Reinforcement Learning for Collaborative Multi-Airport Slot Re-Allocation Under Reduced Capacity Scenarios

Anh Nguyen-Duy[1)†] and Duc-Thinh Pham[1)]

[1)] *Air Traffic Management Research Institute, Nanyang Technological University, Singapore*
[†] *email: nguyendu002@e.ntu.edu.sg*

Airport Collaborative Decision Making (A-CDM) is currently implemented to foster collaboration for efficient airport slot allocation. In the ASEAN region, where a central decision-making authority is not available, each airport reserves its autonomy in managing its own airport resources, which leads to different decision-making policies. An effective collaborative airport slot allocation approach needs to demonstrate its ability to collaborate with different slot allocation policies. Reinforcement Learning, a learning-based approach, can make use of interactions between airports to capture the underlying policies of other airports. In this paper, we consider a multi-airport system with different slot allocation policies, consisting of a Reinforcement Learning airport agent interacting with fixed-policy airport agents. We want to validate if the Reinforcement Learning agent can utilize interactions between airports to learn to reallocate slots efficiently under reduced capacity scenarios. We perform validation on the Hong Kong-Singapore-Bangkok hub, with the 2018 OAG data. The performance of the Reinforcement Learning agent is compared with the Nearest Heuristic, which assigns delays based on the nearest available slots. Results show that the Reinforcement Learning agent performs significantly better than the Nearest Heuristic under a heavy-reduced capacity scenario, with a total delay of 84 and 107, respectively. For a medium-reduced capacity scenario, the Reinforcement Learning agent closely resembles the performance of the Nearest Heuristic, with a total delay of 45 and 41, respectively.

Key Words : Reinforcement Learning, Airport Collaborative Decision Making, Airport Slot Re-Allocation

1. Introduction

Airport Collaborative Decision Making (A-CDM) is in the process of implementation to allow stakeholders in the interconnected multi-airport system to effectively collaborate and come up with a joint agreement while optimizing the different objectives of the stakeholders.[1)] In regions without a central authority, such as ASEAN, stakeholders of different authorities reserve their autonomy in managing their resources, one of which is the utilization of airport slots.[12)] In the context of airport slot allocation, the IATA Worldwide Airport Slot Guidelines (WASG) mention "to ensure the most efficient declaration, allocation and use of available airport capacity in order to optimize benefits to consumers, taking into account the interests of airports and airlines, while minimizing congestions and delays".[2)] Different objectives can be derived from the above guidelines, such as minimizing the number of unaccommodated movements, the maximum displacement per movement, and the total system delay, etc. Different stakeholder groups belonging to different authorities can formulate different subsets of the above objectives while taking into account the local guidelines.[3)] Different subsets of objectives can lead to different airport decision-making policies, not to mention when considering the different underlying optimization algorithms. In such a complex multi-airport system, there is no universal model that can holistically capture all stakeholders' preferences.[3)] Furthermore, in regions like ASEAN, it is not guaranteed that the centralized solution will be accepted by all parties. Minor disagreements can lead to the collapse of the system solution. A decentralized approach, which allows different airports to form their own policies, is necessary for deployment in reality.

Current airport slot allocation approaches range from exact methods, such as Mixed Integer Linear Programming (MILP), to sub-optimal methods, which are heuristics and learning-based approaches. Due to the NP-hard nature of the slot allocation problem, sub-optimal methods are preferred over exact methods.[4)] Heuristics rely on a fixed set of rules, which cannot make use of new scenarios, thus limiting the possibility of finding new solutions and being vulnerable to changing scenarios[5),6)] Reinforcement Learning, on the other hand, can automate the search process and freely explore new strategies. Additionally, learning-based methods such as Reinforcement Learning have the ability to utilize new scenarios as inputs to enhance performance, thereby demonstrating the ability to self-evolve[7),8)] Reinforcement Learning also has a broad literature in decentralized coordination, making it a suitable candidate for the multi-airport slot allocation problem.[9)]

