keywords: Air traffic management, Fukuoka approach control area, Reduce radar vector, Deep reinforcement learning

1 Introduction

The number of air passengers in the world has been increasing in both domestic and international flights until the Covid-19 pandemic in 2020 as shown in Fig. 1. Peach Aviation, which is the first LCC (Low-Cost Carrier) in Japan, commenced its services in 2012. Since then, LCCs have rapidly expanded throughout Japan, leading to a significant increase in air traffic flow. By 2018, the number of international air passengers per year exceeded 100 million as shown in Fig. 2.

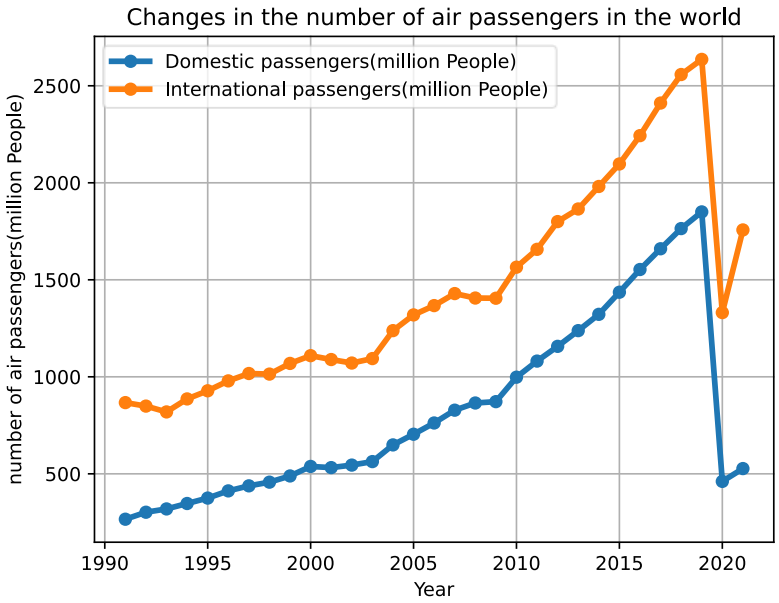

Fig. 1. Number of air passengers in the world [1].

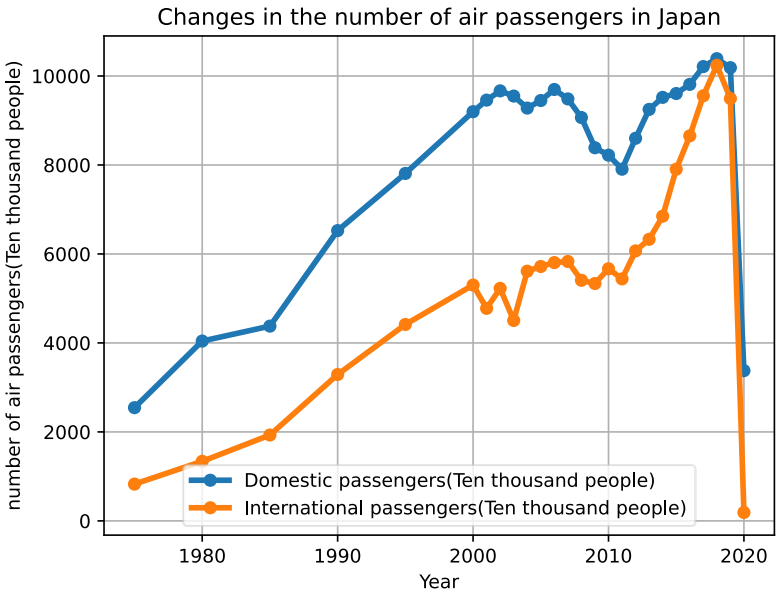

Fig. 2. Number of air passengers in Japan [2].

This situation made air traffic control at Fukuoka Airport very challenging. Fukuoka Airport is the fourth largest airport in Japan in terms of the number of aircraft landings per year [3], while it has only one runway. A comparison of the number of aircraft landings per year at major airports in Japan and those divided by the number of runways of the corresponding airport are shown in Fig. 3 The figure reveals that Fukuoka Airport holds the top position in the number of aircraft landings per runway. In 2016, Fukuoka Airport was designated as a "congested airport" by the Civil Aviation Bureau (CAB) of the Ministry of Land, Infrastructure, Transport, and Tourism of Japan (MLIT). When the air traffic in the approach control area becomes congested and the appropriate separation among the multiple aircraft cannot be maintained, the controller often employs radar vector and holding which tends to lead to delays in arrival traffic. Fig. 4 shows an example of all flight trajectories entering the Fukuoka approach control area on one day (15th December 2019) extracted from the radar data (CARATS open data) which will be explained later. The trajectories which are considered to receive radar vector are shown in red, and the remaining are shown in blue. It is seen from the figure that most of all of the flights received radar vector. The excessive radar vector results in longer flight distance and time, which lead to economic and environmental loss. Consequently, reducing radar vector due to congestion at the approach control area becomes a crucial and pressing issue.

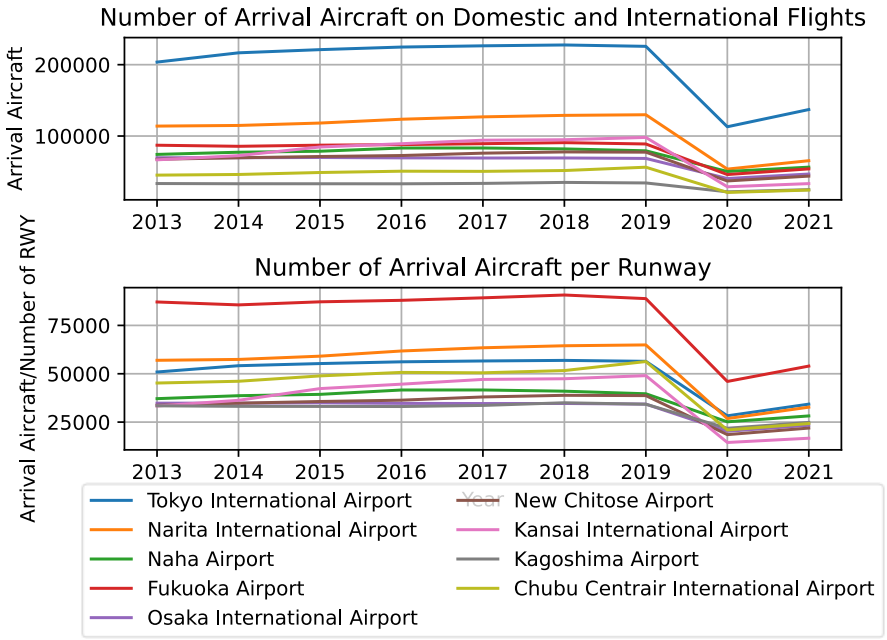

Fig. 3. Trends in the number of aircraft landing at Japanese airports and the number of aircraft landing per runway [3].

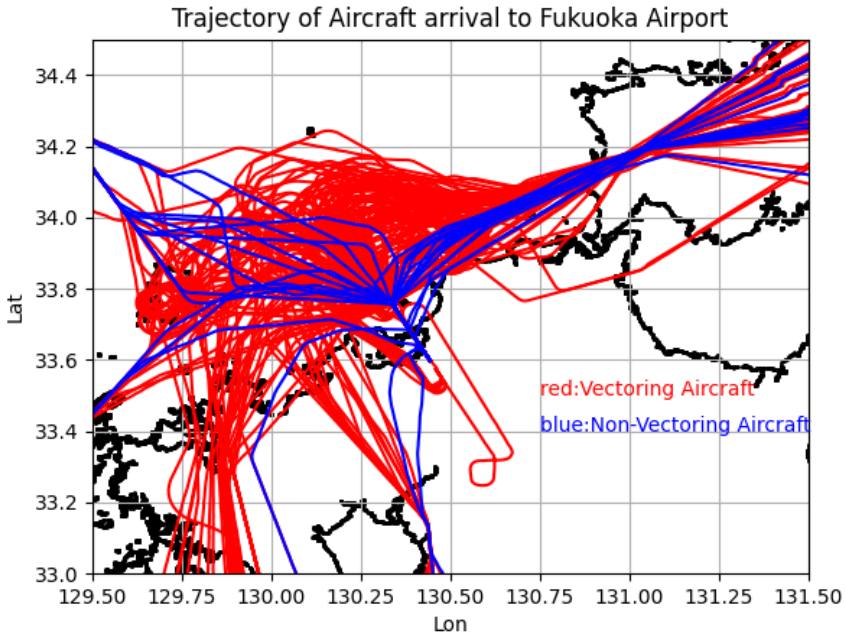

Fig. 4. All flight trajectories in the Fukuoka approach control area on 15th December 2019.

