Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A reinforcement learning approach for multi-fleet aircraft recovery under airline disruption

Junhyeok Lee [a], Kyungsik Lee [a,b], Ilkyeong Moon [a,b,*]

[a] Department of Industrial Engineering, Seoul National University, Seoul 08826, Republic of Korea
[b] Institute of Engineering Research, Seoul National University, Seoul 08826, Republic of Korea

ARTICLE INFO

Article history:
Received 28 February 2022
Received in revised form 1 August 2022
Accepted 16 August 2022
Available online 22 August 2022

Keywords:
Airline disruption
Aircraft recovery
Reinforcement learning
Q-learning
Double Q-learning

ABSTRACT

An airline scheduler plans flight schedules with efficient resource utilization. However, unpredictable airline disruptions, such as temporary closures of an airports, cause schedule perturbations. Therefore, recovering disrupted flight schedules is essential for airlines. Many previous studies have relied on copies of flight arcs, which could affect the quality of solutions, and have not addressed the key measure of airlines' on-time performance as their objective. To fill these research gaps, we propose Q-learning and Double Q-learning algorithms using the reinforcement learning approach for aircraft recovery to support airline operations. We present an artificial environment of daily flight schedules and the Markov decision process for aircraft recovery. The proposed approach is first compared with existing algorithms on the benchmark instance. In comparison with other algorithms, the developed Q-learning and Double Q-learning algorithms obtain high-quality solutions within the proper computation time. To verify that the proposed approach can be applicable to a real-world case and can adapt to realistic conditions, we employ a domestic flight schedule from one of the airlines in South Korea. We evaluate the reinforcement learning approach on a set of experiments carried out on real-world data. Computational experiments show that reinforcement learning algorithms recover disrupted flight schedules effectively, and that our approaches flexibly adapt to various objectives and realistic conditions.

1. Introduction

As international trade and demand for air travel have increased, the number of commercial airlines has grown [1]. 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. Flight schedules are usually established one season ahead of the actual operation, following the forecast of passenger demand and seasonal factors [2]. 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 [3].

Although efficient flight schedules are established, perturbations of flight schedules can occur because of 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 because of interconnected resources (i.e., late arrivals of previous flights cause 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 [4]. In order to alleviate minor stochastic delays and small disruptions, airline schedulers usually add buffer times between flight legs [5]. 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 time (TAT) causes serious damage to airlines' bottom lines. The TAT 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 [6]. Several research studies have focused on the integrated recovery of aircraft, crews, and passengers [7–10]. However, due to the complexity of the airline recovery process, the airline usually segments the process into three stages: aircraft, crew, and passenger recovery [5]. Because aircraft are 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

* Corresponding author at: Department of Industrial Engineering, Seoul National University, Seoul 08826, Republic of Korea.
E-mail address: ikmoon@snu.ac.kr (I. Moon).

affected aircraft to best meet the objectives of the 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 study 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 [6]. In particular, swapping aircraft is a strategy commonly used in the aircraft recovery process, and it is defined as switching flights on a pair of aircraft. On the other hand, because cancellations and ferrying cause airlines great financial loss, they are seldom used in real practice [11]. When schedule disruptions occur, the problem of revising routes for affected aircraft using the previous options is known as the aircraft recovery problem (ARP).

Even though many previous studies have dealt with ARP in various aspects, this study seeks to fill two research gaps in the ARP research area. The first research gap is the practice of using copies of flight arcs. Many previous studies have used copies of flight arcs within network flows to recover the airline schedule by reassigning the aircraft to flights [8,12–14]. However, if a decision maker generates a large number of copies, it becomes more difficult to solve the ARP because the network size is increased. In contrast, if not enough copies are generated, the quality of solutions becomes worse because important copies of flight arcs can be missed. Huang et al. [15] also acknowledged this problem, so they presented the copy generation algorithm that evaluates the importance of generated copies. Although Huang et al. [15] mitigated issues of flight copies, they still relied on flight copies in the solution approach. A detailed discussion about the dilemma of generating the copies of flight arcs will be presented in Section 3.4.

The second research gap is that previous studies did not incorporate the key measures in real airline operations as objectives. Most of the existing research for the ARP focuses on minimizing the total cost incurred by flight delays [16,17]. 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. If the time differences between actual and scheduled flight events exceed the permissible range (specific time frame), the occurrence is defined as a 'flight delay.' 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. Not many studies have been carried out to deal with the objective of minimizing flight delays, except for the study by Liu et al. [18].

Motivated by the above research gaps in existing ARP literature, this study defines the following five research questions to address:

(1) How can we solve the ARP without depending on copies of flight arcs for the solution approach?
(2) Which solution approach can flexibly adapt to various objectives that accommodate the key measure in real airline operations?
(3) If a solution approach suitable to solving research questions (1) and (2) is developed, what advantages does this approach have compared to exiting algorithms?
(4) Is the developed solution approach applicable to real-world airline schedules, and does it adapt to complex conditions?
(5) How can we validate that the solution approach is configurable for various objectives?

In order to answer the above research questions, we adopt the reinforcement learning (RL) approach. The main reasons we adopt this approach will be explained in detail in Section 2 by drawing comparisons with existing literature.

