공학석사학위논문

A Reinforcement Learning Approach for the Aircraft Recovery Problem in Cases of Temporary Closures of Airports

강화학습을 이용한 공항 임시폐쇄 상황에서의 항공 일정계획 복원

2021 년 2 월

서울대학교 대학원

산업공학과

이 준 혁

A Reinforcement Learning Approach for the Aircraft Recovery Problem in Cases of Temporary Closures of Airports

강화학습을 이용한 공항 임시폐쇄 상황에서의 항공 일정계획 복원

지도교수 문 일 경

이 논문을 공학석사 학위논문으로 제출함

2020 년 12 월

서울대학교 대학원

산업공학과

이 준 혁

이준혁의 공학석사 학위논문을 인준함

2021 년 1 월

위 원 장 이 재 욱 (인)

부위원장 문 일 경 (인)

위 원 이 경 식 (인)

Abstract

A Reinforcement Learning Approach for the Aircraft Recovery Problem in Cases of Temporary Closures of Airports

Junhyeok Lee
Department of Industrial Engineering
The Graduate School
Seoul National University

An airline scheduler plans flight schedules with efficient resource utilization. However, unpredictable events, such as the temporary closure of an airport, disrupt planned flight schedules. Therefore, recovering disrupted flight schedules is essential for airlines. We propose Q-learning and Double Q-learning algorithms using reinforcement learning approach for the aircraft recovery problem (ARP) in cases of temporary closures of airports. We use two recovery options: delaying departures of flights and swapping aircraft. We present an artificial environment of daily flight schedules and the Markov decision process (MDP) for the ARP. We evaluate the proposed approach on a set of experiments carried out on a real-world case of a Korean domestic airline. Computational experiments show that reinforcement learning algorithms recover disrupted flight schedules effectively, and that our approaches flexibly adapt to various objectives and realistic conditions.

Keywords: Aircraft recovery problem, Airport closures, Swapping aircraft, Reinforcement learning, Q-learning, Double Q-learning

Student Number: 2019-20784

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As international trade and demand for air travel have increased, the number of commercial airlines has grown [20]. In order to survive in the competitive airline industry, airlines have tried to provide better service to passengers and use resources, such as crew and aircraft, efficiently. Planning flight schedules with effective utilization of such resources contributes to the overall success of an airline. Therefore, airlines spend considerable time and effort planning flight schedules [11].

Although efficient flight schedules are established, perturbations of flight schedules can occur due to uncertain or unpredictable events, such as adverse weather conditions, mechanical malfunctions, airport congestion, and crew member absences. Initial flight delays could propagate to subsequent flights due to interconnected resources (i.e., late arrivals of previous flights causes late departures of subsequent flights). In addition to this, flight delays not only affect passenger satisfaction but also cost airlines billions of dollars. The Federal Aviation Administration (FAA) estimates that the cost of flight delays in the United States costs airlines about $22 billion a year [36]. In order to alleviate minor stochastic delays and small disruptions, airline schedulers usually add buffer times between flight legs [26]. However, in cases of extreme disruptions (e.g., temporary closures of airports), buffer times

cannot prevent long flight delays. In particular, short-haul flight schedules with tight turnaround times cause serious damage to airlines' bottom lines. Turnaround time indicates the time interval on the ground needed to prepare aircraft for subsequent flights.

In order to minimize the damage from disruptions as much as possible, the Airline Operational Control Center (AOCC) initiates a recovery process of airline schedules that involves rescheduling aircraft, crews, and passengers [7]. Several research studies have focused on the integrated recovery of aircraft, crews, and passengers [35, 33, 9, 54]. However, due to the complexity of the airline recovery process, the airline usually segments the process into three stages: aircraft, crew, and passenger recovery [26]. Because aircraft is one of the most valuable resources for an airline, aircraft recovery is typically initiated at the first stage. In this stage, the AOCC reschedules the flight schedule and reroutes the affected aircraft to best meet the objectives of airlines. After aircraft recovery, crew planners reassign crews to aircraft according to the revised flight schedule (i.e., crew recovery). Finally, airline customer service coordinators accommodate misconnected passengers with their best alternative itineraries (i.e., passenger recovery). Among these three stages, this thesis concentrates on the aircraft recovery process.

In the aircraft recovery process, the AOCC makes decisions to restore flight schedules back to initially planned schedules through the following recovery options: canceling flight legs, swapping aircraft, ferrying, and delaying flight departures until connected resources become ready [7]. Especially, swapping aircraft is commonly used in the aircraft recovery process, and it is defined as switching flights on a pair of aircraft. In daily schedules, each aircraft is assigned to a sequence of flights (i.e.,

aircraft routes) [5]. Therefore, swapping aircraft is equivalent to swapping the routes of aircraft. This means that swapped aircraft have to complete each other's remaining flights. On the other hand, because cancellations and ferrying cause airlines great financial loss, they are seldom used in real practice [19]. When schedule disruptions occur, the problem of revising routes for affected aircraft using the previous options is known as the aircraft recovery problem (ARP).

Most of the existing research for the ARP focuses on minimizing the total cost incurred by flight delays. However, the total cost of delays is very sensitive to cost parameters, and estimating the values of parameters is challenging. Furthermore, there are many objectives for aircraft recovery processes, and each airline's objective could be different (e.g., minimizing total delays or the number of flight delays). In particular, minimizing the number of flights delayed over a specific time frame is very important to airlines. The number of flight delays is the key measure of the on-time performance of airlines. Not only do flight delays affect an airline's reputation, but airlines also must pay monetary compensation to passengers who have been on delayed flights. The criterion for defining a flight delay is different from country to country, and varies depending on whether the flight is international or domestic. For example, the FAA defines a flight delay as occurring when the departure or arrival of flights is 15 minutes later than its scheduled time. On the other hand, according to regulations set by the Korean government, flight delays are defined as occurring when the scheduled time of a flight is more than 30 minutes late. Not many studies have been carried out to deal with the objective of minimizing flight delays, except for the study by Liu et al. [31].

Liu et al. [31] studied the ARP by considering the number of flight delays of

more than 30 minutes. In order to get admissible Pareto optimal solutions, this study proposed the MMGA to consider multiple objectives using the method of inequalities (MOI). Minimizing the number of flight delays was just one of the objectives in the array of multiple objectives. A serious weakness of this study, however, was that this algorithm includes indispensable conditions (i.e., minimum turnaround time and flight connection at airports) to meet multiple objectives for hard constraints. Therefore, it was difficult to consider complex realistic conditions, such as multiple fleets and the aircraft balance. This was the case, because as the number of objectives increases, challenges of the computation burden and conflicts between objectives could appear. In addition to this weakness, since this study adopted MOI for finding Pareto solutions with smaller computing efforts, the MMGA found the suboptimal solutions.

To overcome the limitations of previous studies, we adopt the reinforcement learning approach, which has generated a lot of interest from the research community. We interpret the ARP as a sequential decision-making process and improves the behavior of the agent by trial and error. There are three main advantages to using reinforcement learning for the ARP. First, because reinforcement learning is a simulation-based method, it can handle complex assumptions [15]. In air transport management, there are many realistic conditions and factors that affect airline operations. By including these conditions in the simulation (i.e., environment), reinforcement learning could solve the ARP under realistic situations. Second, by just modifying reward functions, reinforcement learning is more flexible in various objectives than operations research methods. For example, reinforcement learning can easily adapt to the objective of minimizing the number of flight delays of more

than 30 minutes, compared to the existing approach, time-line networks. Due to the property of sequential decision making, reinforcement learning approach can simply calculate the number of flights delayed more than 30 minutes. However, in the case of time-line networks, plenty of delay arcs should be created, and additional constraints should be necessary in the mathematical model. Third, since reinforcement learning is an agent-based model, the policy that the agent learned for a case of disruptions can be reused for other cases of disruptions. Reusing the learned policy could accelerate the learning process compared to learning the policy from scratch [39].

To the best of our knowledge, there has been no experimental study of the ARP using reinforcement learning approach. In this thesis, our purpose is to develop the framework applying reinforcement learning approach to aircraft recovery. Specifically, we adopt Q-learning and Double Q-learning for the reinforcement learning algorithms. The proposed framework could support airlines handling schedule perturbations caused by airline disruption. Among various types of disruptions, we deal with the temporary closure of airports, which affect numerous flight operations. In addition, we solve this problem when multiple fleets serve the flight schedule. We utilize a real-world flight schedule and establish various objectives to meet each airline's goals. In summary, the contributions of this thesis are fourfold. First, we propose reinforcement learning algorithms for solving the ARP for the temporary closure of the airport. Existing studies of the ARP utilizes optimization or heuristic methodologies. This thesis, however, adopts reinforcement learning approach, which is an agent-based model. As far as we know, it is the first study to use reinforcement learning algorithm for the ARP. Second, we solve the ARP to optimize various

objectives: minimizing total delays and the number of flight delays of more than 30 minutes and more than 0 minutes. By revising the reward function, we can easily adapt our approach to different objectives compared to optimization or heuristic methodologies. Third, we compare the performance of the proposed reinforcement learning algorithms and the existing algorithm. The policy that the agent learns for aircraft recovery outperforms the existing algorithm for every objective. Fourth, we apply the proposed method to a real-world flight schedule of a Korean domestic airline with multiple fleets. In addition, we consider the constraint that ground handling service could be affected by disruption (i.e., turnaround time extensions).

Chapter 2

Literature Review

We investigated previous studies related to the ARP and a specific disruption that temporary closures of airports occur. The ARP was first introduced by Teodorović and Guberinić [41]. They solved a simple example of the ARP with a heuristic, which decided aircraft routes sequentially. Jarrah et al. [21] proposed two network flow models, one for the cancellation of flights and the other for re-timing flights. With the successive shortest path method, they obtained the solutions of network flow models. Cao and Kanafani [10] extended the method of Jarrah et al. [21], and they proposed a quadratic zero-one programming model. In addition to delays and cancellations, they used ferrying for recovery options. Yan and Yang [52] proposed a framework that is based on the time-line network. Moreover, they proposed time-shifted copies of planned flights in the event of flight delays. They solved this model by using Lagrangian relaxation with subgradient methods. Thengvall et al. [43] added protections arcs and through-flight arcs to the network model proposed by Yan and Yang [52]. This method makes it possible to prevent the original flight schedule from perturbations. Argüello et al. [3] proposed a heuristic method based on the greedy randomized adaptive search procedure (GRASP). Løve et al. [32] presented the steepest ascent local search (SALS) heuristic based on the network

formulation. This algorithm swaps aircraft iteratively until a better solution cannot be found. Rosenberger et al. [38] formulated a set partitioning model to reschedule flight legs and reroute multiple fleet aircraft. In order to efficiently determine a subset of aircraft to reroute, they developed the aircraft selection heuristic. Hu et al. [18] focused on a multi-objective aircraft recovery problem. They designed a heuristic for a large-scale recovery problem with three conflicting objectives. Liang et al. [28] solved the ARP with airport capacity constraint and maintenance flexibility. In order to consider these conditions, they adopted a column generation based heuristic framework. Moreover, in conditions of continuous flight delays, this algorithm optimized the problem more accurately than it did with discrete flight delays.

Several researchers have considered the temporary closure of airports in the ARP. Yan and Lin [51] conducted one of the pioneering studies on this problem. They adopted similar methods proposed by Yan and Yang [52]. They used a time-line network with four types of arcs: flight, ground, overnight, and ferrying. Thengvall et al. [45] proposed three multi-commodity network models for recovering a multiple fleet flight schedule when the hub airport closed. They showed that the preference network model obtained a better solution compared to other network models. This work was extended by Thengvall et al. [44]. They used a bundle algorithm to solve a Lagrangian relaxation of an integer programming formulation. Liu et al. [31] proposed a multi-objective genetic algorithm (MMGA) for short-haul flights in Taiwan. The objectives consisted of hard and soft constraints, and they used Pareto optimization for the multi-objective solution. The problem was decomposed separately according to each type of plane involved in the multiple fleet condition. Therefore, it is not easy to adopt MMGA to multi-fleet airline schedules. Liu et al. [30] utilized

a very similar method of the MMGA and studied it for a multi-objective solution, which included a delayed time variance.

Among various reinforcement learning algorithms, we utilize value-based learning. Q-learning is one of the most popular value-based reinforcement learning algorithm [49], and it has been applied in a large number of domains. In particular, it has been applied in many studies on the transportation area. Šemrov et al. [39] used Q-learning algorithm to reschedule single-track railway trains in Slovenia. Even though the total delays obtained by Q-learning and the first-in-first-out (FIFO) had no significant difference, Q-learning was effective in preventing 'deadlock,' the collision of trains. Jiang et al. [22] conducted research to develop a passenger inflow control strategy for relieving congestion at certain stations in peak hours by using Q-learning. In addition to relieving congestion, the proposed strategy minimized the safety risks of passengers.