Airport slot allocation approaches utilizing Reinforcement Learning use a Centralized Training and Decentralized Execution (CTDE) paradigm, with flights modeled as agents[8)10),11)] This paradigm implicitly assumes that all airports in the multi-airport system adopt the same centralized training policy. It is unlikely that all airports pertaining to different authorities will adopt the same decision-making policy. A pragmatic way of studying Reinforcement Learning in a collaborative multi-airport system is to consider an

airport system with different decision-making policies. To fill in the research gap, we study the use of Reinforcement Learning in a multi-airport system, where the Reinforcement Learning airport agent interacts with other airports adopting a different fixed policy. We aim to validate whether the Reinforcement Learning agent can make use of the interactions between airports to learn an efficient slot re-allocation policy under reduced capacity scenarios. We consider the Hong Kong-Singapore-Bangkok hub for the problem scope. We represent the Changi airport (Singapore) as the Reinforcement Learning agent. We assume scenarios where the capacity at Changi is reduced, thus requiring slot re-allocation. Information sharing is encapsulated in the state observation of the Reinforcement Learning agent. In this paper, we treat the terms "airport slot allocation" and "airport slot re-allocation" equally, which refer to the task of assigning a later slot for flight movements to balance between demand and capacity of the time slots.

Our main contribution is validating the use of Reinforcement Learning against different decision-making policies, which is delivered through the following subtasks:

- We develop a learning environment for the agent to learn to reallocate slots under reduced capacity scenarios in a multi-airport system consisting of different decision-making policies.
- We provide a suitable model of the Reinforcement Learning agent, which consists of the state observations, the reward designs, and the parameter tuning as well as algorithm selection, that can achieve comparable performance with the challenging Nearest Heuristic.
- We analyze the results, for both the training and testing phases to study the learned policy of the agent. We further provide illustrations of how the Reinforcement Learning agent captures the underlying fixed policy to surpass the performance of the Nearest Heuristic.

2. Learning environment

2.1. Problem overview

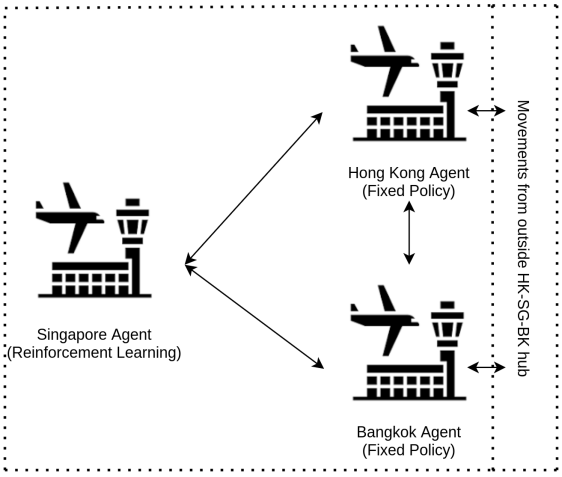

Fig. 1. Interactions between the airport agents in the Hong Kong-Singapore-Bangkok hub.

To apply Reinforcement Learning, the agent must have a learning environment to interact with. In our problem, the learning environment needs to simulate interactions between airports following certain policies. Figure 1 shows the interactions between the airports in the system. In this paper, we consider the Hong Kong-Singapore-Bangkok airport hub. The Changi (Singapore) airport agent uses Reinforcement Learning to form a decision-making policy for slot re-allocation under reduced-capacity scenarios. The other two agents, which represent the Hong Kong and Bangkok airports, will have a fixed policy, that allows flights to be reallocated to the next time slot if necessary. We consider a scenario, where one runway is shut down at the Changi airport (Singapore), which leads to a reduction in the airport capacity. Before the capacity reduction, flight movements of the airports in the hub satisfied the capacity constraints at the corresponding airports. The capacity reduction leads to an imbalance between demand and capacity of the time slots at Changi airport. The Singapore agent, taking on the slot coordinator task, needs to coordinate with the airports in the hub to reallocate slots. If there is no coordination between the airports, the Singapore agent can simply choose the nearest departure/ arrival slot pairs with no capacity violation as the newly assigned time slots for the movements. However, the coordination between airports can help to achieve lower delay (displacement), as per an example given in figure 2. Therefore, the Singapore agent needs to learn a policy that can efficiently coordinate with other airports in the hub. We assume that airports with movements departing/ arriving from outside the hub will accept the movement adjustments if there is no capacity violation.