In this study, our purpose is to develop a framework for applying the RL approach to aircraft recovery in order to make it useful for actual operations. To the best of our knowledge, there has been no experimental study solving the ARP efficiently utilizing the RL approach. Specifically, we adopt Q-learning (QL) and Double Q-learning (DQL) for the RL algorithms. The proposed framework could support airlines handling schedule perturbations caused by airline disruptions. Even though our proposed method can be applied to various types of airline disruptions, we focus on the temporary closure of airports, which affects 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 study are fourfold. First, we propose RL algorithms for solving the ARP for the temporary closure of the airport. Existing studies of the ARP utilize optimization or heuristic methodologies. This study, however, adopts the RL approach, which is an agent-based model, for utilizing the advantages of RL in aircraft recovery. Second, we solve the ARP to optimize various objectives: minimizing total delays and the number of flight delays of more than 30 min and more than 0 min. 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 RL algorithms and the existing solution approaches, such as optimization, meta-heuristic, and real-world strategies, and show the advantages of utilizing RL algorithms for ARP. Fourth, we apply the proposed method to a real-world flight schedule of a South Korean domestic airline with multiple fleets. In addition, we consider the constraint that the minimum TAT could be affected by disruption (i.e., TAT extensions).

The remainder of the study is organized as follows. Section 2 reviews literature related to the RL approach and the ARP. Section 3 describes the problem statement and mathematical formulation of the problem. Principles of RL, the environment of the flight schedule, and the Markov decision process (MDP) are presented in Section 4. Section 5 describes the QL and DQL algorithms. Section 6 shows the results of computational experiments to compare proposed RL algorithms with existing algorithms. Moreover, we evaluate the performance of our approaches by applying them to the real-world case of the South Korean domestic airline, and we suggest managerial insights. Conclusions are presented in Section 7.

2. Literature review

This paper is directly related to three streams of literature: the reinforcement learning approach for air transport management, the ARP in operations management, and the ARP dealing with temporary closures of airports.

The RL approach was adopted in several studies for air transport management. Gosavii et al. [19] formulated airline revenue management as a semi-Markov decision process (SMDP). They solved SMDP with a λ-SMART algorithm based on RL. Balakrishna et al. [20] incorporated RL for predicting taxi-out time in airports. Because airline operations dynamically change, the accurate prediction of taxi-out time is very challenging. Because of this property, the authors adopted RL, and the accuracy of their prediction was relatively high, even without detailed data. Hondet et al. [21] used the QL algorithm in disruption management of airline schedules. However, the performance of the QL algorithm was worse than it was without any controls, and they obtained no significant results. Ruan et al. [22] used the RL-based algorithm, specifically QL, to deal with the operational aircraft maintenance routing problem (OAMRP). By comparing meta-heuristic methods, the authors showed the outperformance of the proposed RL-based algorithm. Hu et al. [23] developed extreme learning machine-based QL, which learned the maintenance decision policies for different aircraft maintenance scenarios. The developed algorithm required only a little information about aircraft maintenance and showed prior performance in adjusting its decision for addressing the variations of several components. Shihab et al. [24] utilized a deep RL algorithm for solving the airline revenue management problem with multiple fare classes with stochastic demand, passenger arrivals, and booking cancellations. Kravaris et al. [25] dealt with the problem of demand and capacity imbalances in actual air traffic management settings with many agents. The deep multi-agent RL method was utilized to alleviate the challenges of scalability and complexity in the suggested problem. In addition, they developed visual analytic tools for rendering information provided by the RL method. Pham et al. [26] utilized the RL approach to resolve conflicts and inherent uncertainties in air traffic. They formulated the conflict resolution task as a sequential decision-making problem. The deep deterministic policy gradient algorithm was used to handle a large and complex action space.

We investigated previous studies related to the ARP in operations management. The ARP was first introduced by Teodorović and Guberinić [27]. They solved a simple example of the ARP with a heuristic, which decided aircraft routes sequentially. Yan and Yang [28] 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. [12] added protections arcs and through-flight arcs to the time-line network model. This method makes it possible to prevent the original flight schedule from perturbations. Løve et al. [29] 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. [30] formulated a set partitioning model to reschedule flight legs and reroute multiple fleet aircraft. In order to efficiently determine a subset of aircraft for rerouting, they developed the aircraft selection heuristic. Khaled et al. [31] presented a multi-criteria repair/recovery framework using multi-objective integer programming for the tail assignment problem.

Liang et al. [32] solved the ARP with airport capacity constraint and maintenance flexibility. In order to consider these conditions, they adopted a column generation based heuristic framework. Vink et al. [33] proposed a heuristic that dynamically solved the recovery of the airline schedules. In this manner, recovery problems for subsequent disruptions are solved based on the previously obtained solutions. Hu et al. [34] proposed an integer programming model to simultaneously consider the airline cost and passengers' willingness in the ARP. The proposed model addressed two objectives, minimizing airline recovery cost and passenger recovery loss. The authors developed a heuristic by combining multi-directional and stochastic neighborhood search algorithms to solve the problem. Huang et al. [15] focused on mitigating the issue that occurred when generating flight copies for solving the ARP. The authors developed the copy evaluation algorithm to solve the ARP through an iterative process of copy generation and filtration. Zhao et al. [14] addressed the ARP with two types of uncertainty in airline disruptions: the length of the disruption and the time when additional information became available. The two-stage and rolling horizon approaches were developed, and two types of uncertainty were handled by considering an extensive range of scenarios.