Reinforcement learning approach was adopted in several studies for air transport management. Gosavii et al. [15] formulated airline revenue management as a semi-Markov decision process (SMDP). They solved SMDP with a λ-SMART algorithm based on reinforcement learning. Balakrishna et al. [4] incorporated reinforcement learning for predicting taxi-out time in airports. Because airline operations dynamically change, the accurate prediction of taxi-out time is very challenging. Due to this property, the authors adopted reinforcement learning, and the accuracy of their prediction was relatively high, even without detailed data. Hondet et al. [17] used Q-learning in disruption management of airline schedules. However, the performance of the Q-learning algorithm was worse than it was without any controls, and they obtained no significant results. To the best of our knowledge, there has been no

experimental study of the ARP using reinforcement learning approach. We propose reinforcement learning algorithms, specifically Q-learning and Double Q-learning, for the ARP in cases of temporary closures of airports. We solve this problem when multiple fleet serve the flight schedule. We utilize a real-world flight schedule and establish various objectives to meet each airline's goals.

The remainder of this thesis is organized as follows. Section 3 describes the problem statement and mathematical formulation of the problem. Principles of reinforcement learning, the environment of the flight schedule, and the Markov decision process (MDP) are presented in Section 4. Section 5 describes Q-learning and Double Q-learning algorithm. Section 6 shows the results of computational experiments to compare proposed reinforcement learning algorithms with the existing algorithms. Moreover, we evaluate the performance of our approaches by applying in the real-world case of the Korean domestic airline, and we suggest managerial insights. Conclusions are presented in Section 7.

Chapter 3

Problem statement

3.1 Characteristics of aircraft, flights, and flight schedule requirements

When the AOCC implements the aircraft recovery, the characteristics of the aircraft and flights should be considered. In actual practice, the AOCC usually swaps aircraft of the same 'subfleet' (i.e., category of aircraft types). Moreover, the minimum turnaround time of each aircraft type is different. The minimum turnaround time depends on the size of the aircraft. The bigger the aircraft, the longer the turnaround time. Considering the above characteristics, we set the following two constraints.

(1) Each aircraft can be swapped with the aircraft that is belonging to the same subfleet.

(2) The turnaround time of each aircraft type is different.

Flights on the schedule are implemented by designated aircraft. Each flight is assigned an origin airport, a destination airport, a flight number, a scheduled time of departure (STD), an actual time of departure (ATD), a scheduled time of arrival (STA), and an actual time of arrival (ATA). The STD and STA are determined in the initial flight schedule, and the ATD and ATA are determined after the event

of flight is implemented. If flight delays happen, the ATD or ATA is later than the STD or STA. Furthermore, the flying duration between an origin airport and a destination airport is fixed. Therefore, differences between the STA and STD, and the ATA and ATD are equivalent, unless a destination airport is closed during the time the aircraft planned to arrive is still in the air. In addition, we consider the following three requirements, which the flight schedule must satisfy.

(1) Minimum turnaround time: When an aircraft lands at the destination airport, a certain amount of times is necessary for preparing the next flight (e.g., runway taxiing, cabin cleaning, refueling, catering). This time is defined as turnaround time. The ground times of the aircraft for consecutive flights must be longer than a minimum turnaround time.

(2) Aircraft balance: The number of aircraft in each airport should be equivalent at the start and end of the day. Without the aircraft balance requirement, another disruption will occur the following day.

(3) Flight connection: In two consecutive flights of an aircraft, the destination airport of the prior flight and the origin airport of the subsequent flight should be the same.

3.2 Definitions of disruptions and recovery options and objectives of the problem

There are many types of flight disruptions, but we consider only one kind: the disruption that occurs when the airport is closed temporarily. Figure 3.1 represents the example of disruptions occurred by temporary closures of airports. When the airport is closed, flights planned to depart or arrive are postponed until the closed airport reopens. Because of this disruption, considerable delays occur in disrupted flights, and also subsequent flights could be affected by delay propagation. Moreover, the aircraft ground handling could take more time than usual due to airport congestion after closure. Therefore, the minimum turnaround time could be increased. To reflect this problem, we assumed that the turnaround time increased for a certain period of time at a closed airport, and we used the term 'turnaround time extension' to refer to this condition.

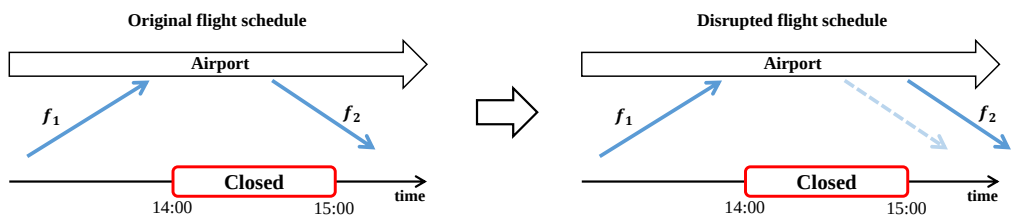

Figure 3.1: Disruptions by temporary closures of an airport

As mentioned in Section 1, there are four types of recovery options that minimize the damage of disruptions as much as possible. Except for cancellation and ferrying, the following recovery options are employed.

(1) Delaying flight departures: The departure times of subsequent flights should be delayed until the connected resources are ready (e.g., satisfaction for min-

imum turnaround time). This recovery option could cause delay propagation to subsequent flights in the route.

(2) Swapping aircraft: During the recovery process, the AOCC can change the aircraft for disrupted flights in order to absorb flight delays. This is defined as swapping aircraft. To illustrate the advantages of this option, we present Figure 3.2. The flight schedule consists of flights, $[f_1, f_2, f_3, f_4]$ and aircraft $[ac_1, ac_2]$. In the original planned schedule, f_1 and f_3 are assigned to ac_1, and f_2 and f_4 are assigned to ac_2. Assume that the disruption of the ATA on f_2 is equal to the STA of f_2, and assume that the ATA of f_1 is later than the STA of f_1. If f_1 and f_3 are assigned to ac_1 in accordance with the initial planned schedule, the departure time of f_3 should be delayed for satisfying the minimum turnaround time requirement. However, when f_2 and f_3 are reassigned to ac_2, and when f_1 and f_4 are reassigned to ac_1, the ATD and STD are equal for f_2, f_3, and f_4. With the aircraft swapping, the AOCC can avoid departure delays of f_3.

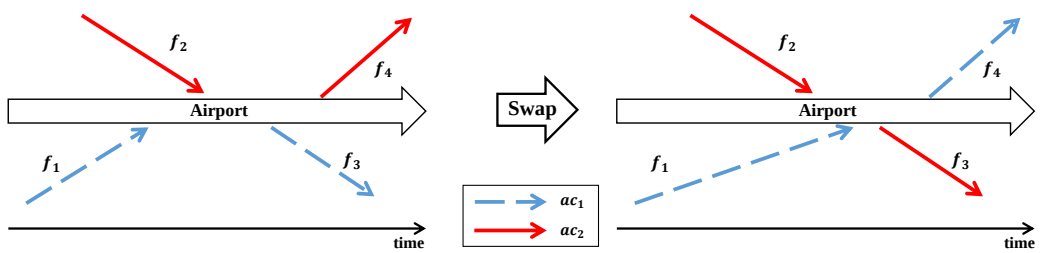

Figure 3.2: Example of swapping aircraft

Turning now to the objectives of the ARP, we adopt three cases with different system objectives by considering the characteristics of the air transport business. Among the two types of flight delays, we consider departure delay for all objectives except for one, in which we consider an arrival delay. The adopted objectives are as

follows.

(1) To minimize the total delays of flights (Case A): This objective ensures the minimization of total delays on the daily flight schedule. This is the most common objective that existing studies used.

(2) To minimize the number of flight delays of more than 30 minutes (Case B): Flight delay is a significant measure for evaluating an airline's on-time performance. If the time differences between actual and scheduled flight events exceeds the permissible range, it is defined as a 'flight delay.' The criterion of the permissible range is different in every country. Therefore, recovering flight schedules to meet flight delay criterion could be necessary for the reputation of the airline. We define flight delay when the actual event time is late over 30 minutes compared to the scheduled event time.

(3) To minimize the number of flight delays of more than zero minutes (Case C): This objective ensures the punctuality of airlines, which affects customer satisfaction, and brand image of the airline. Punctuality is one of the most important aspects that influences customer loyalty. Because of the properties of airline service, customers tend to be loyal to particular airline companies. These customers keep on using a specific airline rather than changing to others, and they influence a huge part of airlines' revenues [23].

(4) To minimize the number of flight delays of more than zero minutes (Case D): We set this objective to avoid considering minor flight delays. Most passengers who take flights for traveling do not consider the flight departure on time as a significant issue. Therefore, the minor delay (e.g., five minutes) cannot affect traveling passengers' satisfaction.

3.3 Assumptions

We accommodate similar assumptions in [31]. The assumptions are as follows:

(1) The flight schedule is a daily schedule. Therefore, the scheduled (initial) flight schedule always satisfies the aircraft balance requirement for that day.

(2) The time horizon to recover the disrupted flight schedule ranges from the start time of an airport's closure to the end of the day.

(3) The ATD of a flight cannot be earlier than the STD of a flight.

(4) The flying time between each airport is fixed.

(5) Every slot (e.g., landing and takeoff) is assumed to be available.

(6) The event times of flights planned to land on or depart from a closed airport are postponed until the reopening time.

(7) When the flight is planned to arrive at a closed airport during the closure and has not yet departed from the origin airport, the departure times of the flight must be postponed until the destination airport is reopened.

(8) Every flight in each aircraft route must take off from the destination airport of the immediately preceding flight.

(9) When the aircraft departed the terminating flight of the scheduled (initial) route, it cannot be swapped with other aircraft. This aircraft lands at the destination airport of the terminating flight of the initial route and finishes the daily schedule.

(10) The change in the sequence of flights in routes is not allowed after swapping aircraft.

Assumptions (1)-(8) are the same as those in [31], and Assumptions (9) and (10) are added for the following three reasons. First, Assumption (9) prevents any

aircraft from taking many flights due to the regulations of minimum crew rest and aircraft maintenance. Second, because of Assumption (1) and (9), aircraft balance requirement is satisfied. According to Assumption (1), if each aircraft takes the terminating flight of initial routes at the end of the daily schedule, the aircraft balance is satisfied. Because Assumption (9) ensures that the terminating flights of initial routes are carried out last by each aircraft, the aircraft balance requirement is satisfied. Third, Assumption (10) can reduce action space. In the early stage of this research, we did not account for Assumption (10), and the extensive action-state space that the agent visited caused extreme flight delays and slowed convergence speed.

Swapping 'arriving aircraft' (i.e., the aircraft assigned to the arrival event) with any aircraft could violate Assumption (10). Therefore, in this thesis, the standard meaning of 'swappable condition' is used to indicate the aircraft that satisfy Assumption (10). We assume that the decision of swapping aircraft is available at the arrival event of the aircraft. After swapping arriving aircraft with each aircraft in flight schedule, it is required to classify them according to whether they satisfy Assumption (10) or not. Let mta_c denotes the minimum turnaround time of aircraft c. We indicate dt_1 as the departure time of subsequent flight of arriving aircraft, and dt_2 as the earliest departure time among remaining flights except dt_1 ($dt_1 < dt_2$). At each arrival event, we generate the list of aircraft that are in the swappable condition using the following algorithm:

Algorithm 1 Generation the list of swappable aircraft

Initialize $S \leftarrow$ empty list
$c \leftarrow$ *arriving aircraft* that is assigned to the arrival event
$dt_1 \leftarrow$ earliest departure time of remaining flights of c
$dt_2 \leftarrow$ second earliest departure time of remaining flights of c
for *each aircraft* i **do**
  if i *is in the air* **then**
    if $\max\{$*planned arrival time of* $i + mta_i, dt_1\} < dt_2$ **then**
      S.*append*(i)
    end
  else
    if $\max\{$*the time* i *has arrived* $+ mta_i, dt_1\} < dt_2$ **then**
      S.*append*(i)
    end
  end
end

3.4 Mathematical formulations

The time-line network mathematical model, which is one of the most comprehensive and practical approaches for the ARP, is formulated based on network flow. A simple example of a time-line network is illustrated in Figure 3.3. There are three types of nodes: supply (S), termination (T), and intermediate (I) nodes. The larger nodes shown in Figure 3.3 are the supply and termination nodes. The supply nodes supply aircraft at the beginning of the time horizon of recovery, and aircraft finish flight schedule when the flow of aircraft is reached to the termination nodes. The smaller nodes represent intermediate nodes and indicate the departure or arrival event of flight at a specific airport and time. There are two types of arcs in the time-line network: ground (G) and flight arcs (N). The ground arcs represent the number of aircraft on the ground preparing for the next flights at the specific airport. The flight arcs consist of scheduled and delay arcs. The scheduled arcs represent original planned flight legs. For considering delays on a particular flight, several delay arcs are built to represent the series of options that move the departure time of flight for later times. In Figure 3.3, three delay options, 10, 20, and 40 minutes, are available for each of the three flights. Therefore, in this example, the set of arcs for each flight ($N(f)$) covers four flight arcs: one scheduled arc and three delay arcs.