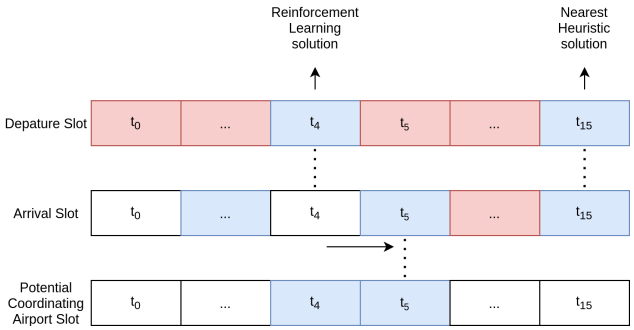

Fig. 2. A solution illustration: The Reinforcement Learning agent achieves lower displacement by capturing the policy of the arrival airport, which can assign a delay to one of its movements to create a slot for the RL agent. (Red: Over-capacity slot, White: Capacity of the slot equals 0, Blue: Under-capacity slot).

2.2. Generated scenarios

At the beginning of each episode, a scenario, that consists of the flight movements and the airports' capacity, is distributed randomly by the learning environment. To create the scenarios, we use the 2018 OAG data, which consists of flight movements of the calendar year. Each day of the dataset forms a separate scenario. For the airport capacity, the capacity of the Changi airport is assumed to be either 4

movements per 5-minute time slot (heavy-reduced capacity) or 5 movements per 5-minute time slot (medium-reduced capacity), which is in accordance with the case of a one-runway shutdown. Other airports' capacity is assumed to be fixed across a scenario. Further details of the airport capacity of the airports will be explained later.

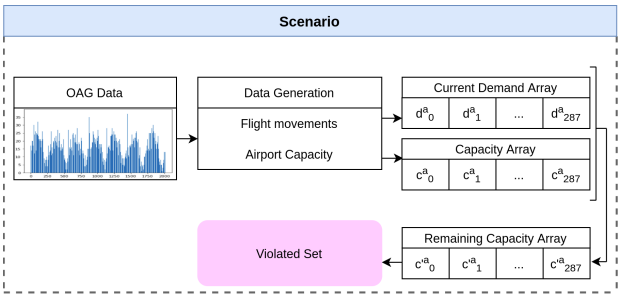

Fig. 3. An overview of a scenario and its encapsulated data.

Figure 3 shows the overview of a scenario distributed by the learning environment. We obtain the flight movements from the 2018 OAG data. Each flight movement can be either a departure or an arrival movement. A flight movement $m \in M$ is a tuple $(a_{dep}, t_{dep}, a_{arv}, t_{arv})$, where $a_{dep}, a_{arv} \in A$ and $t_{dep}, t_{arv} \in T$. Each departure/arrival movement associated with a time slot $t \in T$ at airport $a \in A$ will add 1 unit to the current demand $d_t^a \in D_T^A$. The current demand d_t^a equals the total number of movements associated with the time slot t_a. The capacity of time slot t at airport a is denoted as c_t^a. The Reinforcement Learning will reallocate the departure movements at the Changi airport to avoid capacity violations. We only consider the departure movement since the airport holds full autonomy to delay a flight. Let $T_{violated} = \{t_S | d_t^S > c_t^S\}$ be the set of all violated time slots of the Singapore agent. If a movement m has $t_{dep} \in T_{violated}$, the movement is added to the violated set V. We provide below a summary of notations for the problem:

- $m = (a_{dep}, t_{dep}, a_{arv}, t_{arv}) \in M$: set of flight movements.
- $A = \{B, H, O, S\}$: set of considered airports denoted by a, encoding the level-3 airports at Bangkok, Hong Kong, other airports outside the hub, and Singapore, respectively.
- $T = \{0, 1, ..., 287\}$: set of 5-minute-interval time slots t
- C_T^A: set of capacity constraint c_t^a of time slot t at airport a
- D_T^A: set of current demand d_t^a for time slot t at airport a
- $T_{violated} = \{t_S | d_t^S > c_t^S\}$: set of all violated time slots of the Changi airport (Singapore).
- $V = \{r = (a_{dep}, t_{dep}, a_{arv}, t_{arv}) | t_{dep} \in T_{violated}\}$: set of unaccommodated movements.