There are three major changes in recent years that will affect the problem of congestion at Fukuoka Airport. Firstly, it is anticipated that air traffic flow will rebound, following the end of the global COVID-19 pandemic, and air demand is projected to reach or even surpass the levels seen in 2019 after 2025 [4]. This implies that the congestion problem at Fukuoka Airport, which had been temporarily alleviated due to the pandemic will resurface as a concern once again. The second point concerns the additional runway construction project at Fukuoka Airport [5]. Presently, the plans are underway to construct an additional parallel runway. The operation using both runways is scheduled to commence in 2025. However, the new runway is planned to be constructed as a "close parallel runway" due to the limit of usable land area. Since simultaneous approaches and departures are not permitted in the current regulations in close parallel runways, its impact on the congestion reduction is deemed to be limited [6]. Therefore, it remains crucial to reduce congestion in the approach control area as in a single runway operation. The third point concerns the change in the approach chart. In July 2023, so-called "Point Merge" was introduced as one of the approach procedures at Fukuoka airport [7][8]. One advantage of this procedure is the reduction in workload of the air traffic controllers. It becomes easy for the controllers to maintain separations of the aircraft because the distances from the initial approach fix (IAF) which is the start waypoint of the final approach to all points on the "Point Merge" course are the same. However, from the perspective of the airline pilots, it is notorious for its rapid reduction in altitude, and longer flight distance along the course at low airspeed which lead to increase

in flight time and fuel consumption, and for the difficulty in predicting the aircraft's arrival time. The current approach for the congestion relief in an approach control area in Japan mainly relies on the radar vector instructions of the air traffic controllers' experience rather than scientific approach such as optimization and the method on some theory. If radar vector can be optimized to reduce flight distance and time while ensuring minimum and sufficient separation, it can lead to reduced congestions as well as flight time and fuel consumption. Given these considerations, our study aims to generate optimal trajectories that reduce excessive distance and time during radar vector in Fukuoka approach control area which is the busiest approach control area in single runway airports in Japan by using a deep reinforcement learning method.

2 Approach control area of Fukuoka airport

Fukuoka Airport is in urban region of Fukuoka City, Japan. It operates with a single runway (magnetic direction 160°/340°) with a domestic and an international terminal located on both sides of the runway respectively. RWY 16 (approach from the north) is used usually in southerly wind conditions, while RWY 34 (approach from the south) is used in northerly wind conditions. However, RWY 16 is actually used in large proportions of operating time for noise abatement [9]. In Japanese air traffic system, approach control areas are usually defined as a cylinder of which radius is from 40 to 60 nm centered at the airfield and the maximum altitudes of 15,000 ft or less. Within this airspace, the air traffic control for the approach control area and the terminal radar control are performed [10]. We have analyzed the congestion in the Fukuoka approach control area using CARATS (Collaborative Actions for Renovation of Air Traffic System) open data [12][13], which is a data set of aircraft radar data provided by MILT. The data includes aircraft positions, virtual callsigns, and aircraft types for scheduled instrument flight rule (IFR) flights at every 10 seconds [11]. Our congestion analysis showed that an increase in radar vector leads to an increase in total flight time, which is associated with congestion periods when there are six or more landing aircraft in the Fukuoka approach control area at the same time. We therefore decided to optimize the trajectories of about 10 aircraft during this congestion period in 3D airspace with appropriate separation. The case we used here is when 9 aircraft were flying simultaneously in the approach control area in 30 minutes from 19:10 to 19:40 on May 14, 2016. The flight trajectories of all 9 aircraft from the initial positions in the Fukuoka approach control area to intermediate fix (IF) are shown in Fig. 5. The star marks(★) in the figure show the waypoints where ATC hand-offs from the neighboring sectors are considered to be performed. The time is the most continuous period of congestion among the data stored in the CARATS open data for 2016. All 9 aircraft use RWY16, which is the most frequent operation in Fukuoka airport. The simulation covers the period from the moment when the aircraft enters the Fukuoka approach control area until it reaches the intermediate fix (IF), which is one of the fixes on an approach chart and leads to final approach for landing through final approach fix (FAF).

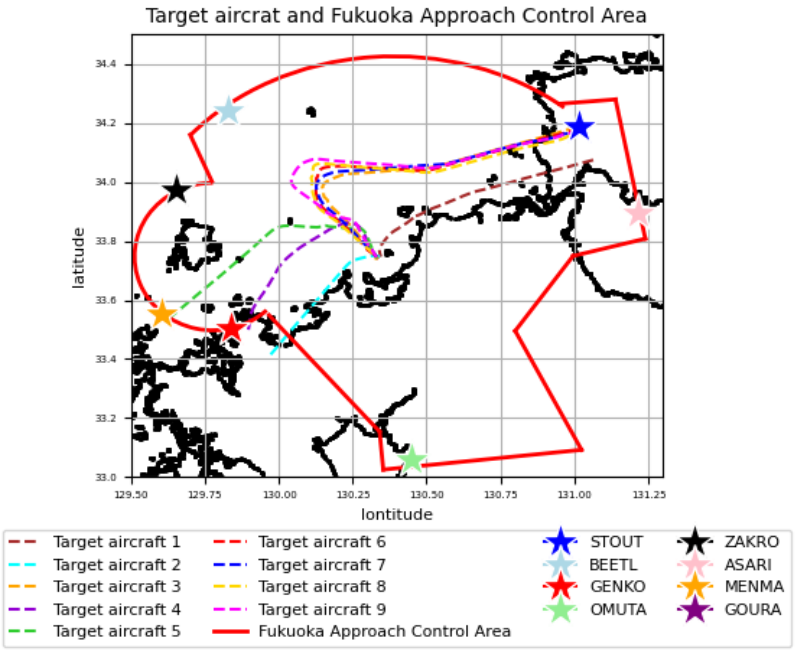

Fig. 5. The flight trajectories of all 9 aircraft from the initial positions in the Fukuoka approach control area to intermediate fix (IF).

3 Previous studies

There are several studies focusing on trajectory optimization in air traffic management. The concept of a self-sustaining air traffic control system was initially introduced by Hwinz Erzberger and colleagues at NASA in 2005 [14]. In Japan, there has been notable interest in the automation and high efficiency of air traffic control management, especially since around 2014, with research exploring subjects like arrival time management using dynamic programming [15]. The emergence of deep reinforcement learning has also gathered attention in the field of air traffic control management. In 2018, Marc Brittain et al. utilized hierarchical deep reinforcement learning algorithms to avoid collisions with limited route changes and speed adjustments within an environment based on NASA's Sector 33 [16]. While this initial study was limited to handling simple problems, the application of deep reinforcement learning in air traffic management has been explored by various research groups and has gradually evolved to tackle more complex collision avoidance scenarios [17][18]. Notably, researchers at IBM and Amazon, led by Supriyo Ghosh, applied ensemble learning to reinforcement learning, using a dataset of 1668 aircraft, and successfully dealt with intricate challenges such as collision avoidance and fuel cost variations [19]. These studies primarily focus on collision avoidance and enroute airspace control, neglecting the approach area. Generally, collision

avoidance in enroute airspace can be addressed by designing trajectories that enable aircraft with their different destinations to pass across with each other safely. Conversely, in the approach area, all aircraft have the same destination, necessitating the design of more time-sensitive and intricate trajectories. The study conducted by Supriyo Ghosh et al. [19] utilized a time step of 4 minutes, which is too long and may not be directly applicable to the problems in the approach area. In this region, each aircraft has a flight time of approximately 10 minutes, and the trajectory needs to be meticulously crafted within that timeframe. Currently, there are limited studies on the optimization of traffic flow in the approach control areas. In our previous studies, Ando employed genetic algorithms [20], and Igari utilized Q-learning [21]. However, both approaches presented certain difficulties. Genetic algorithms require extensive computation time, making them less suitable for real-time systems operating in complex and dynamic environments. On the other hand, Q-learning suffers from issues related to discretization of the state space, resulting in poor learning efficiency due to insufficient environment modeling. In this study, a deep reinforcement learning method is used for the optimization of multiple trajectories in approach control area incorporating wind information. The computation time using the model after learning is considerably shorter, and the state space is represented as continuous values, thereby allowing for more effective handling of complex problems.

4 Method

4.1 Overall Design

The optimization algorithm employed in this study utilizes deep reinforcement learning based on a centralized Deep Q network (DQN). To construct the simulation environment of the 3-D flight trajectories of multiple aircraft in the Fukuoka approach control area considering winds aloft. An object-oriented architecture was implemented using Python 3. Fig. 6 shows the Overall system design. The simulation environment includes three classes: "aircraft class," "controller class," and "DQN agent class,". Throughout the process, the "Aircraft" instance stores the information of individual aircraft, and the "Controller" instance extracts the “State” from the "Aircraft" instance and provides it to the "DQN Agent" instance. For the sake of maintaining separation among multiple aircraft, the neural network is partitioned into two parts; one is for the position in a horizontal plane and the other is for altitude. The "DQN Agent" instance receives “State” and utilizes it in the internal neural network to generate an “Action”. This “Action” is then passed to the "Controller" instance for the determination of the appropriate next change of true air speed (TAS), heading, and pressure altitude of an aircraft. The time step of the 3-D flight simulation of the aircraft as a point mass is 5 seconds, and during the simulation, the current “State”, future “State”, “Action”, and subsequent “Rewards” are accumulated and learned by the neural network until the aircraft arrives at intermediate fix (IF). By leveraging rewards over 3000 iterations, the learning process generates trajectories of multiple aircraft that effectively reduce the time and the distance of radar vector, while maintaining proper separation as guided by the neural network.