Based on the problem defined in Sections 3.1 and 3.2, a mathematical formulation of time-line network model is developed. The notations used in the mathematical formulation are as follows:

Sets

F	set of flights
N	set of flight arcs (including scheduled and delay arcs)
G	set of ground arcs
S	set of supply nodes
T	set of termination nodes
I	set of intermediate nodes
P	set of subfleets
R	set of aircraft routes
$O^+(i,p)$	set of arcs originating at node i for subfleet p
$O^-(i,p)$	set of arcs terminating at node i for subfleet p
$N(f)$	set of flight arcs covering flight f; $N(f) \subset N$
$N(r)$	set of flight arcs belonging to the aircraft route r; $N(r) \subset N$

Parameters

c_{np}	delays incurred for flight arc n for subfleet p
b_{sp}	initial supply of aircraft at supply node s for subfleet p
b_{tp}	number of aircraft required at termination node t for subfleet p
u_{gp}	capacity for ground arc g for subfleet p
q_{np}	unique number of flight which contains flight arc n for subfleet p
t_{qp}	initial scheduled departure time of unique flight number q
$\tilde{t}_{np}$	actual departure time of flight altered by flight arc n for p

Decision variables

x_{np} flow on flight arc n for subfleet p

z_{gp} flow on ground arc g for subfleet p

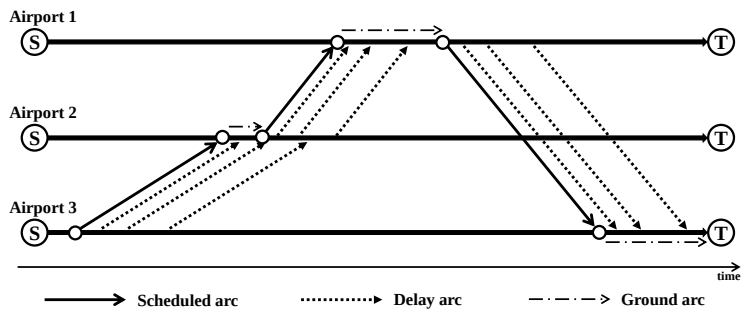

Figure 3.3: Simple example of time-line network structure

A single time-line network is utilized for a subfleet, as illustrated in Figure 3.3. In order to take into account a multi-fleet condition, we utilize $|P|$ number of time-line networks. The following is the mathematical formulation of the ARP. The model is developed based on the mathematical formulation proposed by Thengvall et al. [45].

$$\min \sum_{p\in P}\sum_{n\in N} c_{np}x_{np} \tag{3.1}$$

$$\text{s.t.} \sum_{g\in G\cap O^+(s,p)} z_{gp} + \sum_{n\in N\cap O^+(s,p)} x_{np} = b_{sp}, \quad \forall s\in S, \forall p\in P \tag{3.2}$$

$$\sum_{g\in G\cap O^+(i,p)} z_{gp} - \sum_{g\in G\cap O^-(i,p)} z_{gp} + \sum_{n\in N\cap O^+(i,p)} x_{np} - \sum_{n\in N\cap O^-(i,p)} x_{np} = 0, \quad \forall i\in I, \forall p\in P \tag{3.3}$$

$$\sum_{g\in G\cap O^-(t,p)} z_{gp} + \sum_{n\in N\cap O^-(t,p)} x_{np} = -b_{tp}, \quad \forall t\in T, \forall p\in P \tag{3.4}$$

$$\sum_{p\in P}\sum_{n\in N(f)} x_{np} = 1, \quad \forall f\in F \tag{3.5}$$

$$x_{ap} \le x_{bp}, \quad \forall a,b\in N(r), \forall r\in R, \forall p\in P, t_{q_{ap}} \le t_{q_{bp}}, \tilde{t}_{bp} < \tilde{t}_{ap} \tag{3.6}$$

$$x_{np}\in\{0,1\}, \quad \forall n\in N, \forall p\in P \tag{3.7}$$

$$0\le z_{gp}\le u_{gp}, \quad \forall g\in G, \forall p\in P \tag{3.8}$$

$$z_{gp}\in \mathbb{Z}, \quad \forall g\in G, \forall p\in P \tag{3.9}$$

The objective function (3.1) is minimizing the total flight delays incurred by disruption. The c_{np} is modified according to the defined objective. Constraints (3.2) and (3.4) represent the aircraft balance constraints. Constraint (3.3) is the balance equation for intermediate nodes. Constraint (3.5) is the flight cover constraint, which ensures that every flight is implemented at scheduled time or delayed. Constraint

(3.6) is the route sequence constraint, which ensures Assumption (10). Constraint (3.7) is a binary constraint. Constraint (3.8) is the capacity constraint for ground arcs. Constraint (3.9) ensures that the flow of ground arcs is integer.

There are three weak points in this model. First, without Constraints (3.5) and (3.6), this problem is equivalent to the single commodity flow problem, which is a well-solved problem. However, the single commodity flow problem becomes NP-hard problem by adding the side constraint, Constraint (3.5) [13]. By implementing computational experiments utilizing the optimization solver, we observed that the optimization solver could not get the best feasible solution within 10 minutes with Constraint (3.6), but could get the best feasible solution within 10 minutes without Constraint (3.6). Therefore, due to the fact that the time-line network model without Constraint (3.6) is not more difficult than the model with Constraint (3.6), we could infer that the time-line network model with Constraint (3.6) is also NP-hard problem. Second, because it is necessary to create delay arcs to account for delaying flight departure, a trade-off between quality of recovery schedule and computation time exists. The more delay arcs could guarantee better outcomes while the problem size increases [51]. Third, because the decision variables of this model represent the flow of flight and ground arcs of aircraft, an additional algorithm is required to transform the arc-based solutions to the route-based solutions.

Unlike the methods mentioned above, we interpret the ARP as a sequential decision-making and develop an MDP model for this problem. In addition, we utilize the reinforcement learning algorithms to find optimal policy in the MDP. In this manner, we can adopt delay options for every discrete-time (minutes). Also, constructing flows of aircraft and aircraft routing can be implemented simultaneously.

Chapter 4

Reinforcement learning for aircraft recovery

In order to apply reinforcement learning to aircraft recovery, a well-defined environment, agent, and MDP adequate to the given problem are significant. Section 4.1 describes the interaction between the agent and the environment, the action-value function, and the exploration-exploitation dilemma. Section 4.2 explains the structure of the environment of flight schedule operation. Section 4.3 presents the details of states, actions, and reward functions of MDP.

4.1 Principles of reinforcement learning

Reinforcement learning is an agent-based approach to find an optimal policy that would maximize cumulative rewards by trial and error in a given environment. Through interaction with the environment, the agent discovers the optimal or near-optimal action a_t at a specific state s_t. The reward r_t and the next state s_{t+1} are observed when the agent takes action a_t. By observing the reward signal, the agent can assess the quality of action. Figure 4.1 depicts how the agent interacts with the environment. In this thesis, the agent corresponds to the AOCC, which makes decisions for aircraft recovery; and the flight schedule system, which includes every aircraft, flight, airport, and timetable, is the environment.

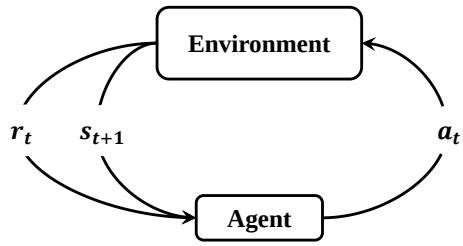

Figure 4.1: Framework of reinforcement learning

Because reinforcement learning is a framework for sequential decision making, we consider not only immediate reward r_t but also long-term rewards. In such a setup, the agent seeks to maximize the return, which is defined as the sum of future discounted rewards: $G_t = \sum_{k=n+1}^{\infty} \gamma^{k-t-1} r_k$. The policy is the agent's way of establishing behavior in a given situation. It can be defined as a mapping from states to probabilities of each action: $\pi(a|s)$. The action-value function $q_\pi(s, a)$ estimates the quality of taking an action a at the state s following policy π [40]. The action-value function $q_\pi(s, a)$ expects the return G_t starting with the state s, taking action a under policy π: $q_\pi(s, a) = E_\pi[G_t|s_t = s, a_t = a]$. The purpose of reinforcement learning is to find optimal policies (π^*) which share optimal action-value function: $q_{\pi^*}(s, a) = \max_\pi q_\pi(s, a)$.

Among valid actions, the agent takes action a_t at the state s_t depending on $q_\pi(s_t, a_t)$. If the agent always takes action with the maximal value of the action-value function, it is a suitable strategy to maximize return on the immediate step (exploitation). In order to produce better total rewards in the long term, however, it is necessary to choose other valid actions (exploration). This challenge is referred to as the exploration-exploitation dilemma. Various methods have been proposed to balance exploration and exploitation, and the ε-greedy is one of the most commonly

used strategies. The ε-greedy takes action a in accordance with the following policy:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|A(s)|} & if\ a = \text{argmax}_a\, Q(s,a) \quad (4.1) \\ \dfrac{\varepsilon}{|A(s)|} & otherwise \quad (4.2) \end{cases}$$

where $A(s)$ denotes the action space at state s. In addition to the ε-greedy, we adopt an optimistic initialization. The ε-greedy with optimistic initialization is effective on stationary problems [40]. This method biases the initial action-value estimates and ensures extensive exploration. In this thesis, we initialize action-value estimates to zero and employ negative rewards. Then, the reward is less than any starting estimates of action-value, causing extensive exploration in the early stage. We set parameter ε to 0.97^n where n means the number of the current episode. Therefore, ε, the probability of choosing an action for exploration, decays over time. The first learning episode could start with a big ε. However, ε converges to a small value with the learning process. Therefore, as the iterations of the episode increases, the agent chooses action for exploitation.

4.2 Environment

We built an artificial environment of flight schedule operation based on the assumptions in Section 3.3. Figure 4.2 represents the structure of the environment. There are three principal elements in the proposed environment. The first element is aircraft routes. There are two types of routes: the scheduled and actual route. Initially, every aircraft is assigned to the scheduled route. After a disruption, the AOCC reroutes aircraft by swapping each aircraft's scheduled routes, and the actual routes for aircraft are established at the terminal state. The second element is the aircraft. There are several states of aircraft that contain the status of each aircraft at a given time (e.g., flying status, assigned route, and location). Based on the states of aircraft, the valid actions are determined. The third element compromises events of flights. The database of the flight events is utilized, and it contains the information of the actual time of flight events, and whether a given flight is implemented or not.

In the initialization stage, the state of the simulation is updated based on the input instance: flight schedule and disruption information. The time step is defined as events of flights, and there are two events in a flight: departure and arrival. Among the departure or arrival events of flights not implemented yet, the earliest one to occur is first designated to the time step, and a single aircraft is assigned to the corresponding event. When the time horizon of recovery begins at the start of the day, one episode is finished after $2|F|$ time steps are executed, where $|F|$ be the set of flights. On the other hand, if the time horizon of recovery begins at the middle of a day, the number of time steps that have to be executed is less than $2|F|$. The terminal state of each episode is the time step when the latest arrival of a flight is executed. At the arrival event, the AOCC can decide which aircraft will swap with

arriving aircraft. If the AOCC decides not to swap aircraft, only the states of arrived aircraft are updated. Otherwise, if two aircraft are determined to swap their assigned routes, the states of those two aircraft are updated. At the departure event, if the ground time of 'departing aircraft' (i.e., the aircraft assigned to the departure event) does not satisfy the minimum turnaround time requirement, the AOCC delays the actual departure time until the minimum turnaround time is met.