Several researchers have considered the temporary closure of airports in the ARP. Yan and Lin [35] conducted one of the pioneering studies on this problem. They adopted similar methods proposed by Yan and Yang [28]. They used a time-line network with four types of arcs: flight, ground, overnight, and ferrying. Thengvall et al. [36] 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. Liu et al. [18] 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 adapt an MMGA to multi-fleet airline schedules.

Liu et al. [18] studied the most relevant problem to our research and also considered various objectives such as flight delays of more than 30 min. In order to get admissible Pareto optimal solutions, this study proposed an 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 TAT and flight connections at airports) to meet multiple objectives for hard constraints. Therefore, it was difficult to consider complex realistic conditions, such as multiple fleets and 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, an MMGA found the suboptimal solutions.

To overcome the limitations of previous studies, we propose the RL approach, which has generated a lot of interest from the research community. We interpret the ARP as a sequential decision-making process and improve the behavior of the agent by trial and error. There are four main advantages to using RL for the ARP. First, because RL is a simulation-based method, it can handle complex assumptions [19]. 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), RL could solve the ARP under realistic situations. Second, by just modifying reward functions, RL is more flexible in meeting various objectives than are operations research methods. Third, since RL 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 [37]. Fourth, the time-line network mathematical model, which is one of the most comprehensive and practical approaches for the ARP, can suffer from the trade-off between the quality of the recovery schedule and the computation time. In contrast, we solve this challenge by developing the environment of RL to make decisions in discrete minutes.

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 TAT of each aircraft type is different. The minimum TAT depends on the size of the aircraft. The bigger the aircraft, the longer the TAT. Considering the above characteristics, we set the following two constraints.

(1) Each aircraft can be swapped with the aircraft that belongs to the same subfleet.
(2) The TAT 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 a 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 TAT: When an aircraft lands at the destination airport, a certain amount of time is necessary for preparing the next flight (e.g., runway taxiing, cabin cleaning, refueling, catering). This time is defined as TAT. The ground times of the aircraft for consecutive flights must be longer than a minimum TAT.
(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. Fig. 1 represents the example of disruptions that 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 TAT is increased for a certain period from the reopening time of the airport. We refer to this period as the 'extension period' and the increased minimum TAT as the 'extension time.' To reflect this problem, we assumed that the TAT increased (extension time) for a certain period of time (extension period) at a closed airport, and we used the term 'TAT extension' to refer to this condition. For example, assume the situation that the airport is closed from 2:00 p.m. to 3:00 p.m., and the minimum TAT of the aircraft is 30 min. If the extension period is set to three hours and the extension time is set to 15 min, every aircraft planned to depart between 2:00 p.m. and 5:00 p.m. from the disrupted airport must satisfy the extended minimum TAT, 45 min.

As mentioned in Section 1, there are four types of recovery options that minimize the damage of disruptions as much as possible. In real practice, flight cancellations and ferrying are used as a last resort because these options incur a lot of cost [38]. Because of these reasons, many research studies did not consider cancellation and ferrying as recovery options [39–41]. We employ the following recovery options, except for cancellation and ferrying.

(1) Delaying flight departures: The departure times of subsequent flights should be delayed until the connected resources are ready (e.g., satisfaction for minimum TAT). 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. In daily schedules, each aircraft is assigned to a sequence of flights (i.e., aircraft routes). Therefore, swapping aircraft is equivalent to swapping the routes of aircraft, which means that swapped aircraft have to complete one another's remaining flights. To show the advantages of this option, the simple example of swapping aircraft is illustrated in Fig. 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 ATA of 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 TAT 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. Therefore, with the aircraft swapping, the AOCC can avoid departure delays of f_3.

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 overall daily flight schedule. This is one of the most common objectives that existing studies used.
(2) To minimize the number of flight delays of more than 30 min (Case B): Flight delay is a significant measure for evaluating an airline's on-time performance. Throughout this study, we note flight delay when the actual event time is late by more than 30 min, compared to the scheduled event time in accordance with the regulation of the South Korean government.
(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 the 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 [42].

3.3. Assumptions

We accommodate the following major assumptions used within the modeling framework:

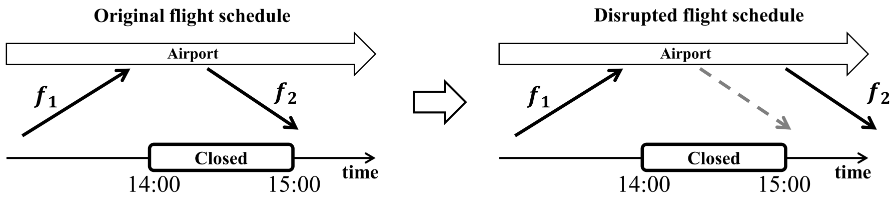


Fig. 1. Disruptions by temporary closures of airport.