2.3. Learning mechanism

The learning environment will distribute a scenario with the flight movements and the airports' capacity at the start of the episode. The capacity and the current demand of each time slot of an airport a are encapsulated in the form of arrays, where $Capacity\_Array^a = [c_0^a, c_1^a, ..., c_{287}^a]$ and current demand array $Current\_Demand\_Array^a = [d_0^a, d_1^a, ..., d_{287}^a]$. Since each day has 288 5-minute time slots, the arrays have a dimension of (1x288). The remaining capacity array $Remaining\_Capacity\_Array^a = [c'^a_0, c'^a_1, ..., c'^a_{287}]$ is derived from the capacity and the current demand, where $c'_t = c_t - d_t$. From the remaining capacity, we obtained the violated set V. At the beginning of each time step, an unaccommodated movement will be randomly picked from the violated set. It is reasonable to solve each movement one by one since this mechanism has been applied for heuristics,[14].[15] The chosen movement will carry the information, which is encapsulated as state observations, the inputs for making decisions. At each time step, there will be three cases after the agent takes action:

- Case 1: The newly assigned time slot of the chosen movement satisfies the capacity constraints of both the departure and the arrival airports.
- Case 2: The newly assigned time slot of the chosen movement satisfies the capacity constraints of the departure airport (Changi, Singapore) but not the arrival airport. However, if the arrival airport can make room for the new time slot of the chosen movement by delaying one of its movements based on its policy, then the chosen movement will be assigned to the new time slot.
- Case 3: If Case 1 and Case 2 are not satisfied, then the time slot of the chosen movement remains the same.

The current demand arrays, the remaining capacity arrays, and the violated set will be updated based on one of the above cases. At the end of each time step, the learning environment will provide feedback (reward) to the agent. The agent will update its model following the feedback. The whole process repeats until a stopping condition is triggered. The stopping condition of an episode is either the violated set is cleared or the predefined number of maximum steps per episode has been exceeded.

3. Reinforcement Learning model

3.1. Action space

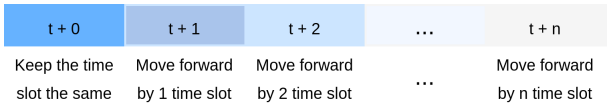

Fig. 4. Action space of the Reinforcement Learning agent.

Figure 4 shows the action space of the agent. The agent's actions are delaying a movement up to 12 time slots, i.e., move forward by n time slots ($+n$), where $n <= 12$, or keep the current time slot. Increasing the number of actions, i.e., the maximum displacement, is possible. However, there are two drawbacks. First, too many actions can affect the training efficiency.[16] Secondly, delaying a movement too much will not guarantee schedule acceptability.[13] Owing to the above

reasons, we impose the maximum displacement of 1 hour (12 time-slot displacement) by limiting the number of actions.

3.2. State observation

In Reinforcement Learning, the inputs for the agent to make decisions are encapsulated as state observations. The design of the state observation can affect the performance of the agent. Too many inputs can lead to inefficiency in training; not enough information, on the other hand, results in lower performance or not achieving model convergence. As per the figure 2, the agent at least needs to have information regarding the demand and capacity of the corresponding departure/ arrival time slots of the distributed movement. For better coordination, the agent also needs information on the potential movements, which the arrival airport can assign delays to accommodate the shifted movements of the Reinforcement Learning agent. In summary, the agent needs three pieces of information, information on the time slots of the agent, information on the time slots of the associated arrival airport, and information on the time slots of the potential coordinating airport.