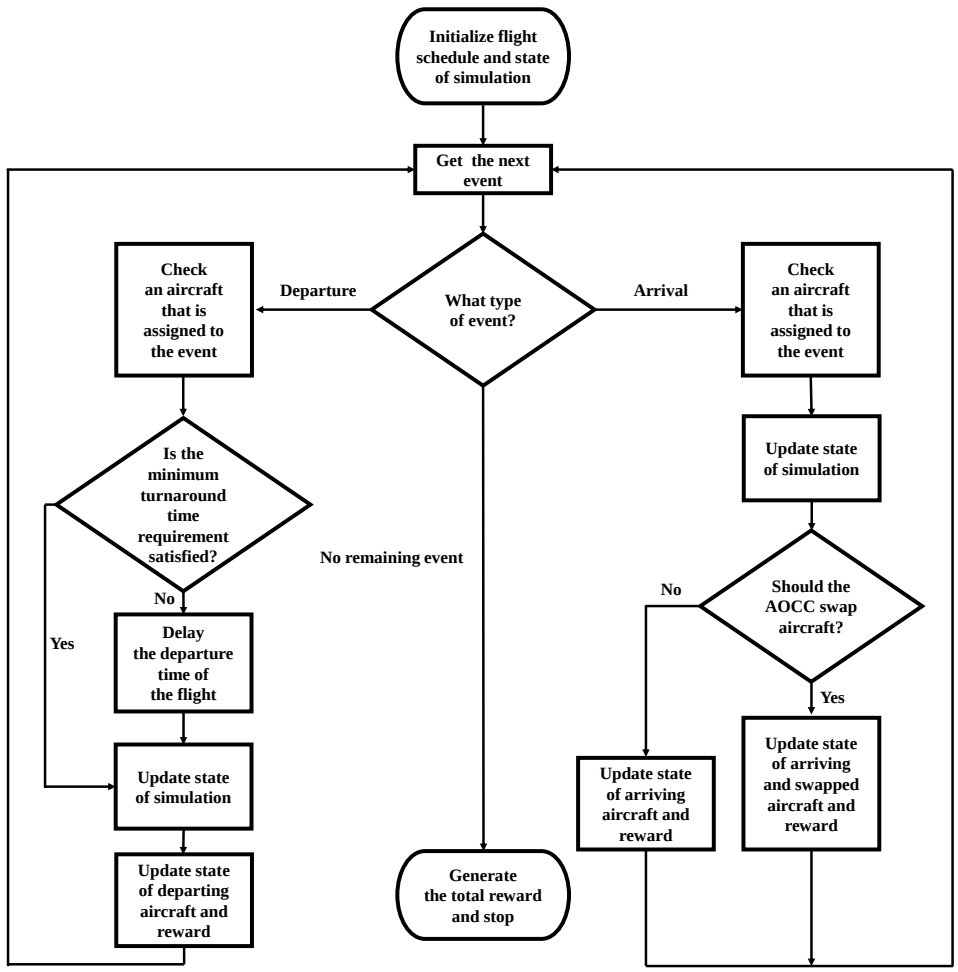

Figure 4.2: Structure of the environment

4.3 Markov decision process

The problem of reinforcement learning can be formalized as MDPs, which is a mathematical formulation for sequential decision making. An MDP is defined as a tuple (S, A, R, P, γ) that is composed of five components— a set of states, S; a set of actions, A; the reward function, R; the transition probability function, P; and the discount factor, γ.

In every time step, the agent receives states from the environment and takes action based on those current states. Among lots of information in the environment, the information that is essential for the learning agent is defined as the MDP's state. We combine the states of aircraft and the event information to define the state of MDP (s, i). The states of aircraft are as follows: future route, previous route, and binary parameter for indicating whether aircraft is flying or on the ground. Let $u_c \in U$, $p_c \in P$, and $b_c \in B$ denote future routes, previous routes, and the binary parameter of aircraft $c \in C$ where U, P, B, and C represent the set of future route, previous route, binary parameter, and aircraft in flight schedule, respectively. Because all aircraft in the flight schedule has this state, the state of aircraft, s, is defined as the following tuple: $s = (u_1, \cdots, u_{|C|}, p_1, \cdots, p_{|C|}, b_1, \cdots, b_{|C|})$

Because the number of routes is equal to the number of aircraft in the flight schedule, the set of previous and future route of each aircraft is as follows: $U_c = \{1, 2, 3, \cdots, |C|\}$, $P_c = \{1, 2, 3, \cdots, |C|\}$, $\forall c \in C$. The future route means the route that an aircraft is going to take for departing immediately after. This route is determined by the time point when an aircraft is planned to be assigned for the next departure. Because each aircraft is assigned one-on-one with the route, the size of future route space for all the aircraft is equal to $|C|!$. The previous route is defined

as the route that an aircraft took for departing immediately before. This route is determined by the time point when an aircraft was most recently assigned for the past departure. The size of the previous route space for all the aircraft is equal to $|C|^{|C|}$ because the assigned previous route of aircraft can be overlapped. When the aircraft c is on the ground, the binary parameter (b_c) is zero, otherwise one. Therefore, the size of this state space for all aircraft is equal to $2^{|C|}$.

The event information represents the type of event on every flight. Because there are two types of events in each flight, the size of state space of the event information is equal to $2|F|$, where F is the set of flights in the flight schedule. The set of states of the event information, I, is as follows: $I = \{1, 2, 3, \cdots, 2|F|\}$. The state of the MDP at given time step, t, is denoted as a tuple: (s_t, i_t), $\forall s_t \in S$, $\forall i_t \in I$. Therefore, the total size of the environment state space is equal to $(2|C|)^{|C|+1}|F||C-1|!$. Even though the size of state space is extensive, the actual visited state is relatively small due to the fact that the state is not dramatically rearranged at each time step. The maximum number of state elements that can be changed at each time step is five: i) the previous, and ii) the future route, and iii) the binary parameter of aircraft assigned to the event, and iv) the future route of swapped aircraft, and v) the event information. In addition, the overlapping of previous routes seldom occurs in the environment due to the time interval between flights.

In this thesis, the action set, A, is defined as all aircraft operating on the flight schedule. The action selected at each time step, t, is $a_t \in A$, which represents an aircraft swapped with the departing or arriving aircraft. The action space is defined as follows: $A = \{1, 2, \cdots, |C|\}$. If the action corresponds to the aircraft assigned to the given time step, it is equivalent to the condition of not swapping aircraft.

Recall from Section 4.2 that the AOCC can swap aircraft at the arrival event. In other words, only one action is valid at the departure event. Moreover, at the arrival event, the action space can be reduced due to the swappable condition. Figure 4.3 depicts how states are changed by swapping actions. At the first time step, Aircraft 1 is assigned to the arrival of Flight 1 (arriving aircraft). The AOCC decides to swap Aircraft 1 and Aircraft 2, which are scheduled to arrive at the same airport. Assuming that Aircraft 3 does not satisfy the swappable condition, Aircraft 1 can not swap with Aircraft 3. At the second time step, the future route states of Aircraft 1 and Aircraft 2 are changed, and the binary parameter state of Aircraft 1 is changed to zero due to the previous time step. At the third time step, because the departure of Flight 2 is assigned to Aircraft 1 at the second time step, the previous route and binary parameter state of Aircraft 1 is changed. The arrival of Flight 7 is assigned to Aircraft 2, and the AOCC decides not to swap. Therefore, only the binary parameter state of Aircraft 2 is changed to zero at the fourth time step.

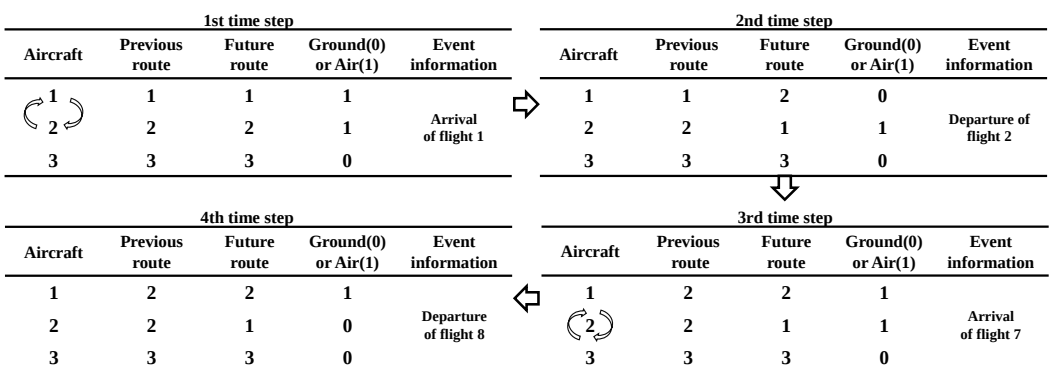

Figure 4.3: Example of states changed by the actions of the agent

The reward function defines the purpose of the agent and indicates what is a good or bad action at a specific state within the environment. Formulating a

reward function appropriate to the objectives of reinforcement learning problem is important for guiding the agent to achieve its goal. We define three reward functions in accordance with each objective. We indicate t_f^d as the STD, and $\tilde{t}_f^d$ as the ATD of flight $f \in F$. In Case A, reward r_t is the departure delay of flight n ($\min\{t_n^d - \tilde{t}_n^d, 0\}$). In Case B, if the departure delay of flight n is more than 30 minutes ($\tilde{t}_n^d - t_n^d > 30$), reward r_t is -1, otherwise 0. In Case C, if the departure delay of flight n is more than zero minutes ($\tilde{t}_n^d - t_n^d > 0$), reward r_t is -1, otherwise 0.In Case D, if the departure delay of flight n is more than five minutes ($\tilde{t}_n^d - t_n^d > 5$), reward r_t is -1, otherwise 0.

The discount factor, γ, determines how much emphasis is given to immediate rewards. If γ is close to 0, it means that the agent puts more importance on immediate rewards. On the other hand, there is more emphasis on the future rewards when γ is close to 1. The parameter setting of the discount factor will be explained in Section 6.

Chapter 5

Reinforcement learning algorithms

In this section, we present reinforcement learning algorithms for solving the ARP. Among the various reinforcement learning algorithms, we adopt Q-learning and Double Q-learning. Section 5.1 presents the concept and procedure of the Q-learning algorithm. Section 5.2 examines the overestimation bias problem of the Q-learning algorithm and presents the Double Q-learning algorithm, which alleviates the overestimation bias.

5.1 Q-learning algorithm

As was stated in Section4.1, the purpose of solving the MDP is that finding the optimal action-value function at the state and action pairs (s, a). Based on the classical dynamic programming, we can find the optimal action-value function by utilizing the Bellman optimality equation [6]. However, in problems involving complicated systems with numerous states, it is difficult to compute the values of the transition probability. This difficulty is called the 'curse of modeling' [14]. In contrast with the dynamic programming, model-free algorithms of reinforcement learning does not need to calculate the transition probability. Q-learning is the reinforcement learning algorithm utilizing the model-free concept and employs Bellman equation

and temporal difference learning [49]. Especially, temporal difference learning generalizes beyond the Robbins-Monro algorithm, which is a representative method in a stochastic approximation [37]. By approximating the optimal action-value, it is not necessary to compute the transition probability, and Q-learning can avoid the curse of modeling.

Due to the property of the off-policy approach, Q-learning utilizes two independent policies: a behavior policy and a target policy. The agent takes action a_t based on the value of the learned action-value function, Q function, and follows a behavior policy (i.e., ε-greedy policy) for s_t while learning with a target policy (i.e., greedy policy) for s_{t+1}. We initialize the Q function to zero for all states and actions at the first stage, and we update the Q function until the end of the learning. The current state of the Q function, $Q(s_t, a_t)$, is updated by the next states of the Q function, $Q(s_{t+1}, a')$, with the greedy action. The learned Q function approximates the optimal action-value by updating with the following equation iteratively:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)] \tag{5.1}$$

In this equation, α is the step size parameter of the learning rate of temporal differences δ_t. Utilizing the concept of temporal differences, the agent can avoid having to wait until the end of each episode, and can update the Q function immediately after it visits a pair of state and action. The Q function is updated with δ_t, depending on the size of the learning rate, α, which determines the step size in δ_t direction. Moreover, since our problem is a stationary problem, we compare the total reward of episode (ρ), which is a summation of observed rewards r_t in an episode, with

the maximum total reward of episode (ρ_{max}) and update the maximum value. The procedure of Q-learning algorithm is presented as follows:

Algorithm 2 Q-learning

Initialize $Q[(s,\, i),\, a]$, for all $s \in S$, $i \in I$, $a \in A$
$\rho_{max} \leftarrow 0$
for *each episode* **do**
  Initialize s and i
  $\rho \leftarrow 0$
  $t \leftarrow 0$
  while i_t *is not the terminal state* **do**
    Choose a_t using ε-greedy policy based on $Q[(s_t,\, i_t),\, a]$
    Take action a_t and observe r_t, $(s_{t+1},\, i_{t+1})$
    $\rho \leftarrow \rho + r_t$
    $\delta_t \leftarrow r_t + \gamma \max_{a'} Q[(s_{t+1},\, i_{t+1}),\, a'] - Q[(s_t,\, i_t),\, a_t]$
    $Q[(s_t,\, i_t),\, a_t] \leftarrow Q[(s_t,\, i_t),\, a_t] + \alpha\delta_t$
    $t \leftarrow t + 1$
  end
  $\rho_{max} \leftarrow \max\{\rho_{max}, \rho\}$
end

5.2 Overestimation bias and Double Q-learning algorithm

Even though Q-learning has been successfully applied to many different applications, Q-learning can sometimes overestimate action-value functions (the overestimation bias). The performance of Q-learning algorithm suffers from the overestimation bias, which can impede the agent from learning an optimal policy and have a negative impact on the convergence rate [46]. This problem is caused by the noise in the environment and the property that utilizing a single estimator and the max operator to determine the value of next state [16].