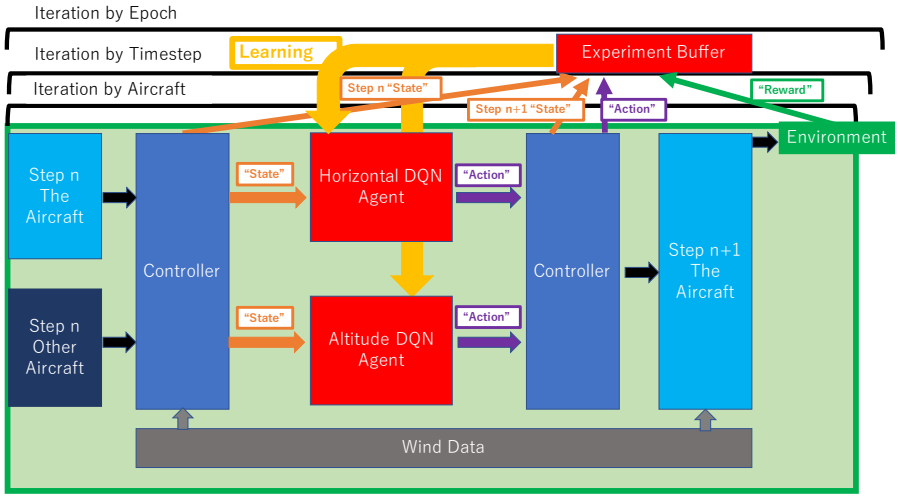

Fig. 6. Overall system design of the optimization.

4.2 Aircraft Class

The "Aircraft Class" encompasses essential information of each aircraft, including latitude, longitude, pressure altitude, true air speed (TAS), heading, time of entry into the approach control area, and the order of entry into the approach control area. The "Aircraft Class" receives additional information from the "Controller Class," such as “instructed TAS”, “instructed pressure altitude”, and “instructed heading “as shown in Fig. 7. which shows the flow of changing the actual information from the instructions. The airspeed in real air traffic communications (ATC) is instructed by indicated airspeed (IAS), but true airspeed is used in this study for the convenience of the calculation. The latitudes, longitudes, pressure altitudes, and ground speeds of the nine aircraft contained in CARATS open data mentioned previously at the boundary of the Fukuoka approach control area were used as the initial values of the nine aircraft. An instance of each aircraft is created using the "Aircraft Class" at the time of entry to the Fukuoka approach control area.

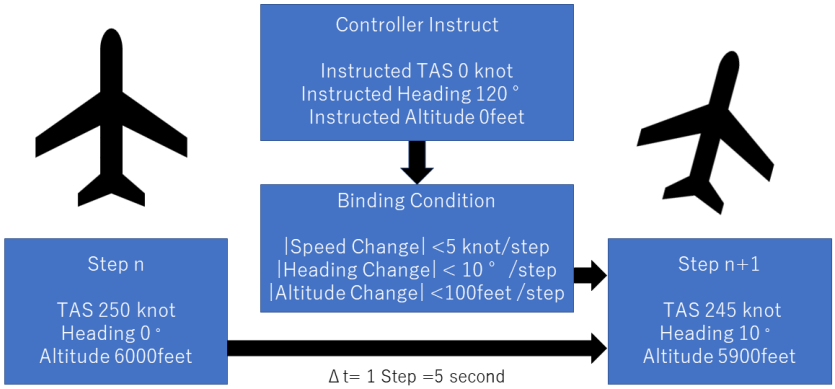

Fig. 7. Flow of changing the actual information from the instructions

The "Aircraft Class" has several “methods”. The “Acceleration and deceleration method” changes true airspeed (TAS_n) at step n to the extent that the speed constraint is satisfied to instructed $TAS_{,n+1}$ as shown in Equation (1).

$$\begin{cases} TAS_{n+1} = TAS_n + a \times \Delta t \ [m/s] \\ a = 1 \left(\dfrac{\left| TAS_{instructed,n} - TAS_n \right|}{5} > 1 \right) [m/s^2] \\ a = \dfrac{TAS_{instructed,n} - TAS_n}{5} \left(\dfrac{\left| TAS_{instructed,n} - TAS_n \right|}{5} \leq 0.51444 \right) [m/s^2] \\ \Delta t = 5 \ [s] \end{cases} \quad (1)$$

The parameter of 0.51444 (m/s^2) is the acceleration constraint. The reason for the half-way value is that 1 (knot/s) was initially established as the threshold value, and it is debatable what value is appropriate. However, according to the BADA manual [22], the threshold value for acceleration established for aircraft safety is 2 (feet/s^2), or about 0.6 (m/s^2), and 0.51444 (m/s^2) is not considered too large a risk for acceleration. It is important to note that this reinforcement learning method allows for constraints to be placed on the rate of descent, taking aircraft safety and comfort into consideration.
The “updating heading, latitude, longitude method” changes heading angle λ_n, latitude lat_n, and longitude lon_n as shown in Equation (2) to the extent that the heading angle constraint is satisfied.

$$\begin{cases} \lambda_{n+1} = \lambda_n + r_1 \times r_2 \times \Delta t \ [°] \\ r_1 = +1 \left(\lambda_{instructed,n} - \lambda_n \geq 0 \ or \ -180 > \lambda_{instructed,n} - \lambda_n \geq -360\right) [-] \\ r_1 = -1 \left(0 > \lambda_{instucted,n} - \lambda_n \geq -180\right) [-] \\ r_2 = 2 \left(\frac{\left|\lambda_{instructed,n} - \lambda_n\right|}{5} > 2\right) \left[\frac{°}{s}\right] \\ r_2 = \frac{\left|\lambda_{instructed,n} - \lambda_n\right|}{5} \left(\frac{\left|\lambda_{instructed,n} - \lambda_n\right|}{5} \leq 2\right) \left[\frac{°}{s}\right] \\ \Delta t = 5 \ [s] \\ lat_{n+1} = lat_n + \frac{(TAS_n \times \cos(\lambda_{n+1}) + WindSpeed_{SouthToNorth}) \times \Delta t \times 360}{2\pi \times Radius_{Earth}} \ [°] \\ lon_{n+1} = lon_n + \frac{(TAS_n \times \sin(\lambda_{n+1}) + WindSpeed_{WestToEast}) \times \Delta t \times 360}{2\pi \times Radius_{Earth} \times \cos(lat_n)} \ [°] \end{cases} \quad (2)$$

The "updating altitude method" changes the pressure altitude Alt_n to the extent that the pressure altitude constraint is satisfied for the instructed altitude Alt_{n+1} as in Equation (3).

$$\begin{cases} Alt_{n+1} = Alt_n + d \times \Delta t \ [feet] \\ d = -2000 \left(Alt_n - Alt_{instructed,n} > 2000/60\right) \left[\frac{feet}{s}\right] \\ d = -1000 \left(0 < Alt_n - Alt_{instructed,n} \leq 2000/60\right) \left[\frac{feet}{s}\right] \\ d = 0 \left(Alt_n - Alt_{instructed,n} < 0\right) \left[\frac{feet}{s}\right] \\ \Delta t = 5 \ [s] \end{cases} \quad (3)$$

The parameter of 2000/60 [feet/s] is a constraint on the rate of descent. What value is appropriate is a matter of debate. However, if we refer to the BADA PTF file [22], 2000 (feet/min) is a typical effective rate of descent, and there is no danger of the effective rate being too large. It is important to note that this reinforcement learning method allows for constraints to be placed on the rate of descent, taking aircraft safety and comfort into consideration.
With these three equations, the TAS, heading, and pressure altitude change at each step as shown in Fig. 7.

4.3 Wind Information

Wind information contained in the grid point value (GPV) data which is the numerically predicted atmospheric data provided by the Japan Meteorological Agency's Mesoscale Model (MSM) was utilized for calculating the position of individual aircraft in the simulation. To obtain wind information which corresponds to that in the simulation time duration, 3-hourly GPV data was linearly interpolated timewise and spacewise and used for updating the heading, latitude, and longitude of an aircraft in the "Aircraft Class"

and served as the inputs to the neural network. Methods for linear interpolation of the GPV data have been developed in previous studies [23][24].