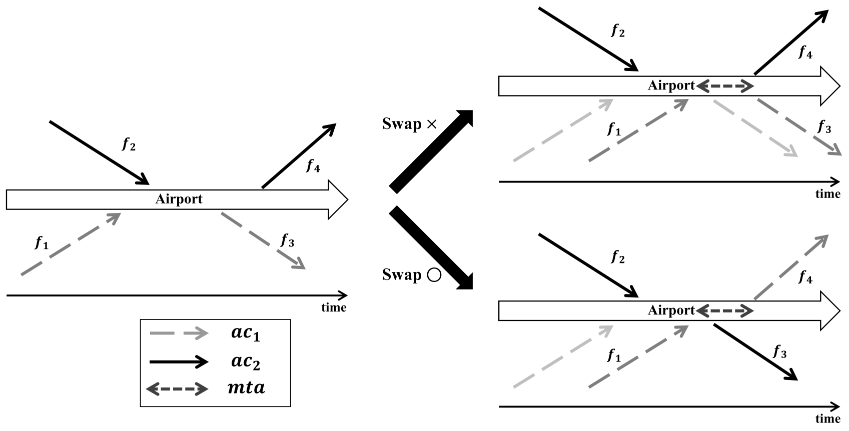


Fig. 2. Example of swapping aircraft.

(1) The flight schedule is a daily schedule. The 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 minimum TAT of each aircraft is different, depending on the type of aircraft.
(5) Flight cancellations and ferrying are not allowed.
(6) The flying time between each airport is fixed.
(7) Every slot (e.g., landing and takeoff) is assumed to be available.
(8) The event times of flights planned to land on or depart from a closed airport are postponed until the reopening time.
(9) Although aircraft at a closed airport are in the air, aircraft cannot land at the closed airport until the reopening time.
(10) 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.
(11) If the aircraft takes the last flight in the sequence of flights in the scheduled (initial) route, it cannot be swapped with other aircraft. Therefore, this aircraft finishes the daily schedule.
(12) The sequence of flights in every aircraft route must be unchanged after swapping aircraft.

The above main assumptions were utilized to formulate the mathematical model in Section 3.4, and we designed the environment of flight schedule operation for the RL approach based on these assumptions in Section 4.3. Assumptions (1)–(10) are the same as those in Liu et al. [18]. These assumptions were used to reflect the situation that aircraft operation is disrupted by airport closures. In addition, several studies also adopted these assumptions to simplify the problem [39,43].

We added Assumptions (11) and (12) for the following three reasons. First, Assumption (11) prevents particular aircraft from taking excessive flights. Second, because of Assumptions (1) and (11), the aircraft balance requirement is satisfied. According to Assumption (1), if each aircraft takes the last flight of initial routes at the end of the daily schedule, the aircraft balance is satisfied. Because Assumption (11) ensures that the last flights of initial routes are carried out the latest by each aircraft, the aircraft balance requirement is satisfied. Third, Assumption (12) can reduce action space for the RL algorithm. In the early stage of this research, we did not account for Assumption (12), 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 (12). Therefore, in this study, the standard meaning of the 'swappable condition' is used to indicate the status of the aircraft that satisfy Assumption (12). 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 the flight schedule, it is required that they be classified according to whether they satisfy Assumption (12) or not. A detailed explanation for generating the candidates of aircraft that satisfy the swappable condition is described in Appendix A.

3.4. Mathematical formulations

The time-line network mathematical model is one of the most comprehensive and practical approaches for the ARP that utilizes copies of flight arcs. This model is formulated based on network flow, and a simple example of the time-line network is illustrated in Fig. 3. A single time-line network is utilized for a single subfleet, as illustrated in Fig. 3. In order to take into account a multi-fleet condition, we utilize $|P|$ number of time-line networks where P denotes the set of subfleets. There are three types of nodes: supply (S), termination (T), and intermediate (I) nodes. The larger nodes shown in Fig. 3 are the supply and termination nodes. The supply nodes supply aircraft at the beginning of the

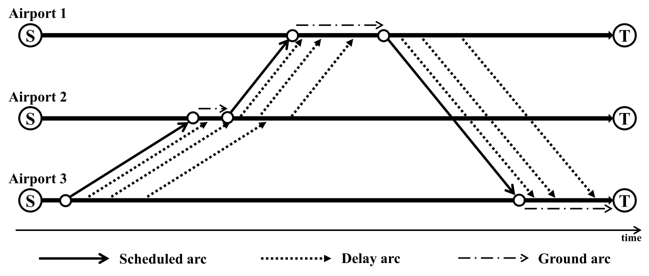


Fig. 3. Simple example of time-line network structure for a single subfleet.

time horizon of recovery, and the 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 originally 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. For instance, three delay options are available for each of the three flights in Fig. 3. Therefore, in this example, the set of arcs for each flight ($N(f)$) contains four flight arcs: one scheduled arc and three delay arcs. At last, we define the function $h : N \times P \rightarrow F$ indicating the flight f covered by the flight arc n and for subfleet p.

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
$t_{h(n,p)}$	Initial scheduled departure time of flight $h(n,p)$
$\tilde{t}_{np}$	Actual departure time of flight $h(n,p)$ altered by flight arc n for subfleet p

Decision variables

x_{np}	Flow on flight arc n for subfleet p
z_{gp}	Flow on ground arc g for subfleet p

The following is the time-line network mathematical formulation for the ARP (TLN). The model is developed based on the mathematical formulation proposed by Thengvall et al. [36].