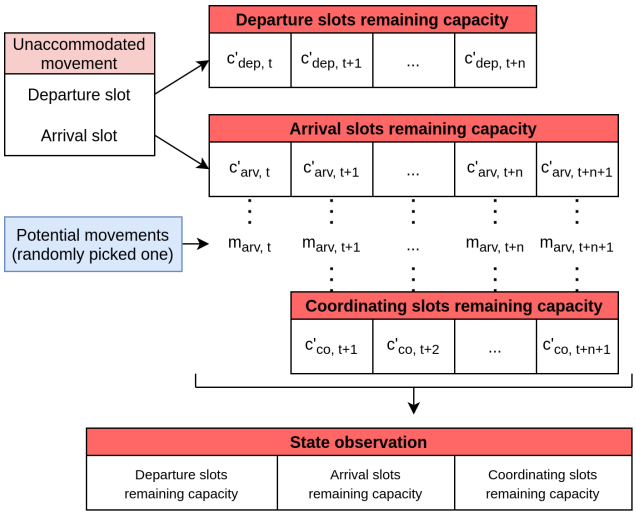

Fig. 5. State observation of the Reinforcement Learning agent.

Figure 5 shows the design of the state observation. Since we assume fixed capacity per scenario, both the capacity and demand information can be reflected in the remaining capacity c'_t. It is intuitive to provide the agent with the remaining capacity of all 288 time slots. However, such a design will result in a huge state observation of size $(288, 3)$, following the three pieces of information identified earlier. The abundance of irrelevant information results in inefficient training as the agent cannot identify which part of the state observation contributes to the outcomes. For example, it is not necessary to have information on the time slot 200 when the current investigating movement is assigned at time slot 100. In this design, we only provide the agent with the remaining capacity of the time slots associated with the actions of the agent. However, since the arrival airport policy involves delaying one of its movements to the next time slot, we additionally provide $c'_{arv,t+n+1}$. The arrival airport information involves $n+1$ time slots. Each time slot t holds different movements $m_{arv,t}$, which can be shifted following the fixed policy to make room for the currently unaccommodated movement. The number of movements per time slot is dynamic. Considering all potential coordinating movements leads to a complicated, yet inefficient, state observation design. Thus, we randomly pick one potential coordinating movement in each time slot and provide the agent with the remaining capacity of the associated time slot of the coordinating airport, $c'_{co,t}$. If there is no potential coordinating movement in a time slot, then $c'_{co,t} = 0$.

3.3. Reward design

The reward design plays an important role in ensuring the Reinforcement Learning agent can learn an efficient policy.[17] We want the agent to learn to assign new time slots so that there is no capacity violation and to minimize the displacement. We transfer these objectives into the following reward components:

- $R_{local} = z * (-|t - t_m|)$, where t is the newly assigned time slot and t_m is the original time slot of the movement. This reward component discourages the agent from displacing the movement further from the original time slot. The constant number z normalizes the reward to a smaller number to ensure training efficiency. The agent receives this reward component at every time step.
- $R_{solving} = +s$, where s is a constant number, is given to the agent if the considered movement is cleared out of the violated set. This reward component encourages the agent to resolve the capacity violations. We balance between the $R_{solving}$ and the R_{local} by letting the $R_{solving} = z * (n + 1)$ Balancing between the two components is important to avoid the agent's behavior of prolonging the episode by creating more unaccommodated movements to earn more $R_{solving}$.
- $R_{time\_step} = -p$, where p is a constant number, is given to the agent at every time step. This reward component encourages the agent to solve the problem faster and prevent any undesired behavior of prolonging the episode.

4. Experiments

4.1. Training scenarios

We use the OAG data for flight movements. Figure 6 demonstrates a distribution of the movements across 288 time slots in a single day at Changi airport. The OAG data is the allocated movements satisfying the capacity constraints at the strategic phase. If there is no unexpected event causing capacity reduction, we can assume the capacity based on the obtained movements. Thus, besides the Changi airport, we assume the capacity of a time slot of other airports equals the number of movements in that time slot plus either 0 or 1, i.e., $c'^{a}_{t} \in \{0, 1\}, \forall a \in \{B, H, O\}, \forall t \in T$. This assumption creates dense scenarios, with limited capacity, making

it more challenging for the agent to re-allocate slots. For the Changi Airport (Singapore) capacity, we assume a capacity reduction due to one-runway closure. The capacity of the airport is assumed to be fixed $c^S_t = 4$ (heavy-reduced capacity) and $c^S_t = 5$ (medium-reduced capacity) $\forall t \in T$. We perform training with medium-reduced capacity scenarios; and perform testing with two types of scenarios to analyze the performance of the Reinforcement Learning agent on unseen scenarios.