4.4 DQN method

This section describes the principles and applications of the DQN method. The problem setup of reinforcement learning is formulated as a Markov decision process [25] in which the agent sequentially determines its actions based on the state of the environment. Reinforcement learning is the learning of the best behavioral rules (optimal measures) in the environment by using the rewards obtained as a result of the action. The behavioral policy π is a probability distribution that returns an "Action" according to the state "State", and this policy itself is treated as a variable in reinforcement learning [26]. The policy is learned with the goal of obtaining the optimal policy π^* that maximizes the expected return $E^{\pi}[R_0]$. The expected return $E^{\pi}[R_0]$ is the expected value of return R_0 when the strategy is run with π. The revenue R_0 is defined as in equation (4) [26].

$$R_0 = r(s_1, a_1) + \gamma r(s_2, a_2) + \gamma^2 r(s_3, a_3) + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4}$$

Where s is the "State", a is the "Action", $r(s, a)$ is the reward function that expresses how much "Action" a is worth in a certain "State", and γ is the discount rate. The discount rate is a positive number less than 1. A smaller discount rate means that the "Current Reward" is more important, while a larger discount rate means that the "Future Reward" is more important. To obtain the optimal policy π^*, the value of the policy itself must be evaluated, and DQN uses the value function Q. The value function is the expected return from a given state/action pair, (s, a) and is expressed as in Equation (5).

$$Q(s, a) = E[R_0 | s_0 = s, a_0 = a] \tag{5}$$

The value function Q makes the value function and measures to be updated [26]. There are two methods for updating the value function Q. One is the policy iteration method, in which the relationship between the policy and the value function is clarified for Equation (5), and then the policy π is updated so that the expected return is always improved. The other is the value iteration method, which takes advantage of the sequential nature of the value function and repeatedly updates the value function Q so that it increases, resulting in the optimal policy π^*. DQN belongs to the value iteration method. The optimal value function Q^*, which yields the optimal measure π^*, can be transformed using equation (5) as in equation (6)[26].

$$\begin{aligned} Q^*(s_t, a_t) &= E^{\pi^*}[R_0 | s_0 = s_t, a_0 = a_t] \\ &= r(s_t, a_t) + \gamma \sum_{s_{t+1}, a_{t+1}} p(s_{t+1} | s_t, a_t)\, max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned} \tag{6}$$

Where $p(s'|s,a)$ is the state transition probability from "State": s to the "Next State": s'. The subscript t represents time, and the transitions in equation (6) are time-series transitions. That is, the optimal value function is affected by the later optimal value function in a time-series manner. This equation (6) is called the Bellman optimal equation [25], and the problem of the Markov decision process boils down to solving the Bellman optimal equation, that is, finding the optimal value function Q^*.

Consider solving this Bellman optimal equation. It is noteworthy that the optimal value function Q^* at the current time is determined by the combination of the optimal value function Q^* at the future time, as shown in equation (6). For example, in a simple maze problem where the goal must be reached within a certain number of times, the possible future time states are finite, so the Bellman optimal equation can be solved rigorously. This method is called dynamic programming [25]. However, in complex problems such as air traffic control management, the possible future time states are very complex. For example, if the fuel limit is not considered, as in the present case, a future state that is clearly not optimal but keeps circling forever can be a candidate, and the number of candidate future states increases cumulatively depending on the positional relationships of multiple aircraft. Since dynamic programming is not realistic in such cases, it is necessary to solve the Bellman optimal equation in an approximate manner. This is the sample approximation [26] of the Bellman operator used in reinforcement learning. This means that the Bellman optimal equation is gradually solved by sampling the information of the learning environment little by little and reflecting the results in the value function Q by learning the rewards obtained by performing random action selection called search on the learning environment in reinforcement learning. This Bellman optimal operator is shown in Equation (7).

$$Q(s_t,a_t) = (1-\alpha)Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma max\big(Q(s_{t+1},a')\big)\right) \tag{7}$$

In this case, α is the learning rate and is a positive number less than or equal to 1. If α is large, learning is faster and convergence is harder, and if α is small, learning is slower, and convergence is easier. Q learning and DQN are learning methods that use this update rule to update the value function Q, converge the Q function to the optimal value function Q^*, and obtain the optimal strategy π^* [27].

To converge strictly to the optimal value function Q^* with this learning method, all possible states must be experienced to converge to the optimal value function. However, as mentioned above, it is impossible to experience all possible states in a complex problem such as air traffic control management. Therefore, in learning, it is important to strike a balance between search, in which actions are randomly selected to experience states that have never been experienced, and utilization, in which the action with the highest value function is selected to bring the value function Q close to the optimum using the states that have been experienced. To achieve this balance, the method of taking a random action with probability ε and selecting the action with the highest value function with probability (1- ε) is called the epsilon-greedy method [27] and is used in many cases, including this simulation It is employed in many cases.

The content of the value function Q is a two-dimensional table, the number of cases of "State" x the number of cases of "Action". In the case of the air traffic control

management problem, the number of cases of “State” is the number of states of each aircraft multiplied by the number of aircraft, which is extremely large, so it is unrealistic to handle it as a two-dimensional table. Therefore, the value function Q itself is approximated by Neural Network (NN). NN is a learning model that mimics the neural circuits of the brain and has a multilayered perceptron structure that takes an input, multiplies the weights together, calculates the sum, and returns an output [28]. It is known that this multilayer perceptron can approximate nonlinear functions because it can calculate multiple inputs with multiple weights and return an output, and by using this to approximate the value function Q, DQN is a method to solve the problem of the number of cases of “State” to be large [29].

When the value function Q is functionally approximated by NN, there is a problem that parameter updates are affected by the most recent observations, resulting in a large bias in the estimates and difficulty in convergence. This is because the observed states, actions, and rewards are time series, so the states before and after are strongly correlated, and overlearning occurs with respect to this correlation. To avoid this situation, there is a method called experience replay [30], in which observed series data are stored in a Replay Buffer, and when estimating the value function Q, the data are randomly extracted from the Replay Buffer to reduce the effect of autocorrelation and learn. This method is also utilized in this simulation.

In other words, to review the DQN flow in this simulation, the “Controller Class” creates “State” from the “Aircraft Class” and “Wind Data”. The NN is in the “DQN Agent class” and determines “Action” from “State” using the epsilon-greedy method. Equations (1) ~ (3) are used to update the Aircraft Class from action a. Once again, the “Controller Class” records the “State” of the next step, and then the “Reward” is passed from the learning environment. “State”, “Action”, the “State” of the next step, and “Reward” are stored in the Replay Buffer, and the NN model is trained using experience replay. This process is repeated 3000 times with different rewards r as described below, which is the application method of DQN in this simulation.

4.5 Controller Class

The "Controller Class" contains several crucial parameters, including the azimuth from the Intermediate Fix (IF), the aircraft's TAS, the minimum distance of distances between the aircraft and the IF, the distances between all combinations of two aircraft and the east-west and north-south wind speed values at the current latitude, longitude, and pressure altitude of an aircraft. These parameters are used to determine the “State” which passes as the inputs to the neural network. This "State" is obtained by the "DQN Agent class", which is used to select the Action to obtain the maximum value when making the exploitation in the epsilon-greedy method. Currently, the neural network in the "DQN Agent class" functions as a value function Q. The neural network checks the "Reward" received from the environment and the "State" after the action, then updates the value function Q by equation (7). The "Controller Class" includes a method to determine whether an aircraft has arrived at the IF or not. When an aircraft reaches the IF, its state changes from “Moving” to "Stopped" or "Arrived," and the aircraft is removed from the environment. The "Controller Class" also has a method to convert the “Action”

received from the neural network into “instructed TAS”, “instructed pressure altitude”, and “instructed heading” to the aircraft. In this simulation, we set four possible “instructed TAS” (200 knots, 225 knots, 275 knots, and 300 knots), six possible “instructed heading” (0°, 60°, 120°, 180°, 240°, and 300°), and six possible “instructed pressure altitude” (2100 feet, 4000 feet, 6000 feet, 8000 feet, 10000 feet, and 12000 feet). We intended to set these possible instructed values in order to make the learning process easier by making the “Action space” smaller. It may be thought that the possible instructed values shown above are too coarse, but by the binding conditions shown in Fig. 6, the velocity changes only 5 m/s, the heading angle changes only 10 degrees, and the pressure altitude changes only every 100 meters per every 5 seconds respectively because the simulation progresses every 5 seconds. The “State space” is large enough to be applicable for practical situations because even real ATC instructions are not so fine.

4.6 DQN Agent Class

“State” is then passed to the neural network within the "DQN Agent Class," and actions are obtained as instructions of the “Controller Class." The "DQN Agent Class" is instantiated twice, once as a neural network for the position in a horizontal plane and then for the altitude. The neural network configuration is 1024-256-128-action size. A batch size of 32 and a buffer size of 10000 are used. An empirical playback method is employed during training, with a learning rate of 0.05, and training is conducted using the Adam Optimizer [31].

4.7 Neural Network for horizontal position optimization

The horizontal neural network takes “State” such as the azimuth from the Intermediate Fix (IF), the distance between the aircraft and the IF, the aircraft's TAS, the minimum distance of distances between the aircraft and the IF, and the east-west and north-south wind speed values at the current latitude, longitude, and pressure altitude of an aircraft as inputs. The output for airspeed is one of 4 levels of “instructed TAS” and for heading is one of 6 levels of “instructed heading” for each time series. Therefore, the system's output consists of 24 patterns (4×6) of actions. Until the aircraft's arrive at the IF, the neural network learns through accumulated “Rewards” as shown in Table 1 for the first 2,000 learning steps.

Table 1. "Reward" for the first 2000 steps in horizontal neural networks

Goals to be achieved	Condition for imposing	"Reward"
Compliance with minimum aircraft separation	when the minimum separation between the aircraft and other planes is within 5 nm.	Penalty of -15,000 points
Compliance with speed limit when reaching IF	when aircraft reach IF.	|TAS at IF - 200 knots| × 20.
Reduction of flight time	No condition (imposed on all aircraft).	Penalty of – (flight time (seconds) × 20).