In order to reduce the overestimation bias, Hasselt [16] proposed Double Q-learning, which employs the double estimator, Q^A and Q^B. In contrast with Q-learning, one of the Q functions in Double Q-learning is chosen randomly determined between Q^A and Q^B for separating sets of experiences. The updating process of Double Q-learning uses the following equations:

$$Q^A(s_t, a_t) \leftarrow Q^A(s_t, a_t) + \alpha[r_t + \gamma Q^B(s_{t+1}, \underset{a'}{\operatorname{argmax}}\, Q^A(s_{t+1}, a')) - Q^A(s_t, a_t)] \quad (5.2)$$

$$Q^B(s_t, a_t) \leftarrow Q^B(s_t, a_t) + \alpha[r_t + \gamma Q^A(s_{t+1}, \underset{a'}{\operatorname{argmax}}\, Q^B(s_{t+1}, a')) - Q^B(s_t, a_t)] \quad (5.3)$$

In the process of updating the Q function, each Q is updated by the next state of the other Q function. In Q-learning example, in cases in which Q^A is chosen, $Q^A(s_{t+1}, a^A)$ is used for the next state value to update Q^A, where $a^A = \operatorname{argmax}_a Q^A(s_{t+1}, a)$. However, in Double Q-learning, $Q^B(s_{t+1}, a^A)$ is used for the next state value, in which each Q can learn from different sets of experience.

Both Q functions are unbiased estimators of true action-value because they are updated on the same problem, and the experience of each Q is different. Therefore, the overestimation bias can be alleviated in Double Q-leaning, though using double

estimators sometimes underestimates the action-value functions. For detailed proof of the principles of Double Q-learning, refer to Hasselt [16]. The procedure of the Double Q-learning algorithm is presented as follows:

Algorithm 3 Double Q-learning

Initialize $Q^A[(s,\, i),\, a]$, and $Q^B[(s,\, i),\, a]$, for all $s \in S$, $i \in I$, $a \in A$
$\rho_{max} \leftarrow 0$
for *each episode* **do**
    Initialize s and i
    $\rho \leftarrow 0$
    $t \leftarrow 0$
    while i_t *is not the terminal state* **do**
        Choose a_t using ε-greedy policy based on $Q^A[(s_t,\, i_t),\, a] + Q^B[(s_t,\, i_t),\, a]$
        Take action a_t and observe r_t, $(s_{t+1},\, i_{t+1})$
        $\rho \leftarrow \rho + r_t$
        $x \leftarrow\ generateRandom(0, 1)$
        if $x < 0.5$ **then**
            $a^A \leftarrow \text{argmax}_{a'}\, Q^A[(s_{t+1}, i_{t+1}), a']$
            $\delta_t^A \leftarrow\ r_t + \gamma Q^B[(s_{t+1}, i_{t+1}), a^A] - Q^A[(s_t, i_t), a_t]$
            $Q^A[(s_t, i_t), a_t] \leftarrow\ Q^A[(s_t, i_t), a_t] + \alpha\delta_t^A$
        else
            $a^B \leftarrow \text{argmax}_{a'}\, Q^B[(s_{t+1}, i_{t+1}), a']$
            $\delta_t^B \leftarrow\ r_t + \gamma Q^A[(s_{t+1}, i_{t+1}), a^B] - Q^B[(s_t, i_t), a_t]$
            $Q^B[(s_t, i_t), a_t] \leftarrow\ Q^B[(s_t, i_t), a_t] + \alpha\delta_t^B$
        end
        $t \leftarrow\ t + 1$
    end
    $\rho_{max} \leftarrow \max\{\rho_{max}, \rho\}$
end

Chapter 6

Computational experiments

We conduct four computational experiments for following different purposes. In Section 6.1, in order to measure the performance of our algorithms, we compare Q-learning and Double Q-learning with the MMGA, taking into account the same problem definition of Liu et al. [31]. In Section 6.2, we conduct experiments to analyze the quality of solutions depending on the number of delay arcs in the time-line network model. In Section 6.3, for validating the proposed approach to real-world application, we conduct experiments with the flight schedule of a Korean domestic airline, taking into account realistic constraints (e.g., turnaround time extension and multiple fleet conditions). In Section 6.4, to verify proposed reinforcement learning algorithms are customizable for various objectives, we implement experiments with different reward functions. All experiments were conducted based on the computational environment, AMD Ryzen 7 2700X Eight-Core Processor with 32 GB of RAM in Windows 10. Every algorithm and artificial environment was coded using Python 3 language, and the all experiments about the time-line network model were conducted by FICO XPRESS-IVE version 8.6. In addition, we suggest several managerial insights that are helpful for airline operation decision-making in Section 6.5.

6.1 Comparison between reinforcement learning and existing algorithms

In this section, we employ the flight schedule presented in Liu et al. [31], which consists of five airports, 70 flights, and seven aircraft of a single fleet. For every aircraft, the minimum turnaround time is 30 minutes. Because of the property of a short-haul flight schedule, most of the time intervals between flight legs are equal to the minimum turnaround time. The experiments were conducted within the disruption scenario in which the Taipei Sungshan (TSA) airport was temporarily closed.

In order to compare the performance of the MMGA and our proposed algorithms, we excluded multiple fleet and turnaround time extension conditions. We set the value of the discount factor, γ, to 0.9 and the learning rate, α, to 0.95 for all experiments. To properly evaluate the performances of Q-learning and Double Q-learning, we implemented 50 learning runs with different random seeds. In real practice, it is recommended that the process should be less than three minutes for real-time decisions [32]. Although we could find the better quality of the solution with a long length of training episodes, we set the length of training episodes to 3,000, which meant that each learning run would be terminated at the three-thousandth training episode. In some experiments, we observed that after a particular episode, the total reward of each episode (ρ) did not change and converged to a specific value in some experiments. Therefore, we included a 'convergence episode' to ensure that the agent does not implement avoidable training episodes. In this thesis, the 'convergence episode' stands for the number of episodes in which the value of ρ is the same across 500 episodes. In other words, when the convergence episode is small, the speed of convergence is faster. One learning run is terminated, and computation

times are calculated at the convergence episode or the episode 3,000.

We utilized the 'Without Swapping' (WSAP) and the MMGA for comparing the performance of Q-learning and Double Q-learning. The WSAP is a recovery method in which the swapping aircraft is not considered for the recovery option. In real practice, flight schedulers build extra buffer time between consecutive flights for absorbing unpredictable flight delays. Therefore, without implementing additional recovery options, just delaying flights' departure time could be an effective strategy. For implementing the above strategy, we utilized the environment proposed in Section 4.2. In contrast with reinforcement learning approach, there was only one valid action (e.g., initial aircraft route) in every time step when carrying out the WSAP. Thus, implementing only one episode is required for the WSAP. The pseudocode of WSAP is as follows:

Algorithm 4 WSAP

Initialize $at_c \leftarrow 0$, for all $c \in C$
$E \leftarrow$ the set of every flight's events
while *E is not empty* **do**
  $e \leftarrow$ the earliest event in E
  $f \leftarrow$ the flight which includes e
  $c \leftarrow$ an aircraft assigned to f
  $mta_c \leftarrow$ minimum turnaround time of aircraft c
  if *departure event of* f **then**
    $delay \leftarrow \max\{mta_c - (t_f^d - at_c), 0\}$
    $\tilde{t}_f^d \leftarrow t_f^d + delay$
    $\tilde{t}_f^a \leftarrow \tilde{t}_f^d + (t_f^a - t_f^d)$
  else
    $at_c \leftarrow \tilde{t}_f^a$
  end
  $E \leftarrow E \setminus e$
end

The MMGA, which was proposed by Liu et al. [31], is a multi-objective genetic

algorithm with the MOI. The multiple objectives consist of two hard constraints and three soft constraints. The hard constraints are minimum turnaround time and flight connection requirements, which decide whether the solution is feasible or not. The soft constraints are equivalent to the objectives of this thesis (i.e., Cases A, B, and C). In the time-line network model, we built 120 delay arcs and each delay arc represented delaying one minute for every flight leg, which means that departure delays were allowed up to two hours. The experiment results of the time-line network model was represented as the TLN. The Gap_L was used to assess the optimality of the best feasible solution of each algorithm (BFS) comparing it with the solution of linear programming relaxation of the time-line network with 200 delay arcs (TLN_{LP}).

$$Gap_L = \frac{(BFS) - (TLN_{LP})}{(TLN_{LP})} \times 100\%$$

We compared the performance of every algorithm with two disruption scenarios, airport closures for one hour and two hours. For the first scenario, the TSA airport was closed from 2:00 p.m. to 3:00 p.m, and all of the flights scheduled to depart or arrive at TSA in closed period were delayed until the airport was reopened. Detailed experiment results are illustrated in Table 6.1. As stated in Section 4.3, we employed different reward functions for each objective of Cases A, B, and C. The objective value stands for the maximum value of the total rewards in every episode (ρ_{max}). The results of the objective value, convergence episode, and computation times are obtained from averaging results of 50 runs with different random seeds. Among all of the algorithms, the WSAP showed the worst performance in terms of objective values for all Cases. For Cases A and B, the objective values of Double Q-learning, Q-learning and TLN were the same. For Cases C and D, the TLN outperformed other

algorithms, and Double Q-learning and Q-learning showed the second best performance. Moreover, the objective values of Q-learning and Double Q-learning were always the same regardless of different random seeds. In Case A and B, Q-learning converged faster than Double Q-learning, but in Case C, Q-learning converged slower than Double Q-learning. In every experiment, the proposed reinforcement learning approach could finish computing within one minute, except Q-learning in Case C.

Table 6.1: Comparison of performance of algorithms for one hour closure period

Case	Method	OBJ	Conv ep*	Times (sec)	Gap_L(%)
A	Double Q	430.00	2243.22	52.42	0.00
	Q	430.00	2125.94	46.51	0.00
	TLN	430.00	-	30.34	0.00
	MMGA**	435.00	-	In minutes	1.16
	WSAP	525.00	1.00	0.03	22.09
B	Double Q	6.00	2096.36	49.22	1.52
	Q	6.00	1855.92	40.20	1.52
	TLN	6.00	-	29.74	1.52
	MMGA**	6.00	-	In minutes	1.52
	WSAP	7.00	1.00	0.03	18.44
C	Double Q	13.00	2302.30	55.03	16.62
	Q	13.00	3000.00	67.09	16.62
	TLN	12.00	-	36.76	7.65
	MMGA**	14.00	-	In minutes	25.59
	WSAP	17.00	1.00	0.03	52.51
D	Double Q	12.00	1848.72	42.61	10.68
	Q	12.00	1913.14	40.81	10.68
	TLN	11.00	-	30.75	1.46
	WSAP	16.00	1.00	0.03	47.57

* Conv ep means the convergence episode.
** The results of MMGA are referred from from Liu et al. [31].

Figure 6.1 depicts the learning curves of Q-learning, WSAP, and MMGA for each objective. The learning curve represents the average value of ρ of the 50 learning runs with the different random seeds. Due to the fact that the objective value of Q-learning and Double Q-learning is the same, Double Q-learning is omitted in Figure

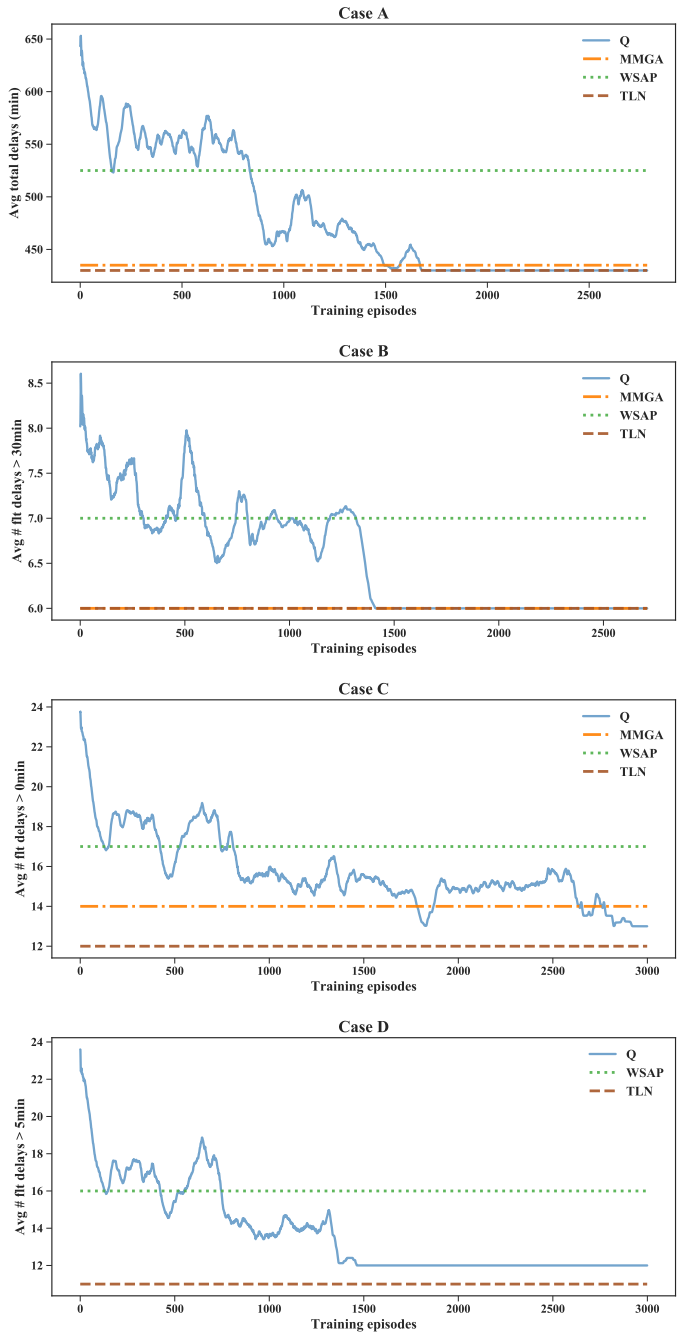

Figure 6.1: Learning curves of algorithms for the ARP

6.1. Every learning run of Q-learning was terminated before the end of the training episode. The solid line depicts the learning curve of Q-learning. The flat dotted, chain, and dashed lines depict the objective value of the WSAP, the MMGA, and the TLN, respectively. As the number of training episodes increases, the agent of Q-learning learns better policies that lead to an improved value of ρ. In addition, because the exploration rate, ε, is decreasing, the value of ρ converges. Interestingly, Q-learning showed poorer performance than the WSAP in the early stage of training episodes. This meant that implementing swapping aircraft without considering subsequent flights might lead to larger flight delays compared to a recovery strategy employing only one recovery option (i.e., delaying flight departures).