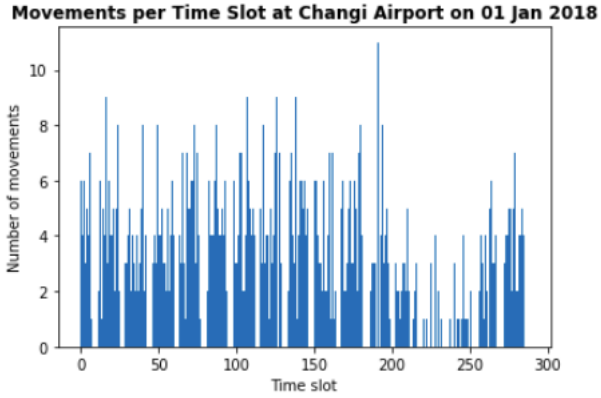

Fig. 6. The number of movements per time slot at Changi airport on 01 Jan 2018.

4.2. Learning algorithms and hyper-parameters

We adopt the Deep Q-Network (DQN) as the learning algorithm since DQN is well-suited for discrete-action problems. Experience replay and target network techniques are also incorporated since they are beneficial for data utilization and training stability. Details of the algorithm can be found in.[18)]

The hyper-parameters are discount rate 0.99, learning rate 0.0001, target update cycle 10000, and number of training steps 1000000. The number of actions n is 12, i.e., a maximum displacement of 60 minutes per movement. The z in R_{local} is 0.1. The p in R_{time_step} is 1.3 to offset the reward obtained from solving a movement. This will help to eliminate any behavior of creating more unaccommodated movements to gain higher rewards. Lastly, The number of maximum steps per episode equals twice the number of unaccommodated movements per episode.

4.3. Convergence analysis

Figure 7 and figure 8 show that the DQN agent is able to achieve convergence evidenced by the increasing trend in the average episode reward and the decreasing trend in the average episode length. These trends prove that the agent can solve faster, ie., shorter episode length, and allocate slots more efficiently, i.e., higher rewards. The average number of unaccommodated movements per episode is 15.03; which suggests that there is still room for the agent to improve the performance since the current average episode length is 23.47. For an episode with 15 unaccommodated movements, the expected optimal performance after converging can be approximately 10.5, with an episode length of 15. Increasing the number of training steps can further improve the performance of the agent.

Fig. 7. Episode reward convergence of the Deep Q-Network (DQN) model.

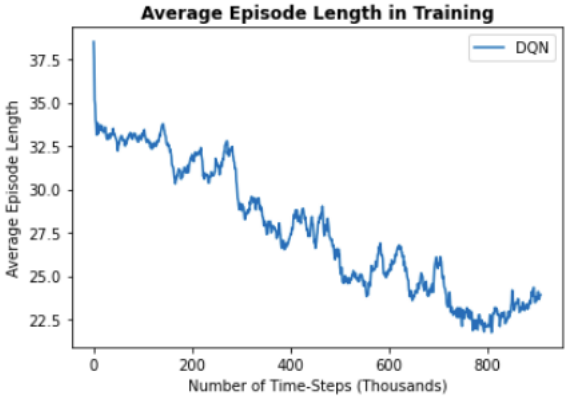

Fig. 8. Episode length convergence of the Deep Q-Network (DQN) model.

4.4. Performance testing

We use these metrics for performance comparison:

- Total delay $M_1 = \sum_{m \in M} |t - t_m|$.
- Maximum displacement across all movements $M_2 = \max_{m \in M} |t - t_m|$.
- Number of unaccommodated movements $M_3 = u$.
- Average displacement per movement $M_4 = M_1/(U - M_3)$, where U is the number of unaccommodated movements at the beginning of each episode.