This pre-learning phase prioritizes reaching the IF in the shortest possible time while de-emphasizing separations. For the subsequent 3,000 learning steps, the reward changes as shown in Table 2.

Table 2. "Reward" for the subsequent 3000steps after 2000 steps in horizontal neural networks

Goals to be achieved	Condition for imposing	"Reward"
Compliance with minimum aircraft separation	when the minimum separation between the aircraft and other planes is within 5 nm.	Penalty of -30,000 points
Compliance with speed limit when reaching IF	when aircraft reach IF.	|TAS at IF - 200 knots| × 20.
Reduction of flight time	No condition (imposed on all aircraft).	Penalty of – (flight time (seconds) × 10).

The penalty of reduction of flight time was set to be smaller and the penalty of compliance with minimum aircraft separation was set to be larger than that of the pre-learning phase because in this learning phase, we are focusing on compliance with minimum aircraft separation and slowing down of the instructed TAS.

4.8 Neural Network for altitude optimization

The neural network takes "State" which is the same as the "State" for the neural network for horizontal position optimization as inputs. As its output, it provides six levels for "instructed altitude" as its "Action". Upon the aircraft's arrival at the IF, the neural network learns from the accumulated "Rewards" shown in Table 3. These reward values are used to ensure the altitude instructions for safer and more efficient during the approach phase.

Table 3. "Reward" for altitude optimization

Goals to be achieved	Condition for imposing	"Reward"
Keep appropriate altitude	when the distance from the IF is 20 nm.	Penalty of -10×|Altitude - 10000feet| points
Keep appropriate altitude	when the distance from the IF is 10 nm.	Penalty of -10×|Altitude - 6000feet| points
Keep appropriate altitude	when the distance from the IF is 5 nm.	Penalty of -10×|Altitude - 4000feet| points
Keep appropriate altitude	when the distance from the IF is 2.5 nm.	Penalty of -10×|Altitude - 2000feet| points

4.9 Computational Environment

The computational environment used for this study consisted of Ubuntu 22.04 running on a Ryzen 5 5600G processor with an RTX 3090 GPU, 128GB of memory, and a 1TB SSD. Python 3.10.6 64-bit was the programming language used for implementation. The primary library utilized in this research was dezero, which is a library based on Chainer and incorporates PyTorch design principles. Additionally, we employed pygrib to handle GPV data, allowing for effective data management and analysis.

5 Results

The flight trajectories of the nine aircrafts in horizontal plane obtained by the DQN are shown as solid lines in Fig. 8 together with the trajectories of CARATS open data as broken lines.

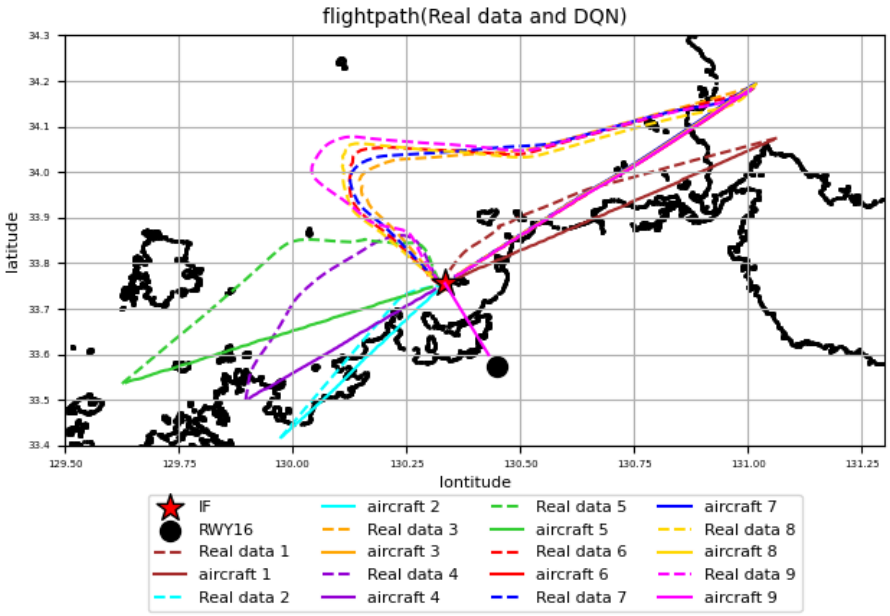

Fig. 8. Generated trajectories in horizontal plane obtained by DQN.

The trajectories of the same aircraft are shown in the same color. The resulting trajectories are successfully generated in which the radar vector is minimized. From Fig. 8, it is evident that all optimized trajectories (solid lines) take almost direct paths to the Intermediate Fix (IF) while most of the aircraft om CARATS open data (broken lines) evidently received radar vector and their trajectories widely spread to offshore. Fig. 9 shows time history of the distances to IF in generated trajectories by comparing the distances to IF in CARATS open data. Fig. 10 shows time history of GS (Ground Speed) in generated trajectories and GS in CARATS open data. From Fig. 9 and Fig. 10, we observe that all aircraft are heading straight to the IF without any radar vector and holding so that the distances to IF decrease uniformly. However, their true air speeds shown in Fig. 10 seem rapidly reduced to slower speeds than those observed in the CARATS open data. At Fukuoka approach control area, when there are six or more arriving aircraft, approach control is performed by two controllers. One instructs the first five and the other instructs the remaining aircraft. In such a situation, the air traffic controller in charge of the remaining aircraft is considered to issue radar vector instructions with ample separation to the aircraft instructed by the first controller. This tendency could explain the considerable radar vector observed for aircraft #6, #7, #8 and #9 in Fig. 9 shown as the trajectories in CARATS open data. However, since DQN optimization employs a neural network that is shared across all 9 aircraft for learning, there is no need for excessive radar vectors as human controllers did. Instead, the minimum distance path can be generated, leading to a significant reduction in radar vector

distance for later arriving aircraft. Consequently, the radar vector distance is also greatly reduced for the aircraft arriving later in the sequence. In Figure 8, a reversed arriving order to IF against the entering order to the approach control area is seen for aircraft #1(solid brown) and aircraft #4(solid purple) in the DQN result, while aircraft#1(broken brown) reached IF in advance of aircraft #4(broken purple) in CARATS open data as "First come, first served" basis. Since the winds aloft predominantly blow from west to east in the Fukuoka approach control area, it is thought that the reduction in total flight time was performed by the DQN by accounting for the wind speed which makes the ground speed of the aircraft coming from the west (aircraft #4) faster than that coming from the east (aircraft #1). This observation suggests that sequencing arrival aircraft considering several factors including wind speed is better than the sequencing by "First come, first served" basis, as also highlighted in a previous study [32]. The efficiency of reducing the speed of all aircraft, as depicted in Figure 10, requires careful consideration. The reduction in total flight time and total flight distance were attained by slowing down all aircraft, but stall speed has to be considered in the future work, though the lowest speed attained here is not below typical stall speeds of airlines.

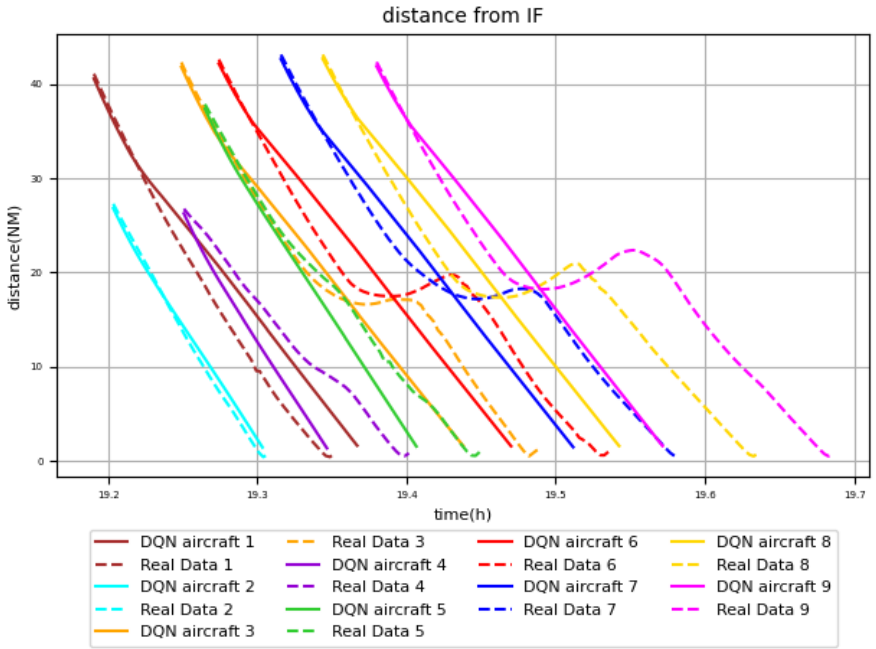

Fig. 9. Time history of the distances to IF in generated trajectories.