For the second scenario, the TSA airport was closed from 2:00 p.m. to 4:00 p.m, the detailed experiment results are represented in Table 6.2. In Cases A and B, the quality of solutions of TLN was better than Double Q-learning and Q-learning. However, in Cases C and D, we observed that the objective values of Double Q-learning and Q-learning showed better optimality than the solution of TLN. As with the result of Table 6.1, the WSAP showed the worst performance compared to other algorithms.

Table 6.2: Comparison of performance of algorithms for two hour closure period

Case	Method	OBJ	Conv ep*	Times (sec)	Gap_L(%)
A	Double Q	1990.00	3000.00	88.11	7.28
	Q	1990.00	3000.00	83.43	7.28
	TLN	1855.00	-	32.39	0.00
	WSAP	2025.00	1.00	0.03	9.16
B	Double Q	22.00	2998.90	87.54	22.76
	Q	22.00	3000.00	83.48	22.76
	TLN	21.00	-	33.23	17.18
	WSAP	26.00	1.00	0.03	45.08
C	Double Q	27.00	2596.64	75.18	4.93
	Q	27.00	2209.20	58.77	4.93
	TLN	29.00	-	35.07	12.70
	WSAP	31.00	1.00	0.03	20.48
D	Double Q	27.00	2605.32	76.28	6.44
	Q	27.00	2248.00	59.49	6.44
	TLN	29.00	-	38.71	14.32
	WSAP	30.00	1.00	0.03	18.26

* Conv ep means the convergence episode.

6.2 Performance of the TLN varying the size of delay arcs

In this section, we conduct computational experiments for the TLN varying the number of delay arcs for each flight leg. The detailed results for the objective value depending on the number of delay arcs are presented in Table 6.3. The columns labeled $|D|$ show the number of delay arcs for each flight leg. For the instance of one hour closed period, the TLN could not get the feasible solution when less than 50 delay arcs were built. Besides, if less than 80 delay arcs were built for the instance of two hours closed period, the feasible solution could not obtained with the TLN. Even though the objective value of the TLN became better as the size of delay arcs increase, the problem size also became bigger. In particular, except for the objective of Case A, we observed that the TLN could not get the solution within ten minutes when more than 180 delay arcs were built.

By analyzing the above experiments, we could observe a difficulty in determining the appropriate number of delay arcs for the TLN. The solution's quality improved with the increase of the number of delay arcs, but the computation time increased dramatically since the number of decision variables and constraints increased. Moreover, there was a feasibility issue in the TLN. Because, when there were not enough delay arcs, the mathematical always obtained the infeasible solution. On the other hand, due to the property of simulation based approach, the proposed reinforcement learning algorithms avoid the process of determining the size of delaying arcs. In addition, we resolved the feasibility issue of the TLN since we built the environment obtaining a feasible solution (i.e., the recovered flight schedule) for every episode. Therefore, the reinforcement learning approach could adapt flexibly to various flight instances, compared to the TLN requiring to determine the cost parameters and the

number of delay arcs from scratch.

Table 6.3: Performance of the TLN depending on the number of delay arcs

Airport	Closed period	$\|D\|$	OBJ			
			Case A	Case B	Case C	Case D
TSA	14:00∼15:00	40	inf	inf	inf	inf
		50	480	7	17	16
		60	430	6	13	13
		70	430	6	13	12
		80	430	6	13	12
		90	430	6	12	11
		120	430	6	12	11
	14:00∼16:00	70	inf	inf	inf	inf
		80	1855	23	30	30
		90	1855	21	29	29
		120	1855	21	29	29
		180	1855	-*	-*	-*

* Cannot solve within ten minutes.

6.3 Aircraft recovery for a complex real-world case: a Korean domestic airline

In this section, the flight schedule of an August 2020 domestic flight schedule from one of the airlines in Korea was used as the input for further analysis. There are 20 aircraft operating 94 flights from six airports: Gimpo (GMP), Jeju (CJU), Busan (PUS), Ulsan (USN), Cheongju (CJJ) and Gwangju (KWJ). Table 6.4 presents the detailed information of the number of aircraft, the minimum turnaround times, and the types of aircraft that could be swapped for each type of aircraft. The aircraft consisted of four types, and the minimum turnaround time of each type of aircraft was different. Moreover, there are three subfleets: i) A220, and ii) B737, and iii) B777 and A330. The feasible types of aircraft that could be swapped were different depending on each aircraft. In contrast with Section 6.1, we considered the turnaround time extension and multiple fleet conditions in this experiment. We compared the performance of Q-learning, Double Q-learning, and the WSAP, without the MMGA, which cannot be applied to the turnaround time extensions and multiple fleet conditions.

Table 6.4: Aircraft information of the flight schedule of a Korean domestic airline

Aircraft type	Units	Turnaround time (min)	Substitutions
A220	11	40	A220
B737	5	40	B737
B777	2	50	B777, A330
A330	2	50	B777, A330

Ten disruption scenarios of airport closures were proposed to validate the efficiency of the algorithms. The detailed information of the scenarios is summarized

in Table 6.5. All times in the category of closed periods were reported in minutes. The disruption scenarios were chosen for covering a wide range of disruption types with the following properties. First, the closed airport was determined in light of the traffic volume at that airport. Among the airports in Korea, the number of passengers and flights at the CJU is the greatest for domestic flight operations. Moreover, temporary closure frequently occurs at the CJU due to the often severe weather conditions in Jeju. Considering the characteristics of airports in Korea, the closed airports corresponding to high, medium, and low traffic volumes were selected as the CJU, GMP, and CJJ, respectively. Second, the simulated scenarios considered the various lengths of the closed periods of airports. The closed periods of airports for the scenarios were decided based on the fact that the closed periods in real-world cases at Jeju are usually about one to two hours. Third, because airports that have been closed for a long time are usually more congested, scenarios with longer closed periods of airports had more extended periods and turnaround time extensions.

Table 6.5: Information of disruption scenarios

Scenario	Closed period	Closed airport	Turnaround time extension	
			Extension period	Extension time
1	840∼900 (1hr)	CJU	-	-
2	840∼900 (1hr)	CJU	3hr	+10 min
3	900∼990 (1.5hr)	CJU	4hr	+15 min
4	860∼980 (2hr)	CJU	5hr	+20 min
5	890∼950 (1hr)	GMP	3hr	+10 min
6	860∼950 (1.5hr)	GMP	4hr	+15 min
7	870∼990 (2hr)	GMP	5hr	+20 min
8	900∼960 (1hr)	CJJ	3hr	+10 min
9	860∼950 (1.5hr)	CJJ	4hr	+15 min
10	840∼960 (2hr)	CJJ	5hr	+20 min

In this section, we set the numerical value of γ to 0.7 and α to 0.85. Because the state space of this problem was larger than in the problem of Section 6.1, a lot of episodes had to be implemented to obtain valid results when the value of γ and α was large. Therefore, we set smaller values of γ and α compared to Section 6.1. We implemented five different random seeds for each scenario and for all Cases. When the value of ρ did not converge until the end of the training episode, the computation times at episode 3,000 were utilized, and the 3,000 was used for the convergence episode for individual learning runs.

The detailed results for every scenario are presented in Table 6.6. The experimental results were obtained by averaging the results of five runs with different random seeds. We can note that proposed reinforcement learning algorithms outperformed the WSAP, and Double Q-learning showed the best performance in terms of objective values. Furthermore, Double Q-learning and Q-learning completed learning within two minutes in all experiments, but three out of 40 (Double Q-learning for Cases B, C, and D in Scenario 4). Even in these three experiments, the computation time was close to two minutes. Obviously, as the length of the closed period increased, the total flight delay incurred in the schedule increased. If the disruption occurred in an airport with high traffic volume (CJU), many flight operations were affected compared to a low traffic volume airport (CJJ). Therefore, when an airport with high traffic volumes temporarily closed, our experiments demonstrated that the total flight delay increased as the length of the closed period increase. In the case when the disruption occurred in the CJU, the total delays of Scenario 4 increased dramatically compared to Scenario 2 (to 1,545 from 154.2). However, in the case of the CJJ, the increase of the total delays between Scenarios 8 and 10 was relatively

small (to 234 from 90). In addition, comparing the experiment results in Scenarios 1 and 2, the turnaround time extension increased the objective values of Cases A, C, and D. However, because the extension time was 10 minutes, the objective value of Case B was not affected.

Considering the experiment for the flight schedule of a Korean domestic airline, the Q-learning suffered from convergence problem due to the overestimation bias. However, Double Q-learning alleviated challenges of convergence compared to Q-learning. We set the length of training episode to 10,000 to compare the performance of Double Q-learning and Q-learning. Figure 6.2 shows the learning curves of Q-learning, Double Q-learning and the WSAP for Cases A, B, C, and D in Scenario 2. The dashed line depicts the learning curve of Double Q-learning. The learning curves were obtained by averaging the value of ρ of five learning runs with different random seeds. In Case B, Double Q-learning, Q-learning, and the WSAP had the same result of objective value, one. The convergence episode of Double Q-learning was smaller than that of Q-learning. However, in Cases A and C, the agent of Q-learning could not learn a policy that improves the value of ρ continuously until the end of training episodes, and the total reward diverged. On the other hand, the agent of Double Q-learning showed steady improvement and learned a policy that led to the smallest objective value among the comparison algorithms. In contrast with Q-learning, the value of ρ converged at the convergence episode in Double Q-learning.

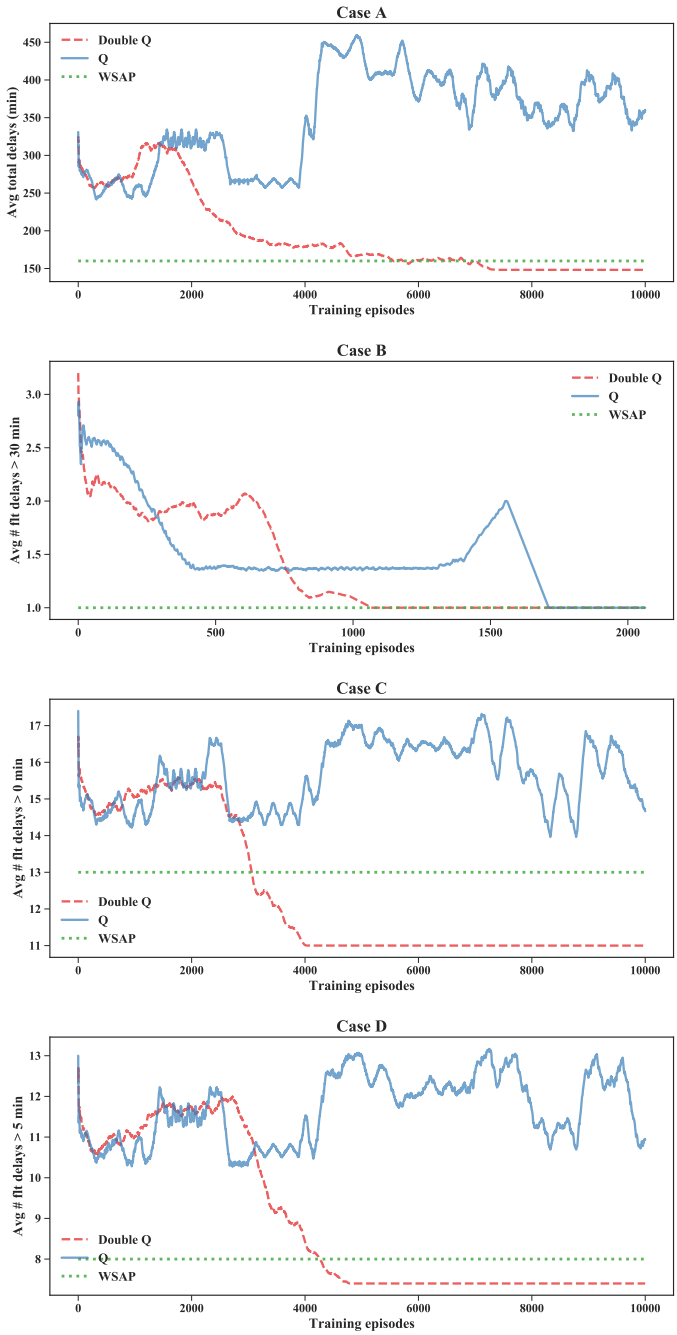

Figure 6.2: Learning curves of algorithms for the flight schedule of a Korean domestic airline in Scenario 2