TLN

$$\min \sum_{p\in P}\sum_{n\in N} c_{np}x_{np} \tag{1}$$

$$\text{s.t.} \sum_{g\in G\cap O^+(s,p)} z_{gp} + \sum_{n\in N\cap O^+(s,p)} x_{np} = b_{sp}, \quad \forall s\in S, \forall p\in P \tag{2}$$

$$\sum_{g\in G\cap O^+(i,p)} z_{gp} - \sum_{g\in G\cap O^-(i,p)} z_{gp} + \sum_{n\in N\cap O^+(i,p)} x_{np} - \sum_{n\in N\cap O^-(i,p)} x_{np} = 0, \quad \forall i\in I, \forall p\in P \tag{3}$$

$$\sum_{g\in G\cap O^-(t,p)} z_{gp} + \sum_{n\in N\cap O^-(t,p)} x_{np} = -b_{tp}, \quad \forall t\in T, \forall p\in P \tag{4}$$

$$\sum_{p\in P}\sum_{n\in N(f)} x_{np} = 1, \quad \forall f\in F \tag{5}$$

$$x_{np} \le x_{\nu p}, \quad \forall r\in R, \forall p\in P, \forall n,\nu\in N(r) : (t_{h(n,p)} \le t_{h(\nu,p)}) \wedge (\tilde{t}_{\nu p} < \tilde{t}_{np}) \tag{6}$$

$$x_{np} \in \{0,1\}, \quad \forall n\in N, \forall p\in P \tag{7}$$

$$0 \le z_{gp} \le u_{gp}, \quad \forall g\in G, \forall p\in P \tag{8}$$

$$z_{gp} \in \mathbb{Z}, \quad \forall g\in G, \forall p\in P \tag{9}$$

The objective function 1 is minimizing the total flight delays incurred by disruption. The c_{np} is modified according to the corresponding objective. Constraints represent the aircraft balance constraints. Constraint 3 is the balance equation for intermediate nodes. Constraint 5 is the flight cover constraint, which ensures that every flight is implemented at the scheduled time or delayed. Constraint 6 is the route sequence constraint, which ensures Assumption (12). Constraint 7 is a binary constraint. Constraint 8 is the capacity constraint for ground arcs. Constraint 9 ensures that the flow of ground arcs is integer.

There are three weak points in this model. First, without Constraints 5 and 6, this problem is equivalent to the single commodity flow problem, which is a well-solved problem. However, the single commodity flow problem becomes an NP-hard problem by adding the side constraint, Constraint 5 [44]. Second, a trade-off between quality of recovery schedule and computation time exists because creating delay arcs is necessary to account for delaying flight departure. The more delay arcs could guarantee better solution quality while the problem size increases. Furthermore, when there were not enough delay arcs, a feasible solution could not be guaranteed. 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 route-based solutions.

Unlike the methods mentioned above, we interpret the ARP as sequential decision-making and develop an MDP model for this problem. In addition, we utilize the RL algorithms to find the 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. By using the MDP model and RL algorithms in the ARP, the proposed solution approach does not depend on copies of flight arcs, which can be the answer to the research question (1).

4. Reinforcement learning for aircraft recovery

Recently, many optimization methods have been developed to solve decision problems. A lot of exact algorithms (e.g., dynamic programming, LP-based algorithms, and decomposition algorithms) are widely used to derive the optimal solution, and a

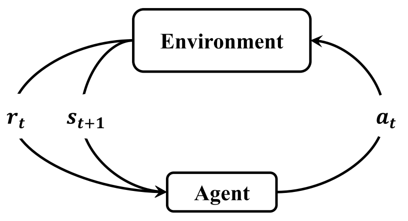


Fig. 4. Framework of reinforcement learning.

commercial optimization solver has been improved to solve the different types of problems. However, there are many challenging decision problems (i.e., NP-hard problems) in the optimization research area. Even though the computing power has increased, it is still difficult to find a 'good' solution for NP-hard problems within the proper computation time. In several real cases (e.g., a routing problem in the navigation business or an evacuation problem after a disaster), it is more important to get a suboptimal solution within the proper times than to derive an optimal solution by spending a lot of time in computation. To address these issues, advanced optimization algorithms, such as heuristics and metaheuristics, have been developed and widely used in many decision problems. Not only have heuristics and metaheuristics been specialized in decision problems, they have also been utilized in many different domains, such as in the machine learning research area [45–47].

The RL approach, which belongs to the machine learning technique, directly mitigates two potential issues of classical dynamic programming (i.e., the curse of dimensionality and the curse of modeling) when solving large-scale problems [48]. Many operation management and combinatorial optimization problems have been solved using RL algorithms within the reasonable computation times [49,50]. Also, the RL approach has achieved great success in solving complex sequential decision-making problems, and many researchers in operations management take a profound interest in the RL approach. This solution approach has been widely employed in various domains: energy management, transportation, and health care [51–53]. As indicated in Section 3.4, the ARP is classified as a challenging decision problem. Motivated by the above successes of RL, we adopt RL as our main solution approach to solve the ARP and to get a 'good' solution within the proper computation times.

Being able to apply the RL to aircraft recovery, environment, agent, and MDP adequate to the given problem is 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

RL is an agent-based approach to finding 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. Fig. 4 depicts how the agent interacts with the environment. In this study, 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.

Because RL 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 [54]. 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 RL 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:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|A(s)|} & \textit{if } a = \operatorname{argmax}_a Q(s, a) \quad \text{(a)} \\ \frac{\varepsilon}{|A(s)|} & \textit{otherwise} \qquad \text{(b)} \end{cases} \tag{10}$$

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 [54]. This method biases the initial action-value estimates and ensures extensive exploration. In this study, 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.