We validate the performance of the DQN agent with the Nearest Heuristic. Results from figure 9 suggest that the Reinforcement Learning agent can learn to assign to the nearest time slot, under a medium-reduced capacity scenario. Since under a medium-reduced capacity scenario, there are still many possible nearest available slot pairs, there are few chances for the agent to achieve Case 2 (introduced earlier in section 2.3). Therefore, the best possible solution would be approximately the same with the Nearest Heuristic. There are 15 unaccommodated movements and in most cases, the Reinforcement Learning agent can assign a delay equal to the delay of the Nearest Heuristic. In summary, the total delay of the RL agent is 45, while the total delay of the

Nearest Heuristic is 41. However, the RL agent is able to maintain a lower maximum displacement across all movements, which is 10, compared to 11 of the Nearest Heuristic. Both the RL agent and the Nearest Heuristic solve all the unaccommodated movements.

Figure 10 suggests that in a heavy-reduced capacity scenario, the agent can make use of coordination to reduce delay. For movements where the nearest available slot pair is far away from the currently considered slot (20-time-slot displacement), the RL agent can achieve a significantly lower delay (with 3 and 9). In summary, the total delay of the RL agent is lower, with 84, compared to 107 of the Nearest Heuristic. The maximum displacement per movement is also lower for the RL agent, with 11, compared to 20. For the number of unaccommodated movements of the Nearest Heuristic, we only take into account the delay within 12 time slots, to ensure a fair comparison with the RL agent. In this sense, the RL agent also achieves a lower number of unaccommodated movements, with 3, compared to 5 of the Nearest Heuristic. Consequently, the average displacement per movement of the RL agent is also lower, with 3.82, compared to 4.86 of the Nearest Heuristic.

	Medium-reduced capacity (c = 5)	
	RL Agent	Nearest Heuristic
	1	1
	2	2
	1	1
	2	2
	5	4
	3	3
	1	1
	1	1
	1	1
	2	2
	2	2
	1	1
	7	4
	5	5
	10	11
M1	45	41
M2	10	11
M3	0	0
M4	3	2.73

Fig. 9. A solution for the medium-reduced capacity scenario (Light yellow: Equal delay, Dark yellow: Lower delay).

5. Conclusion and Discussion

We provide a Reinforcement Learning formulation considering a multi-airport system with different decision-making policies. The results suggest that the DQN agent can learn from the interactions between airports to reduce delays. With a higher capacity reduction, the RL agent can surpass the Nearest Heuristic, by capturing the interactions between airports, to achieve Case 2, resulting in lower delay. There is still room for improvement by extending the number of training steps. In the future, we will expand the problem to include more airports with additional policies and find more informative features to include in the state observation to improve the performance.

In the case of multiple airports using multiple RL-based policies, whether the DQN learning process is still expected to converge is not guaranteed due to two main reasons: 1) The non-stationary happens due to multiple learning agents continuously changing their policies, and 2) The difference in the objectives of the agents. With regard to the first reason, this is a well-recognized problem in Multi-Agent Reinforcement Learning.[9)19)] The state transition and the reward function perceived by each agent depend on the actions of other agents. The actions of the agents depend on their policies, which are not fixed during the learning process. This is called the non-stationary problem caused by the changing behavior of the agents. As for the second reason, different airport agents with RL-based policies may have different sets of objectives, which can conflict with each other. The agents may learn to counter other agents' policies to maximize their rewards. Different techniques to deal with the above issues such as Centralized Training Decentralized Execution

	Heavy-reduced capacity (c = 4)	
	RL Agent	Nearest Heuristic
	1	1
	2	2
	1	1
	11	11
	4	7
	2	2
	3	3
	3	20
	9	20
	1	1
	5	5
	1	4
	3	1
	1	1
	5	5
	6	4
	10	3
	2	2
	4	4
	1	1
	6	6
	3	3
M1	84	107
M2	11	20
M3	3	5
M4	3.82	4.86

Fig. 10. A solution for the heavy-reduced capacity scenario (Light yellow: Equal delay, Dark yellow: Lower delay).

(CTDE), self-play, stabilizing experience replay, opponent modeling, meta-learning, and communication.[19)]

In this problem, we consider an RL-based agent against other agents. Heuristics and learning-based methods are preferred over exact methods.[4)] While heuristics depend on a fixed set of rules, which is not adaptable to new scenarios, learning-based methods can learn from new experiences to improve the policies. In the future, we can study how multiple RL-based airport agents may interact with each other.