Table 6.6: Comparison of performance of algorithms for the flight schedule of a Korean domestic airline

Scenario	Method	Case A			Case B			Case C			Case D		
		OBJ	Conv ep	Times	OBJ	Conv ep	Times	OBJ	Conv ep	Times	OBJ	Conv ep	Times
1	Double Q	120.00	3000.00	116.22	1.00	1202.80	45.32	8.00	3000.00	117.38	4.40	3000.00	115.10
	Q	119.00	3000.00	109.80	1.00	2060.80	73.76	8.00	3000.00	109.32	4.00	3000.00	107.39
	WSAP	135.00	1.00	0.05	1.00	1.00	0.05	10.00	1.00	0.05	6.00	1.00	0.05
2	Double Q	154.20	2981.20	116.97	1.00	1202.80	45.37	11.00	3000.00	118.51	7.00	3000.00	115.83
	Q	161.00	3000.00	110.06	1.00	2060.80	73.84	11.80	3000.00	109.73	7.20	3000.00	107.80
	WSAP	160.00	1.00	0.05	1.00	1.00	0.05	13.00	1.00	0.05	8.00	1.00	0.05
3	Double Q	1128.00	1351.20	47.43	16.00	2571.66	92.67	24.00	3000.00	109.07	20.00	3000.00	106.97
	Q	1128.00	1264.20	41.78	16.00	3000.00	102.19	24.00	2569.20	86.90	20.00	3000.00	100.33
	WSAP	1310.00	1.00	0.05	19.00	1.00	0.05	27.00	1.00	0.05	23.00	1.00	0.05
4	Double Q	1545.00	2393.00	96.28	19.00	3000.00	121.62	29.00	3000.00	122.56	25.00	3000.00	120.07
	Q	1545.00	3000.00	114.56	19.00	3000.00	114.39	29.00	3000.00	115.02	25.00	3000.00	112.70
	WSAP	1765.00	1.00	0.05	22.00	1.00	0.05	32.00	1.00	0.05	29.00	1.00	0.05
5	Double Q	145.60	2737.80	95.69	2.00	1739.40	59.53	8.00	3000.00	105.56	4.00	3000.00	103.18
	Q	142.00	3000.00	98.16	2.00	576.20	17.92	8.00	3000.00	98.60	4.00	3000.00	96.66
	WSAP	385.00	1.00	0.05	6.00	1.00	0.05	12.00	1.00	0.05	9.00	1.00	0.05
6	Double Q	459.20	3000.00	114.72	6.00	3000.00	114.53	12.00	3000.00	114.94	8.00	3000.00	112.50
	Q	481.00	3000.00	107.15	6.00	673.60	22.96	12.60	3000.00	107.30	8.60	3000.00	105.21
	WSAP	695.00	1.00	0.05	10.00	1.00	0.05	16.00	1.00	0.05	13.00	1.00	0.05
7	Double Q	888.00	3000.00	114.27	7.00	3000.00	114.25	19.00	3000.00	114.74	16.00	3000.00	111.73
	Q	918.00	3000.00	107.07	7.80	3000.00	106.57	19.00	3000.00	107.03	16.00	3000.00	104.88
	WSAP	1235.00	1.00	0.05	12.00	1.00	0.05	20.00	1.00	0.05	17.00	1.00	0.05
8	Double Q	90.00	3000.00	99.35	0.00	3000.00	99.02	7.00	2935.60	97.04	3.00	3000.00	97.33
	Q	90.00	3000.00	92.65	0.00	640.00	18.86	7.00	3000.00	92.57	4.00	3000.00	91.07
	WSAP	140.00	1.00	0.00	0.00	1.00	0.05	11.00	1.00	0.05	7.00	1.00	0.05
9	Double Q	147.00	2919.40	105.81	1.00	3000.00	108.65	8.80	2917.40	105.98	5.40	3000.00	107.20
	Q	165.00	3000.00	101.88	1.00	698.40	22.76	9.00	3000.00	101.61	5.00	3000.00	99.80
	WSAP	235.00	1.00	0.05	3.00	1.00	0.05	12.00	1.00	0.05	9.00	1.00	0.05
10	Double Q	234.00	3000.00	116.04	1.60	3000.00	115.93	8.00	3000.00	116.29	6.00	3000.00	114.12
	Q	237.00	3000.00	108.58	1.20	3000.00	108.74	10.00	3000.00	108.35	7.00	3000.00	106.55
	WSAP	320.00	1.00	0.05	3.00	1.00	0.05	12.00	1.00	0.05	9.00	1.00	0.05

6.4 Validation for different objectives

Turning now to evaluating whether the reward functions defined in Section 4.3 have good performance for each objective, we compare four Double Q-learning with different reward functions. Throughout this thesis, we use the terms 'Double Q-learning for Cases A, B, C, and D' to indicate the Double Q-learning algorithm that applies appropriate reward functions for each objective. 'Double Q-learning for Case A' applies $\min\{t_n^d - \tilde{t}_n^d, 0\}$ for reward function and the others apply different reward functions defined for each objective (see Section4.3).

Table 6.7 shows the performance of Double Q-learning for Cases A, B, C, and D in every objective. The objective values were obtained by averaging the results of five learning runs at the convergence episode. The results of Cases A, B, C, and D are obtained with the rescheduled flights, the solution of ρ_{max}. Double Q-learning with appropriate reward functions outperformed other reward functions for customized objectives in all but seven experiments out of 30. Double Q-learning for Case A had the highest objective value for all Scenarios for the objective of Case A. In Case B, Double Q-learning for Case B performed best in all Scenarios except for Scenario 10. Double Q-learning for Case C showed the best performance in all Scenarios except Scenario 2 in terms of Case C. However, in the objective of Case D, Double Q-learning for Case D showed poor performance compared to other objectives.

Table 6.7: Comparison of the performance of Double Q-learning with different reward functions for each objective

Method	Case	Scenario									
		1	2	3	4	5	6	7	8	9	10
Double Q for Case A	A	120.00	154.20	1128.00	1545.00	145.60	459.20	888.00	90.00	147.00	232.00
	B	1.00	1.60	18.00	19.00	2.00	6.00	7.00	0.00	1.00	1.0
	C	8.00	10.20	27.00	29.00	8.00	12.00	21.00	7.00	8.20	10.20
	D	4.00	6.00	21.00	25.00	4.00	8.00	18.00	3.00	4.20	7.20
Double Q for Case B	A	216.00	246.00	1310.00	1601.00	211.00	545.00	904.00	164.00	226.00	323.00
	B	1.00	1.00	16.00	19.00	2.00	6.00	7.00	0.00	1.00	1.60
	C	13.20	15.20	28.60	31.40	11.00	14.80	22.20	11.60	12.80	14.00
	D	8.20	10.20	23.40	27.00	7.00	10.80	19.20	6.80	8.20	10.80
Double Q for Case C	A	120.00	206.00	1364.00	1630.00	151.00	490.00	1062.80	90.00	157.00	260.0
	B	1.00	2.00	19.40	20.00	2.00	6.00	10.40	0.00	1.00	3.00
	C	8.00	11.00	24.00	29.00	8.00	12.00	19.00	7.00	8.80	8.00
	D	4.00	7.40	22.40	26.00	4.00	8.00	17.40	3.00	4.80	5.00
Double Q for Case D	A	126.00	217.80	1250.00	1559.00	151.00	490.00	1225.00	92.00	168.00	265.00
	B	1.00	2.00	17.00	20.00	2.00	6.00	12.00	0.00	1.00	3.00
	C	8.40	11.40	26.00	29.80	8.00	12.00	19.00	7.40	9.60	9.20
	D	4.40	7.00	20.00	25.00	4.00	8.00	16.00	3.00	5.40	6.00

For further comparison, we examined the number of the best performing Double Q-learning for Cases A, B, C, and D for every objective. Figure 6.3 presents a comparative analysis of Double Q-learning with different reward functions for every objective. With the setup of the same random seed and scenario, we compared Double Q-learning for Cases A, B, C, and D, and the method obtained the best objective value was counted as the number of best performing for each objective. In the case in which several Double Q-learning algorithms obtained the same best objective value, all those algorithms were considered for the category of best performing. If an algorithm showed the best performance for every learning runs in ten scenarios, the best performing number was set at 50 (i.e., 5×10). As anticipated, our experiment showed that Double Q-learning with the appropriate reward function for each objective performed best for each objective. This meant that Double Q-learning with the appropriate reward function performed best on each objective when trained specifically to optimize for the corresponding objective. Especially, Double Q-learning for Case A showed the robustness in terms of performances for any type of objective.

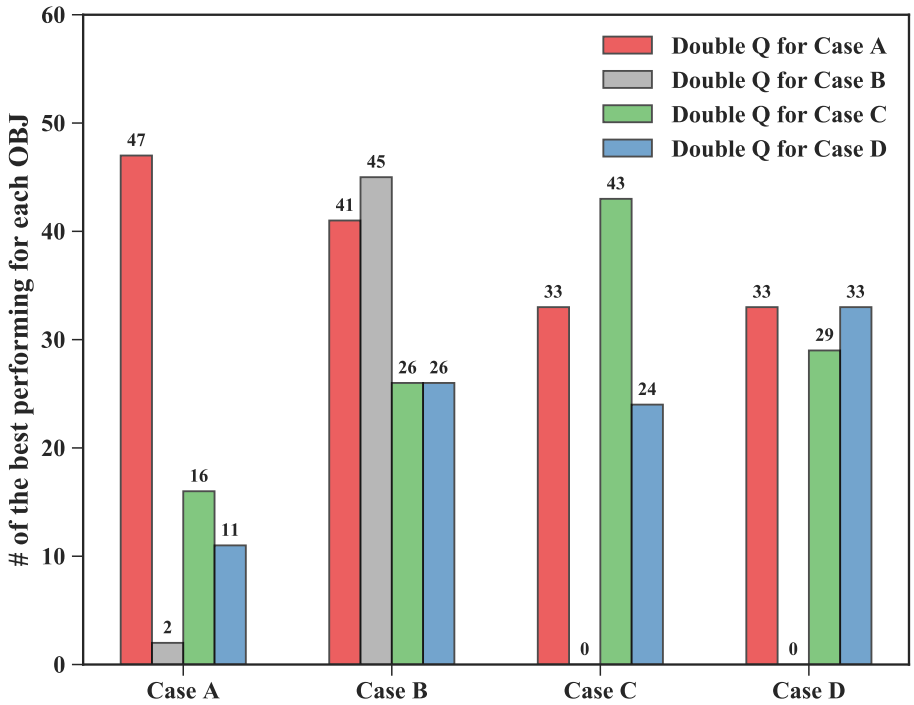

Figure 6.3: Performance for different objectives

6.5 Managerial insights

This thesis offers managerial insights, which could be instructive to airline operations controllers. By analyzing the results of the computational experiments, we derived the following managerial insights.

(1) Based on the computational experiments, the proposed reinforcement learning approach showed effective performance on a real-world flight schedule. In actual practice, most commercial airlines manage disruptions of flight schedules with optimization-based decision tools and long-standing practices established by the airline industry and the practical knowledge of AOCC staff. However, in addition to optimization-based decision tools and AOCC staff expertise,

a decision tool modeled with the reinforcement learning approach could help airlines for making better decisions.

(2) Q-learning is well applied to aircraft recovery of simple flight schedules, which do not consider various realistic conditions. However, if airline operations controllers take into account complex conditions, which create noise in the environment of reinforcement learning, Q-learning could show poor performance due to the overestimation bias. In such cases, therefore, we showed that utilizing Double Q-learning would be an effective strategy.

(3) In this thesis, we suggested three objectives and defined reward functions for each objective. Double Q-learning is configurable to different objectives. Double Q-learning with an appropriate reward function for each objective outperforms other reward functions. Therefore, if airline operations controllers wanted to meet a specific objective, we suggested that they could utilize proposed appropriate reward functions. On the other hand, if airline controllers had no specific goals and wanted to implement aircraft recovery to perform well for overall objectives, we also determined that adopting the reward function for minimizing total delays would be an effective strategy.

(4) Swapping aircraft is not always a valid strategy for aircraft recovery after airport closures. Depending on the conditions of available resources and subsequent flights, swapping aircraft could increase flight delays in the overall flight schedule compared to just delaying the departure time of subsequent flights until the aircraft is ready. Thus, we suggested in this thesis that it is necessary to consider outcomes for subsequent flights when swapping aircraft.