4.2. Environment

We built an artificial environment of flight schedule operation based on the assumptions in Section 3.3. Fig. 5 represents the structure of the environment. One execution of the flowchart in Fig. 5 is defined as one episode for the RL approach. There are three principal elements in the proposed environment. The first element is aircraft routes. There are two types of routes: the scheduled and the 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 comprises 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,

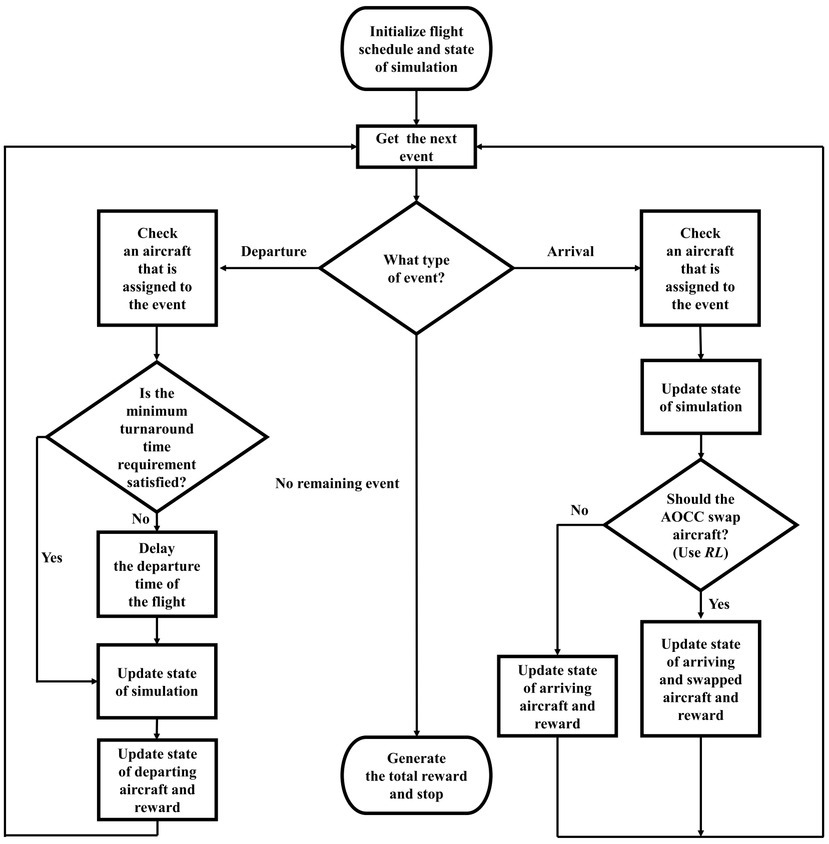

Initialize flight schedule and state of simulation
Get the next event
What type of event?
Departure
Arrival
Check an aircraft that is assigned to the event
Check an aircraft that is assigned to the event
Update state of simulation
Is the minimum turnaround time requirement satisfied?
No remaining event
No
Yes
Delay the departure time of the flight
Should the AOCC swap aircraft? (Use RL)
No
Yes
Update state of simulation
Update state of departing aircraft and reward
Generate the total reward and stop
Update state of arriving aircraft and reward
Update state of arriving and swapped aircraft and reward

Fig. 5. Structure of the environment.

the earliest one to occur is first designated to the time step, and a single aircraft is assigned to the corresponding event. 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. In other words, there are $|F|$ decision points for each day of operation. 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 TAT requirement, the AOCC delays the actual departure time until the minimum TAT is met. In contrast to TLN, the proposed RL framework always guarantees a feasible solution, owing to the above mechanism.

4.3. Markov decision process

The problem of RL can be formalized as MDPs, which is a mathematical formulation for sequential decision making. The 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. The MDP model is utilized to simulate the behavior of the system under the environment of flight schedule operation, which is presented in Section 5.2 [55].

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: $s = (u_1, \dots, u_{|C|}, p_1, \dots, p_{|C|}, b_1, \dots, b_{|C|})$

Because the number of routes is equal to the number of aircraft in the flight schedule, sets of previous and future route are as follow: $U = \{1, 2, 3, \dots, |C|\}$, $P = \{1, 2, 3, \dots, |C|\}$. 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: $I = \{1, 2, 3, \dots, 2|F|\}$. 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 $(2|C|)^{|C|+1}|F \parallel C - 1|!$. 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 study, 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: $A = \{1, 2, \dots, |C|\}$. However, valid action space size is one at the departure event since the AOCC can swap aircraft at the arrival event. Moreover, at the arrival event, the action space can be reduced due to the swappable condition. We provide a small example of states changed by the actions in Appendix B to help readers to understand easier.

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 RL 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 f ($r_t = \min\{t_f^d - \tilde{t}_f^d, 0\}$). In Case B, if the departure delay of flight f is more than 30 min ($\tilde{t}_f^d - t_f^d > 30$), reward r_t is -1, otherwise 0. In Case C, if the departure delay of flight f is more than zero minutes ($\tilde{t}_f^d - t_f^d > 0$), reward r_t is -1, otherwise 0. By simply altering reward functions for corresponding objectives, the RL approach can flexibly adapt to various objectives, which can be the answer to the research question (2).