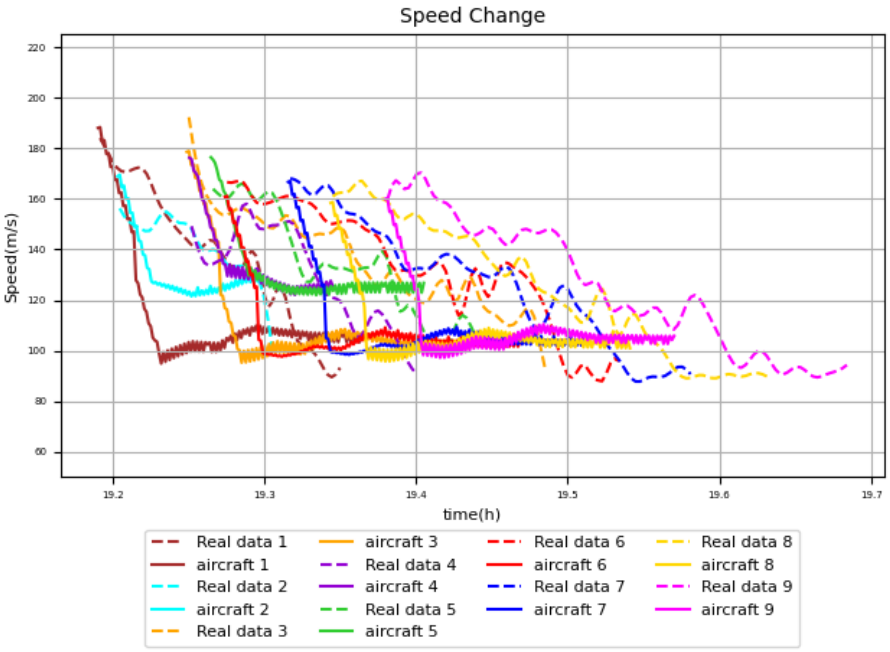

Fig. 10. Time history of aircraft GS (ground speed).

Fig. 11 shows the True Air Speed (TAS), Ground Speed (GS), Uwind (wind blowing from west to east) received by the aircraft, Vwind (wind blowing from south to north) received by the aircraft, and azimuth. Some may be concerned that the sudden drop in GS for aircraft #1, #3, #6, #7, #8, and #9 in Fig. 10 may be unrealistic. However, as can be seen in the red-circled area, there is no abrupt change in TAS or heading. Therefore, the change in GS of the aircraft is due to the strong influence of winds such as Uwind and Vwind. Aircraft #1, #3, #6, #7, #8, and #9 are coming from the east, as shown in Fig. 8. From a certain region, the Uwind strength increases from 10 m/s to 20 m/s, which may have slowed the aircraft's ground speed to a much lower value than the TAS. Thus, we can see that the simulation itself is performed under sufficiently realistic constraints.

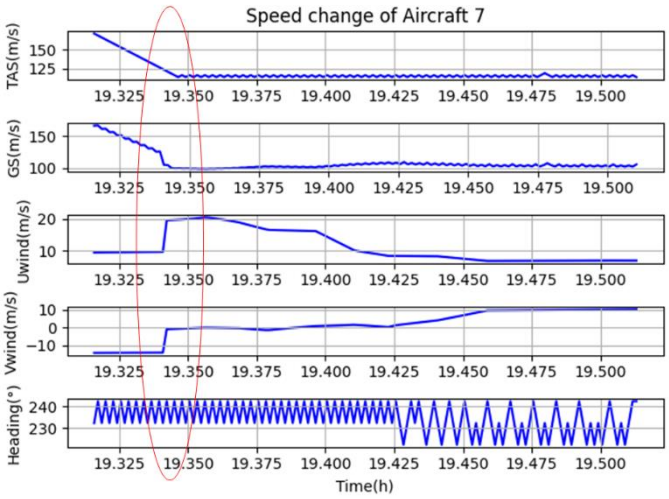

Fig. 11. TAS, GS, Wind Speed and heading history of Aircraft7 obtained by DQN.

Figure 12 shows time history of the altitude of the nine aircraft. It shows that the trajectories that can be applicable to a real situation as a continuous descent were obtained, that is, almost consistent reduction in altitude which aligns with the intended trajectory were obtained. However, there are several parts to be improved in the result. For example, some parts in the time history of altitudes exhibit discontinuities and uneven variations are seen. Additionally, the altitude in some parts violates minimum vectoring altitudes, which should be considered as well as the stall speeds in the further refinement.

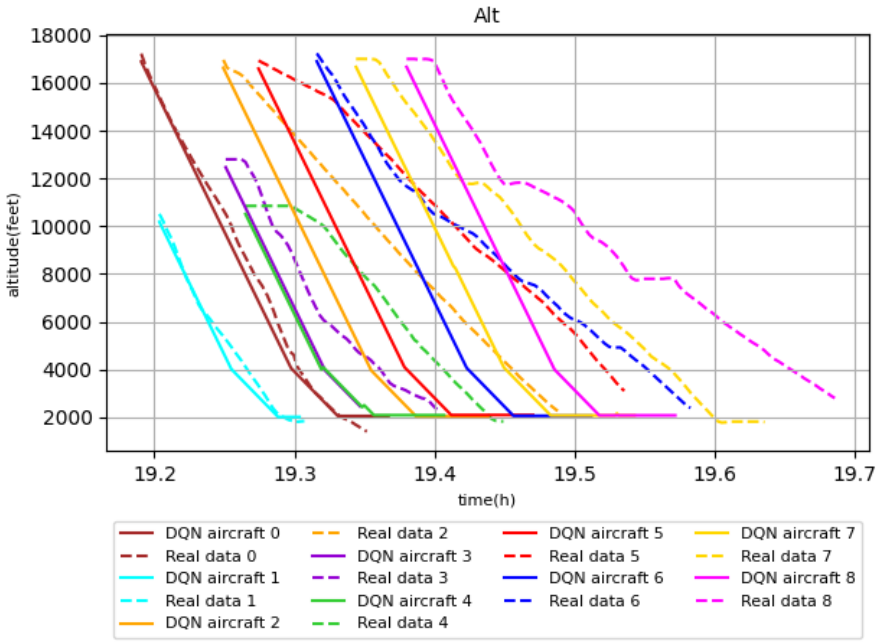

Fig. 12. Time history of the altitude obtained by DQN.

Figure 13 shows the time history of the horizontal distances between the two aircraft (i.e., separation) of all combination in the approach control area before reaching IF. The separations between the combinations of all 9 aircraft is 36, but only those cases where the minimum separation is less than 30000m are shown for simplicity. It is seen from the figure that there is no violation regarding the separation limit of 5nm. We planned the reward to ensure a separation of no less than 5 nm (9260m) between the two aircraft by imposing substantial penalty. Thus, the intended objective of preserving adequate aircraft separation is effectively achieved.

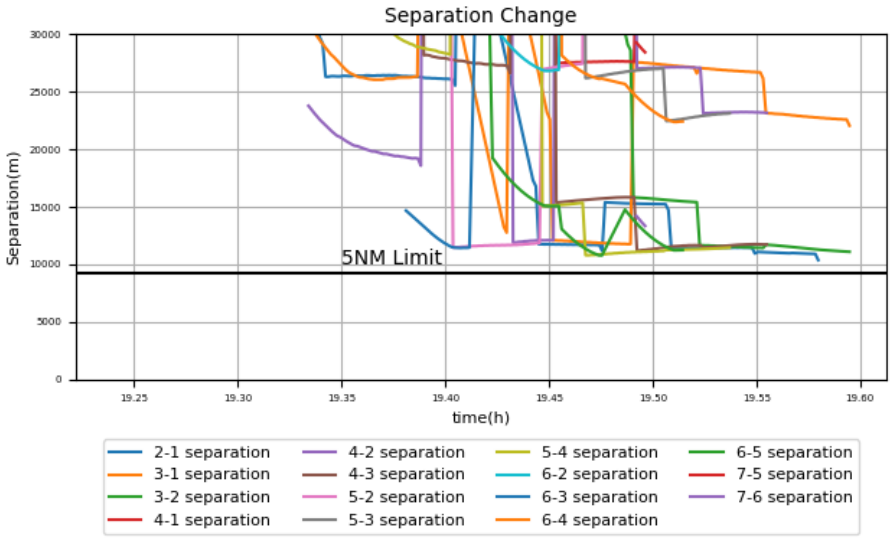

Fig. 13. Aircraft-to-aircraft separation changes

Figure 14 displays the comparison of the flight distances of the optimized flight trajectories of all aircraft and the flight distances of the corresponding aircraft in CARATS open data. The flight distance of all aircraft was reduced in DQN results compared to those in CARATS open data. The cumulative flight distance covered within the approach control area of the nine aircraft is 497.6 nm (922km), while the cumulative distance obtained by the DQN trajectory optimization is 330.4nm (612km). Consequently, this trajectory optimization resulted in a remarkable 33.6% reduction in total flight distance compared to that in the CARATS open data.