Chapter 7

Conclusions

Implementing aircraft recovery to minimize the damage from disruptions is significant for airline's bottom lines, in terms of both customer satisfaction and operations costs. In aircraft recovery, it is necessary to take into account various objectives of airlines and realistic conditions that affect their operations. Considering the advantages of flexibility in establishing various objectives and adapting them to complex assumptions, we adopted the method of reinforcement learning, specifically Q-learning and Double Q-learning, for the ARP. To the best of our knowledge, this thesis is a first step to applying reinforcement learning approach to the ARP, since most existing studies adopted operations research methods instead. Among numerous types of flight disruptions, we concentrated in this thesis on airport closures. We built an environment of reinforcement learning and defined states, actions, and reward functions. Especially, the three different reward functions were defined for each objective. We evaluated the performances of Q-learning and Double Q-learning and compared the results with the MMGA benchmark. Furthermore, in the real-world flight schedule of one of the domestic airlines in Korea, we validated the advantages of utilizing Double Q-learning, which alleviates the overestimation bias of Q-learning. In addition, our computational experiments in this thesis showed that Double Q-

learning with appropriate reward functions outperformed other reward functions in customized objectives. We expect our approach to serve as a bridge for utilizing the reinforcement learning in airline disruption management.

Finally, a number of potential limitations need to be considered. First, the performance of the proposed reinforcement learning approach is rather disappointing in a large scale flight schedule. The state and action space depends on the number of aircraft, and the time step size in an episode increases depending on the number of flights. Therefore, the run time for the agent learning an optimal policy increases as the number of aircraft and flights increases. Second, the values of three control parameters, ε, α, and γ, were selected by performing an informal search. These values significantly affected the performance of reinforcement learning. We experimented with limited parameter settings widely used in the existing literature.

For further research, we intend to extend our study in order to derive general policy, which could be applied to many disruption instances. Because this thesis defined the states and actions based on aircraft routes, it was difficult to obtain a general policy. However, by redefining states and actions and by utilizing suitable technology (e.g., a neural network) to approximate action-value functions, the agent might learn a general policy. Moreover, an airline's engaging in non-profitable duty swaps for flights could cause trouble for changing crews, and could consume other resources. Unnecessary swaps, therefore, might be regarded as measures of deviation from original flight schedules. Thus, revising reward functions that take into account the additional cost of swapping aircraft is reserved for future work.

Bibliography

[1] T. Andersson, *Solving the flight perturbation problem with meta heuristics*, Journal of Heuristics, 12 (2006), pp. 37–53.

[2] M. F. Arguello, *Framework for exact solutions and heuristics for approximate solutions to airlines' irregular operations control aircraft routing problem.*, (1998).

[3] M. F. Argüello, J. F. Bard, and G. Yu, *A grasp for aircraft routing in response to groundings and delays*, Journal of Combinatorial Optimization, 1 (1997), pp. 211–228.

[4] P. Balakrishna, R. Ganesan, and L. Sherry, *Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of tampa bay departures*, Transportation Research Part C: Emerging Technologies, 18 (2010), pp. 950–962.

[5] C. Barnhart, N. L. Boland, L. W. Clarke, E. L. Johnson, G. L. Nemhauser, and R. G. Shenoi, *Flight string models for aircraft fleeting and routing*, Transportation science, 32 (1998), pp. 208–220.

[6] R. Bellman, *Dynamic programming*, Science, 153 (1966), pp. 34–37.

[7] P. Belobaba, A. Odoni, and C. Barnhart, *The Global Airline Industry*, John Wiley & Sons, 2015.

[8] S. Bratu and C. Barnhart, *An analysis of passenger delays using flight operations and passenger booking data*, Air Traffic Control Quarterly, 13 (2005), pp. 1–27.

[9] ——, *Flight operations recovery: New approaches considering passenger recovery*, Journal of Scheduling, 9 (2006), pp. 279–298.

[10] J.-M. Cao and A. Kanafani, *Real-time decision support for integration of airline flight cancellations and delays part i: mathematical formulation*, Transportation Planning and Technology, 20 (1997), pp. 183–199.

[11] J. Clausen, A. Larsen, J. Larsen, and N. J. Rezanova, *Disruption management in the airline industry—concepts, models and methods*, Computers & Operations Research, 37 (2010), pp. 809–821.

[12] N. Eggenberg, M. Salani, and M. Bierlaire, *Constraint-specific recovery network for solving airline recovery problems*, Computers & operations research, 37 (2010), pp. 1014–1026.

[13] M. R. Garey and D. S. Johnson, *Computers and intractability*, vol. 174, freeman San Francisco, 1979.

[14] A. Gosavi, *Reinforcement learning: A tutorial survey and recent advances*, INFORMS Journal on Computing, 21 (2009), pp. 178–192.

[15] A. Gosavii, N. Bandla, and T. K. Das, *A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking*, IIE transactions, 34 (2002), pp. 729–742.

[16] H. V. Hasselt, *Double q-learning*, in Advances in neural information processing systems, 2010, pp. 2613–2621.

[17] G. Hondet, L. Delgado, and G. Gurtner, *Airline disruption management with aircraft swapping and reinforcement learning*, in SESAR Innovation Days 2018 (SID 2018), 2018.

[18] Y. Hu, H. Liao, S. Zhang, and Y. Song, *Multiple objective solution approaches for aircraft rerouting under the disruption of multi-aircraft*, Expert Systems with Applications, 83 (2017), pp. 283–299.

[19] Y. Hu, B. Xu, J. F. Bard, H. Chi, et al., *Optimization of multi-fleet aircraft routing considering passenger transiting under airline disruption*, Computers & Industrial Engineering, 80 (2015), pp. 132–144.

[20] D. Hummels, *Transportation costs and international trade in the second era of globalization*, Journal of Economic Perspectives, 21 (2007), pp. 131–154.

[21] A. I. Jarrah, G. Yu, N. Krishnamurthy, and A. Rakshit, *A decision support framework for airline flight cancellations and delays*, Transportation Science, 27 (1993), pp. 266–280.

[22] Z. Jiang, W. Fan, W. Liu, B. Zhu, and J. Gu, *Reinforcement learning approach for coordinated passenger inflow control of urban rail transit in*

peak hours, Transportation Research Part C: Emerging Technologies, 88 (2018), pp. 1–16.

[23] M. A. Jones, D. L. Mothersbaugh, and S. E. Beatty, *Why customers stay: measuring the underlying dimensions of services switching costs and managing their differential strategic outcomes*, Journal of Business Research, 55 (2002), pp. 441–450.

[24] O. Khaled, M. Minoux, V. Mousseau, S. Michel, and X. Ceugniet, *A multi-criteria repair/recovery framework for the tail assignment problem in airlines*, Journal of Air Transport Management, 68 (2018), pp. 137–151.

[25] M. A. Khamis and W. Gomaa, *Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework*, Engineering Applications of Artificial Intelligence, 29 (2014), pp. 134–151.

[26] N. Kohl, A. Larsen, J. Larsen, A. Ross, and S. Tiourine, *Airline disruption management—perspectives, experiences and outlook*, Journal of Air Transport Management, 13 (2007), pp. 149–162.

[27] R. J. Lawhead and A. Gosavi, *A bounded actor–critic reinforcement learning algorithm applied to airline revenue management*, Engineering Applications of Artificial Intelligence, 82 (2019), pp. 252–262.

[28] Z. Liang, F. Xiao, X. Qian, L. Zhou, X. Jin, X. Lu, and S. Karichery, *A column generation-based heuristic for aircraft recovery problem with airport*

capacity constraints and maintenance flexibility, Transportation Research Part B: Methodological, 113 (2018), pp. 70–90.

[29] H. Lin and Z. Wang, *Fast variable neighborhood search for flight rescheduling after airport closure*, IEEE Access, 6 (2018), pp. 50901–50909.

[30] T.-K. Liu, C.-H. Chen, and J.-H. Chou, *Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm*, Expert Systems with Applications, 37 (2010), pp. 2307–2315.

[31] T.-K. Liu, C.-R. Jeng, and Y.-H. Chang, *Disruption management of an inequality-based multi-fleet airline schedule by a multi-objective genetic algorithm*, Transportation Planning and Technology, 31 (2008), pp. 613–639.

[32] M. Løve, K. R. Sørensen, J. Larsen, and J. Clausen, *Disruption management for an airline—rescheduling of aircraft*, in Workshops on Applications of Evolutionary Computation, 2002, pp. 315–324.

[33] S. J. Maher, *Solving the integrated airline recovery problem using column-and-row generation*, Transportation Science, 50 (2016), pp. 216–239.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., *Human-level control through deep reinforcement learning*, nature, 518 (2015), pp. 529–533.

[35] J. D. Petersen, G. Sölveling, J.-P. Clarke, E. L. Johnson, and S. Shebalov, *An optimization approach to airline integrated recovery*, Transportation Science, 46 (2012), pp. 482–500.

[36] J. Rapajic, *Beyond Airline Disruptions*, Ashgate Publishing, Ltd., 2009.

[37] H. Robbins and S. Monro, *A stochastic approximation method*, The annals of mathematical statistics, (1951), pp. 400–407.

[38] J. M. Rosenberger, E. L. Johnson, and G. L. Nemhauser, *Rerouting aircraft for airline recovery*, Transportation Science, 37 (2003), pp. 408–421.

[39] D. Šemrov, R. Marsetič, M. Žura, L. Todorovski, and A. Srdic, *Reinforcement learning approach for train rescheduling on a single-track railway*, Transportation Research Part B: Methodological, 86 (2016), pp. 250–267.

[40] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[41] D. Teodorović and S. Guberinić, *Optimal dispatching strategy on an airline network after a schedule perturbation*, European Journal of Operational Research, 15 (1984), pp. 178–182.

[42] D. Teodorović and G. Stojković, *Model for operational daily airline scheduling*, Transportation Planning and Technology, 14 (1990), pp. 273–285.

[43] B. G. Thengvall, J. F. Bard, and G. Yu, *Balancing user preferences for aircraft schedule recovery during irregular operations*, Iie Transactions, 32 (2000), pp. 181–193.

[44] ——, *A bundle algorithm approach for the aircraft schedule recovery problem during hub closures*, Transportation Science, 37 (2003), pp. 392–407.

[45] B. G. Thengvall, G. Yu, and J. F. Bard, *Multiple fleet aircraft schedule recovery following hub closures*, Transportation Research Part A: Policy and Practice, 35 (2001), pp. 289–308.

[46] S. Thrun and A. Schwartz, *Issues in using function approximation for reinforcement learning*, in Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum, 1993, pp. 1–9.

[47] 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.

[48] H. Van Hasselt, A. Guez, and D. Silver, *Deep reinforcement learning with double q-learning*, arXiv preprint arXiv:1509.06461, (2015).

[49] C. J. Watkins and P. Dayan, *Q-learning*, Machine Learning, 8 (1992), pp. 279–292.

[50] Z. Wu, Q. Gao, B. Li, C. Dang, and F. Hu, *A rapid solving method to large airline disruption problems caused by airports closure*, IEEE Access, 5 (2017), pp. 26545–26555.

[51] S. Yan and C.-G. Lin, *Airline scheduling for the temporary closure of airports*, Transportation Science, 31 (1997), pp. 72–82.

[52] S. Yan and D.-H. Yang, *A decision support framework for handling schedule perturbation*, Transportation Research Part B: Methodological, 30 (1996), pp. 405–419.

[53] Y. N. Yetimoğlu and M. S. Aktürk, *Aircraft and passenger recovery during an aircraft's unexpected unavailability*, Journal of Air Transport Management, 91 (2021), p. 101991.

[54] D. Zhang, C. Yu, J. Desai, and H. H. Lau, *A math-heuristic algorithm for the integrated air service recovery*, Transportation Research Part B: Methodological, 84 (2016), pp. 211–236.

국문초록

항공사는 보유하고 있는 자원을 최대한 효율적으로 사용하여 항공 일정계획을 수립하기 위해 비용과 시간을 많이 소모하게 된다. 하지만 공항 임시폐쇄와 같은 긴급 상황이 발생하면 항공편의 비정상 운항이 발생하게 된다. 따라서 이러한 상황이 발생하였을 때, 피해를 최대한 줄이기 위해 항공 일정계획을 복원하게 된다. 본 연구는 강화학습을 이용하여 공항 임시폐쇄 상황에서 항공 일정계획 복원 문제를 푼다. 본 연구에서는 항공기 복원 방법으로 항공편 지연과 항공기 교체의 두 가지 방법을 채택하였으며, 항공 일정계획 복원 문제에 강화학습을 적용하기 위해서 마르코프 결정 과정과 강화학습 환경을 구축하였다. 본 실험을 위해 대한민국 항공사의 실제 국내선 항공 일정계획을 사용하였다. 강화학습 알고리즘을 사용하여 기존의 연구에 비해 항공 일정계획을 효율적으로 복원하였으며, 여러 현실적인 조건과 다양한 목적함수에 유연하게 적용하였다.

주요어: 항공 일정계획 복원 문제, 공항 임시폐쇄, 항공기 교체, 강화학습, 큐러닝 알고리즘, 이중 큐러닝 알고리즘

학번: 2019-20784