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.

5. Reinforcement learning algorithms

In this section, we present RL algorithms for solving the ARP. Among the various RL algorithms, we adopt QL and DQL. Section 5.1 presents the concept and procedure of QL algorithm. Section 5.2 examines the overestimation bias problem of QL algorithm and presents DQL algorithm, which alleviates the overestimation bias.

5.1. Q-learning algorithm

As was stated in Section 4.1, the purpose of solving the MDP is that finding the optimal action-value function at the state and action pairs $[(s_t, i_t), a_t]$. Based on the classical dynamic programming, we can find the optimal action-value function by utilizing the Bellman optimality equation [56]. 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' [48]. In contrast with dynamic programming, model-free algorithms of RL do not need to calculate the transition probability. QL is the RL algorithm utilizing the model-free concept and employs Bellman equation and temporal difference learning [57]. Especially, temporal difference learning generalizes beyond the Robbins–Monro algorithm, which is a representative method in a stochastic approximation [58]. By approximating the optimal action-value, it is not necessary to compute the transition probability, and QL can avoid the curse of modeling.

Due to the property of the off-policy approach, QL 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, i_t) while learning with a target policy (i.e., greedy policy) for (s_{t+1}, i_{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, $Q[(s_t, i_t), a_t]$, is updated by the next states of the Q function, $Q[(s_{t+1}, i_{t+1}), a']$, with the greedy action. The learned Q function approximates the optimal action-value by updating with the following equation iteratively:

$$Q[(s_t, i_t), a_t] \leftarrow Q[(s_t, i_t), a_t] + \alpha \Big[r_t + \gamma \max_{a'} Q[(s_t, i_t), a'] - Q[(s_t, i_t), a_t] \Big] \tag{11}$$

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 states and actions. 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 the episode (ρ), which is a summation of observed rewards r_t in an episode, with the maximum total reward of the episode (ρ_{max}) and update the maximum value. The procedure of QL algorithm is presented as follows:

Algorithm 1 Q-learning.

```
*Initialize* $Q[(s, i), a]$, for all $s \in S, i \in I, a \in A$
$\rho_{max} \leftarrow -\infty$
**for** *each episode* **do**
    $\rho \leftarrow 0,\ t \leftarrow 0$
    **while** $i_t$ *is not the terminal state* **do**
        Choose $a_t$ using $\varepsilon$-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 QL has been successfully applied to many different applications, QL can sometimes overestimate action-value functions (the overestimation bias). The performance of QL 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 [59]. 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 the next state [60].

DQL can alleviate the overestimation bias by employing the double estimator, Q^A and Q^B. In contrast with QL, one of the Q functions in DQL is chosen randomly determined between Q^A and Q^B for separating sets of experiences. The updating process of

DQL uses the following equations:

$$Q^A[(s_t, i_t), a_t] \leftarrow Q^A[(s_t, i_t), a_t] + \alpha \Big[r_t + \gamma Q^B[(s_{t+1}, i_{t+1}), \operatorname*{argmax}_{a'} Q^A[(s_{t+1}, i_{t+1}), a']] - Q^A[(s_t, i_t), a_t] \Big] \tag{12}$$

$$Q^B[(s_t, i_t), a_t] \leftarrow Q^B[(s_t, i_t), a_t] + \alpha \Big[r_t + \gamma Q^A[(s_{t+1}, i_{t+1}), \operatorname*{argmax}_{a'} Q^B[(s_{t+1}, i_{t+1}), a']] - Q^B[(s_t, i_t), a_t] \Big] \tag{13}$$

In the process of updating the Q function, each Q is updated by the next state of the other Q function. In QL example, in cases in which Q^A is chosen, $Q^A[(s_{t+1}, i_{t+1}), a^A]$ is used for the next state value to update Q^A, where $a^A = \text{argmax}_{a'} Q^A[(s_{t+1}, i_{t+1}), a']$. However, in DQL, $Q^B[(s_{t+1}, i_{t+1}), a^A]$ 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 DQL, refer to Hasselt [60]. The procedure of DQL algorithm is presented as follows:

Algorithm 2 Double Q-learning.