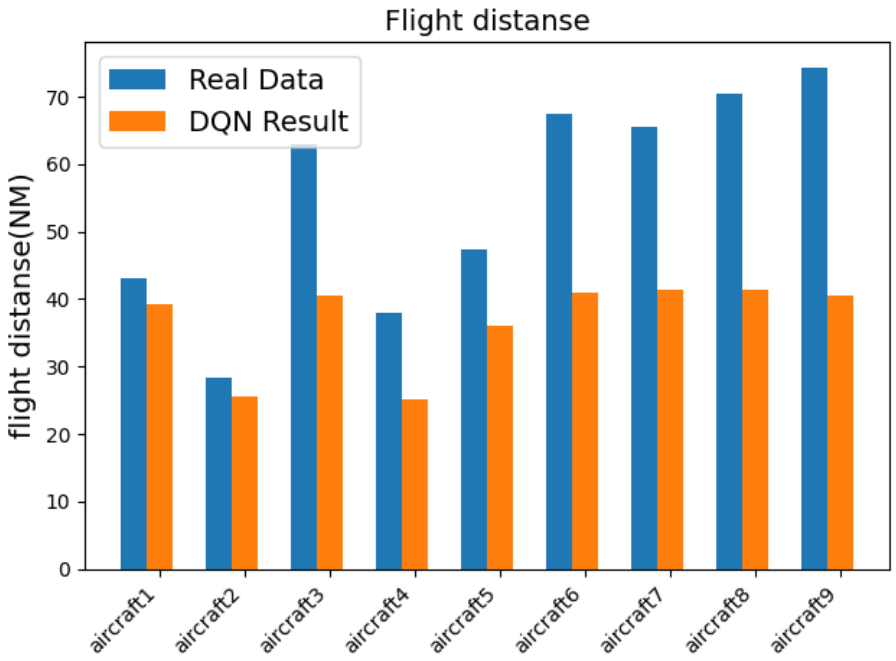

Fig. 14. Comparison of the flight distances of the nine aircraft.

Figure 15 shows the differences in flight time of the nine aircraft similarly comparing the flight time obtained by DQN and those in CARATS open data. The flight time of all aircraft except for the first aircraft was reduced in DQN results compared to those in CARATS open data. The total flight time within the approach control area of the nine aircraft in the CARATS open data is 7065 s, while the total flight time of the optimized trajectories by the DQN is 5390 s. This trajectory optimization resulted in a significant 23.7% reduction in total flight time compared to the CARATS open data in spite of the reduced airspeed explained earlier.

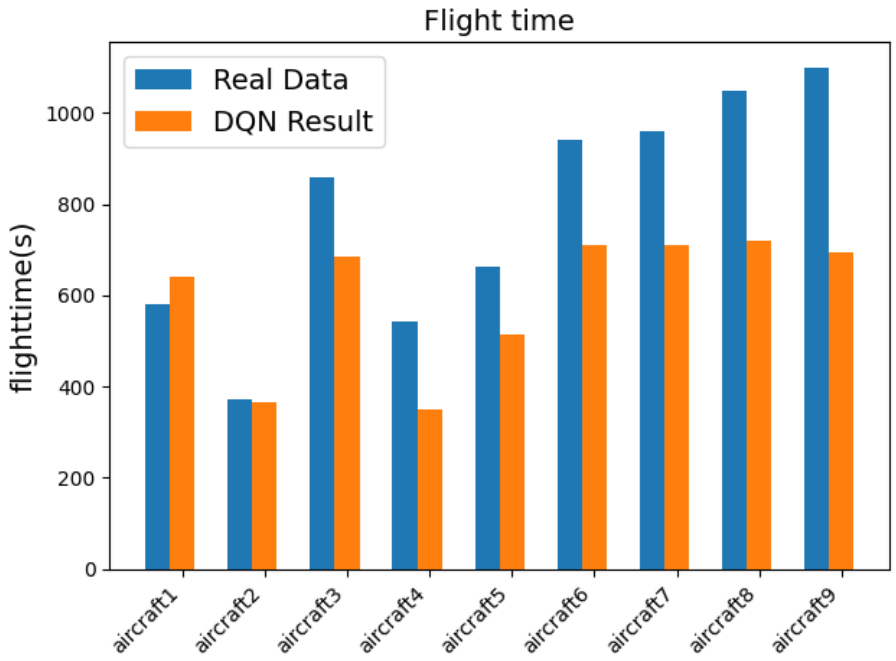

Fig. 15. Comparison of the flight times of the nine aircraft.

Fig. 16 shows the variation of the total reward obtained by each aircraft during one epoch with respect to the horizontal NN. The horizontal axis is the epoch, and the vertical axis is the total reward for the nine aircraft. Since the original data was too scattered to show a clear trend, a low-pass filter was applied after taking a moving average of 1,000 data points. For the moving average, data up to +500 and data down to -499 were added and divided by 1000. For the epochs between 3000 and 3500, which are necessary for the moving average, the data is prepared by learning up to 3500, and for the data below 0, the same value as 0 is inserted as a dummy. From Fig. 16, we can see that the learning of the NN in the horizontal direction progressed smoothly.

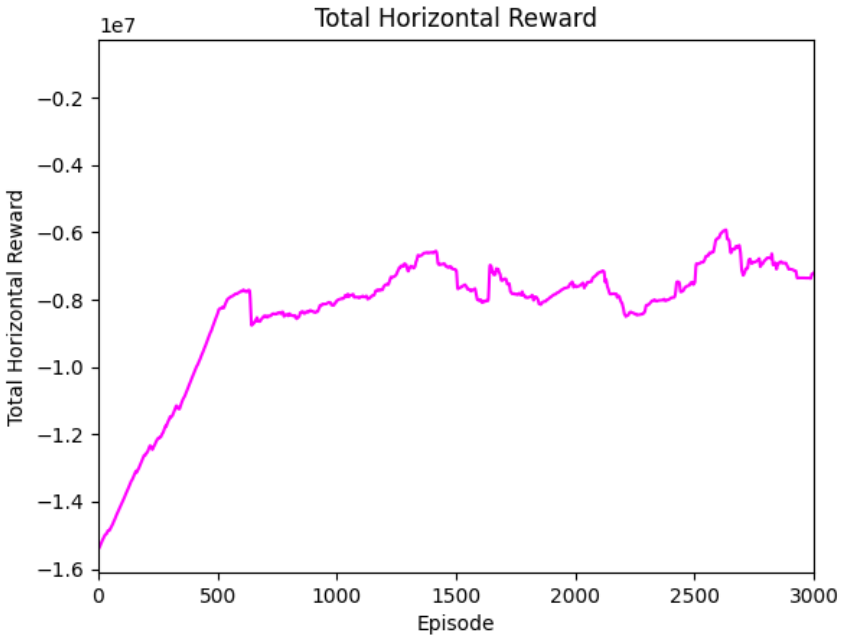

Fig. 16. Total Horizontal Reward for 9 Aircraft.

Fig. 17 shows the change in the total reward obtained by each aircraft during one epoch with respect to the NN in the altitude direction. The horizontal axis is the epoch, and the vertical axis is the total reward for the nine aircraft. Since the original data was too scattered to understand the trend, a low-pass filter was applied after taking a moving average of 1000 times of data as in Fig. 16. Fig. 17 shows that the learning of the NN in the altitude direction progressed once, the performance of the model deteriorated, and the model learned again to reach equilibrium. In the state of equilibrium, as shown in Fig. 11, the direction of altitude seems to work, so the final performance of the model is not considered to be bad. As for the reason why, the learning did not progress uniformly, early convergence due to the simplicity of the problem itself and the occurrence of overlearning are suspected, so it may be necessary to consider learning methods such as early stopping [26] to keep the learning in a good state.

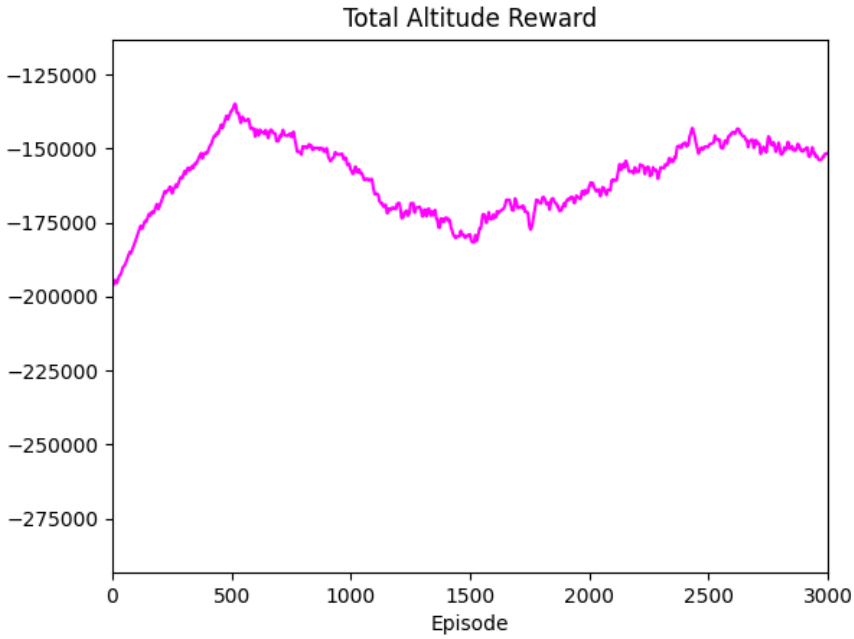

Fig. 17. Total Altitude Reward for 9 Aircraft.