Acknowledgments

Anh Nguyen-Duy is supported by the NTU-Vingroup Graduate Scholarship for his doctoral study, under the supervision of Professor Vu Duong.

References

1) Corrigan S, Mårtensson L, Kay A, Okwir S, Ulfvengren P, McDonald N. Preparing for Airport Collaborative Decision Making (A-CDM) implementation: an evaluation and recommendations. Cognition, Technology & Work. 2015;17:207-18.

2) International Air Transport Association (IATA), Worldwide Airport Slot Guidelines (WASG), 2024, https://www.iata.org/en/programs/ops-infra/slots/slot-guidelines/.

3) Katsigiannis, Fotios and Zografos, Konstantinos.: A Multi-Objective Genetic Algorithm for Airport Slot Allocation Decision-Making, 2024, available at SSRN 4742975.

4) Zografos KG, Madas MA, Androutsopoulos KN. Increasing airport capacity utilisation through optimum slot scheduling: review of current developments and identification of future needs. Journal of Scheduling. 2017;20:3-24.

5) Zhang R, Prokhorchuk A, Dauwels J. Deep reinforcement learning for traveling salesman problem with time windows and rejections. In2020 International Joint Conference on Neural Networks (IJCNN) 2020 (pp. 1-8). IEEE.

6) Wang Y, Liu H, Zheng W, Xia Y, Li Y, Chen P, Guo K, Xie H. Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. IEEE access. 2019;7:39974-82.

7) Ali H, Pham DT, Alam S, Schultz M. A deep reinforcement learning approach for airport departure metering under spatial–temporal airside interactions. IEEE Transactions on Intelligent Transportation Systems. 2022;23(12):23933-50.

8) Chen Y, Xu Y, Hu M. General multi-agent reinforcement learning integrating heuristic-based delay priority strategy for demand and capacity balancing. Transportation Research Part C: Emerging Technologies. 2023;153:104218.

9) Gronauer S, Diepold K. Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review. 2022;55(2):895-943.

10) Chen Y, Xu Y, Hu M, Yang L. Demand and capacity balancing technology based on multi-agent reinforcement learning. In2021 IEEE/AIAA 40th digital avionics systems conference (DASC) 2021 (pp. 1-9). IEEE.

11) Xu Y, Prats X, Delahaye D. Synchronised demand-capacity balancing in collaborative air traffic flow management. Transportation Research Part C: Emerging Technologies. 2020;114:359-76.

12) Duong T, Todi KK, Chaudhary U, Truong HL. Decentralizing air traffic flow management with blockchain-based reinforcement learning. In2019 IEEE 17th international conference on Industrial informatics (INDIN) 2019 (Vol. 1, pp. 1795-1800). IEEE.

13) Zografos KG, Androutsopoulos KN, Madas MA. Minding the gap: Optimizing airport schedule displacement and acceptability. Transportation Research Part A: Policy and Practice. 2018;114:203-21.

14) Ribeiro NA, Jacquillat A, Antunes AP. A large-scale neighborhood search approach to airport slot allocation. Transportation Science. 2019;53(6):1772-97.

15) Wang S, Drake JH, Fairbrother J, Woodward JR. A constructive heuristic approach for single airport slot allocation problems. In2019 IEEE Symposium Series on Computational Intelligence (SSCI) 2019 (pp. 1171-1178). IEEE.

16) Yehudai A, Choshen L, Fox L, Abend O. Reinforcement learning with large action spaces for neural machine translation. arXiv preprint arXiv:2210.03053. 2022.

17) Dayal A, Cenkeramaddi LR, Jha A. Reward criteria impact on the performance of reinforcement learning agent for autonomous navigation. Applied Soft Computing. 2022;126:109241.

18) Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S. Human-level control through deep reinforcement learning. nature. 2015;518(7540):529-33.

19) Papoudakis, Georgios and Christianos, Filippos and Rahman, Arrasy and Albrecht, Stefano V. Dealing with non-stationarity in multi-agent deep reinforcement learning. arXiv preprint arXiv:1906.04737. 2019.