6 Conclusion

We have attempted to generate optimal flight trajectories using DQN in the Fukuoka approach control area in the busiest time. Leveraging deep reinforcement learning, we successfully generated highly efficient trajectories for multiple aircraft. These trajectories directed by the “AI controller” resulted in a great reduction of 33.6% in total flight distance and 23.7% in total flight time. Notably, these optimized trajectories exhibit characteristics of heading directly to the Intermediate Fix (IF) and reducing the airspeed of all aircraft, while ensuring a separation of more than 5 nm between aircraft. As a result, we have compelling evidence that deep reinforcement learning approaches are highly effective in alleviating congestion in the Fukuoka approach control area. The approach control area environment presents a significant level of complexity, and, to the best of the author's knowledge, this study represents the first successful utilization of deep reinforcement learning for congestion relief in the approach control area. Unlike trajectories generated by conventional genetic algorithms and other methods, DQN can yield results promptly once the weights are determined, making this research potentially valuable for effectively alleviating congestion in real-time systems in the future. It is important that our ultimate objective is to reduce congestion in the real-world

scenario, but some of the factors were not fully considered in this simulation. To achieve this, we are going to work on developing a more realistic simulation environment by encompassing a broader range of supported cases and incorporating real-world constraints, including aircraft characteristics and the characteristics of the airspace, and on designing compensation strategies and enhancing models, such as reward clipping and optimizing neural network structures as the future work. By undertaking these efforts, the gap between simulation and real-world implementation can be bridged, and it will ultimately contribute to the development of more efficient and practical air traffic control systems in real-time environments.

References

1. ICAO / Annual Report of the Council, https://www.icao.int/about-icao/Pages/annual-reports.aspx, last accessed 2023/07/23.
2. Japan Civil Aviation Bureau, Ministry of Land, Infrastructure, Transport and Tourism.: [Situation Surrounding Aviation and Future Issues and Initiatives], Kuukou wo torimaku zyoukyou to kongo no kadai torikumi (in Japanese). March 2022, https://www.mlit.go.jp/policy/shingikai/content/001467196.pdf, last accessed 2023/07/23.
3. Ministry of Land, Infrastructure, Transport and Tourism.: [Ranked by airport, based on each fiscal year], Kokudokoutuushou, kuukoubetsu zyunnihyou(in japanese). https://www.mlit.go.jp/koku/15_bf_000185.html, last accessed 2023/07/23.
4. IATA.: Tourism Economics Air Passenger Forecast, March 2022. https://www.iata.org/en/pressroom/2022-releases/2022-03-01-01/, last accessed 2023/07/23.
5. Port and Aviation Department, Kyushu Regional Development Bureau, Ministry of Land, Infrastructure, Transport and Tourism, Japan.: [Environmental Impact Statement for Fukuoka Airport Runway Expansion Project], Kokudokoutsuusyou, kyuushuuseibikyoku, kouwannkoukuubu(in japanese). https://www.pa.qsr.mlit.go.jp/fap/assessment/index.html#stage_1, last accessed 2023/07/23.
6. Japan Ministry of Land, Infrastructure, Transport and Tourism Civil Aviation Bureau.: [Enhancement of Haneda and Narita Airports (Study of Facility Aspects)], Kokudokoutsuushou koukuukyoku, Haneda Narita kuukou no kinou kyouka ni tsuite (in Japanse),: January 2014 ,https://www.mlit.go.jp/common/001030706.pdf, last accessed 2023/07/23.
7. Japan Aeronautical Information Service Center.: KIRIN ARRIVAL. https://aisjapan.mlit.go.jp/html/AIP/html/20230713/frame/index-en-JP.html#efct=20230713, last accessed 2023/07/23.
8. Aeronautical Information Service Center.: OSTEP ARRIVAL/ISKUP ARRIVAL/ ATSAG ARRIVAL/ SARUP ARRIVA. https://aisjapan.mlit.go.jp/html/AIP/html/20230713/frame/index-en-JP.html#efct=20230713, last accessed 2023/07/23.
9. Civil Aviation Bureau, Ministry of Land, Infrastructure, Transport and Tourism.: [Evaluation of Fukuoka Airport Runway Expansion Project at the Time of Adoption of New Project], Fukuokakuukou kassouro zousetsuzigyou ni okeru shinkizigyou saitakuzi no hyouka ni tusite(in japanese). 2014.
10. Kinya Fujiishi.: [Color Illustrated Introduction to Air Traffic Control], Kara- zukai de wakaru koukuukannsei you nyuumon (in Japanese). SB Creative Corporation, 2015.

11. Megumi, Oka.: [CARATS Open Data Overview Briefing], CARATS open data gaiyou setsumei (in Japanese). Electronic Navigation Research Institute, CARATS Open Data Application Promotion Briefing, (2019).
12. Shunsuke, Mori., Naoto, Kitazume., Shinichiro, Higashino., Yoshikazu, Miyazawa.: A Study on the current status of air traffic flow in the Fukuoka Airport's terminal area. Proceedings of the 54st Aircraft Symposium, (2016).
13. Yasutaka, Kawamoto., Yota, Iwatsuki., Shinichiro, Higashino.: Study on the Cause of Air Traffic Delay in Fukuoka Approach Control Area Using CARATS Open Data. APISAT 2023.
14. Erzberger, H.: "Automated conflict resolution for air traffic control.". 25TH INTERNATIONAL CONGRESS OF THE AERONAUTICAL SCIENCES, (2005).
15. Yuki, Higuchi., Naoto, Kitazume., Keiichi, Tamura., Tomoyuki, Kozuka., Yoshikazu, Miyazawa., Mark, Brown.: Optimal Arrival Time Assignment and Control Analysis Using Air Traffic Data for Tokyo International Airport. AIAA Paper 2017-1246, AIAA Guidance, Navigation and Control Conference, Grapevine, Texas, (2017).
16. Marc, Brittain., Peng, Wei.: Autonomous Aircraft Sequencing and Separation with Hierarchical Deep Reinforcement Learning. ICRAT, (2018).
17. Dalmau, R. and Allard, E.: Air Traffic Control Using Message Passing Neural Networks and Multi-Agent Reinforcement Learning. In 10th SESAR Innovation Days (SID), Virtual Event, (2020).
18. Marc, Brittain., Peng, Wei.: Autonomous Separation Assurance in A High-Density En Route Sector: A Deep Multi-Agent Reinforcement Learning Approach. 2019 IEEE Intelligent Transportation Systems Conference – ITSC, (2019).
19. Supriyo, Ghosh., Sean, Laguna., Shiau, Hong, Lim., Laura, Wynter., Hasan, Poonawala.: A Deep Ensemble Method for Multi-Agent Reinforcement Learning: A Case Study on Air Traffic Control. Vol. 31 (2021): Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, (2021).
20. Ryo, Ando., Shunsuke, Mori., Shinichiro Higashino.: Optimization of Air Traffic Flow in Fukuoka Approach Control Area. Lecture meeting of The Japan Society for Aeronautical and Space Sciences Western Branch, (2019).
21. Youta, Igari., Shinichirou, Higashino. and Ryou, Andou.: The Research on Automation of Air Traffic Control in Fukuoka Approach Area with Reinforcement Learning. Lecture meeting of The Japan Society for Aeronautical and Space Sciences Western Branch, (2020).
22. Eurocontrl. : Base of Aircraft data (BADA), last accessed 2023/09/24. (https://www.eurocontrol.int/model/bada)
23. Navinda, Kithmal, WICKRAMASINGHE., Yuto, MIYAMOTO., Akinori, HARADA., Tomoyuki, KOZUKA., Sadanari, SHIGETOMI., Yoshikazu, MIYAZAWA., Mark, BROWN, and Yutaka, FUKUDA.: Flight Trajectory Optimization for Operational Performance Analysis of Jet Passenger Aircraft Trans. JSASS Aerospace Tech. Japan, Vol. 12, No. APISAT-2013, pp. a17-a25, (2014).
24. Navinda, Kithmal, Wickramasinghe.: OPERATIONAL PERFORMANCE ANALYSIS ON JET PASSENGER AIRCRAFT VIA TRAJECTORY OPTIMIZATION, Doctoral dissertation in aerospace engineering.: Kyushu University Graduate School of Engineering, (2015).
25. Richard, Bellman.: Dynamic programming.: Dover Publications, (1957), Republished (2003).
26. Tetsuro, Morimura.: [Reinforcement Learning], Kyoukagakusyuu (in Japanese).: Koudansya, 2019.
27. Watkins, C.J.C.H.: Learning from Delayed Rewards.: PhD thesis, Cambridge University, Cambridge, England, (1989).

28. Takashi, Matsubara.: Properties and Applications of Functions that Deep Learning Approximates.: Journal of the Japan Neurological Society Vol. 25, Np. 4, pp175-180, (2018).
29. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop, (2013) .
30. Long-Ji, Lin.: Self-improving reactive agents based on reinforcement learning, planning and teaching.: Machine Learning volume 8, pages293–321 (1992).
31. Diederik, P. Kingma, Jimmy, Ba.: Adam: A Method for Stochastic Optimization. conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
32. Adriana Andreeva-Mori a, Shinji Suzuki, Eri Itoh. Rule derivation for arrival aircraft sequencing. Aerospace Science and Technology 30 (2013) 200–209, (2013).