```
*Initialize* $Q^A[(s, i), a]$, and $Q^B[(s, i), a]$, for all $s \in S, i \in I, a \in A$
$\rho_{max} \leftarrow -\infty$
**for** *each episode* **do**
    $\rho \leftarrow 0,\ t \leftarrow 0$
    **while** $i_t$ *is not the terminal state* **do**
        Choose $a_t$ using $\varepsilon$-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**
```

6. Computational experiments

In this section, we conducted three computational experiments to answer the research questions (3), (4), and (5) in Section 1. In Section 6.1, in order to measure the performance of our algorithms, we compare QL and DQL with MMGA, and TLN taking into account the same problem definition of Liu et al. [18]. In Section 6.2, 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., TAT extension and multiple fleet conditions). In Section 6.3, to verify proposed RL 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 the artificial environment was coded using Python 3 language. TLN was conducted with FICO Xpress 8.5 and Xpress-Optimizer version 33.01.02, which is a general-purpose mixed-integer program solver. Research questions (3), (4), and (5) are answered by Sections 6.1, 6.2 and 6.3, respectively. At last, we suggest several managerial insights that are helpful for airline operation decision-making in Section 6.4.

6.1. Comparison between reinforcement learning and existing algorithms

In this section, we employ the flight schedule presented in Liu et al. [18], which consists of four airports, 70 flights, and seven aircraft of a single fleet. For every aircraft, the minimum TAT is 30 min. Because of the property of a short-haul flight schedule, most of the time intervals between flight legs are equal to the minimum TAT. The experiments were conducted within the disruption scenario in which the Taipei Sungshan (TSA) airport was closed at 2:00 p.m. and reopened at 3:00 p.m. All of the flights scheduled to depart or arrive at TSA between 2:00 p.m. and 3:00 p.m. were delayed until the airport was reopened. In order to compare the performance of TLN, MMGA and our proposed RL algorithms, we excluded multiple fleet and TAT 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 QL and DQL, we implemented 50 learning runs with different random seeds. Although we could find a better quality of the solution with a long length of training episodes, we set the end of the episodes to 3,000. 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 study, 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 time is calculated at the convergence episode or the episode 3,000.

We utilized the 'Without Swapping' (WSAP) and MMGA for comparing the performance of QL and DQL. The WSAP is a recovery method in which the swapping aircraft is not considered for the recovery option. Thus, all the aircraft always were assigned to their initial scheduled routes. A supplementary explanation and pseudocode of WSAP is added to Appendix C. The MMGA, which was proposed by Liu et al. [18], 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 TAT and flight connection requirements, which decide whether the solution is feasible or not. The soft constraints are equivalent to the objectives of this study (i.e., Cases A, B, and C).

Detailed experiment results are illustrated in Table 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 time was obtained from averaging results of 50 runs with different random seeds. For Case A, DQL, QL, and TLN attained the best objective values, and except for WSAP, every solution method attained the same objective values for Case B. For Case

Table 1
Comparison of algorithms for the ARP.

Case	Method	OBJ[a]	Conv ep[b]	CPUs[c]	Gap[d] (%)
A	DQL	430.00	2,243.22	52.42	0.00
	QL	430.00	2,125.94	46.51	0.00
	TLN	430.00	–	127.13	0.00
	MMGA[e]	435.00	–	In minutes	1.16
	WSAP	525.00	1.00	0.03	22.10
B	DQL	6.00	2,096.36	49.22	0.00
	QL	6.00	1,855.92	40.20	0.00
	TLN	6.00	–	126.06	0.00
	MMGA[e]	6.00	–	In minutes	0.00
	WSAP	7.00	1.00	0.03	16.67
C	DQL	13.00	2,302.30	55.03	8.33
	QL	13.00	3,000[*]	67.09	8.33
	TLN	12.00	–	117.02	0.00
	MMGA[e]	14.00	–	In minutes	16.67
	WSAP	17.00	1.00	0.03	41.67

[a]OBJ : $-\rho_{max}$.
[b]Conv ep : Convergence episode.
[c]CPUs : Computation time in seconds.
[d]Gap (%) $:= \frac{\text{OBJ by each method} - \text{OBJ by TLN}}{\text{OBJ by TLN}} \times 100$.
[e]The results of MMGA are referred from Liu et al. [18].
[*]End of the episode was reached.

Table 2
Aircraft information of the flight schedule of a Korean domestic airline.

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

C, TLN attained the best objective values compared to other methods. However, DQL and QL took much less computation time for solving the problems compared to TLN and MMGA for every objective. Although WSAP could solve in 0.03 s, the quality of the solution was much poorer than the solution of other algorithms. The proposed RL approach could finish computing around one minute in every experiment, but TLN and MMGA required several minutes to solve the problem. Moreover, the objective values of QL and DQL were always the same regardless of different random seeds.

6.2. 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 South 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). The reasons why this study implemented a case study in South Korea, and the detailed description of the input data set were discussed in Appendix D. Table 2 presents the information of the number of aircraft, the minimum TAT, and the types of aircraft that could be swapped for each type of aircraft. The aircraft consisted of four types, and the minimum TAT 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 TAT extension and multiple fleet conditions in this experiment. We compared the performance of QL, DQL, and WSAP, without TLN and MMGA, which cannot be applied to the TAT extensions and multiple fleet conditions.

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 3. 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. Considering the characteristics of airports in South 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 TAT extensions.

The detailed results for every scenario are presented in Table 4. The experimental results were obtained by averaging the results of five runs with different random seeds. We could note that QL and DQL outperformed WSAP, and especially DQL showed the best performance in terms of objective values. Furthermore, DQL and QL completed learning within two minutes in all experiments, but two out of 30 (i.e., DQL for Cases B and C in Scenario 4). Even in these two experiments, the computation time was close to two minutes. Therefore, utilizing the RL algorithms is appropriate for real-time decisions based on the fact that it is recommended that the process should be less than three minutes for real-time decisions in actual practice [29].

Fig. 6 shows how the length of the closed period affects the flight operations depending on the traffic volume in the airport. As the length of the closed period increased, the flight schedule was affected severely at every airport. In particular, CJU (high traffic volume) was affected much more severely compared to GMP (middle traffic volume), and CJJ (low traffic volume) as the scale of disruption became large. For example, 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).

Fig. 7 shows the learning curves of QL, DQL, and WSAP for Cases A, B, and C in Scenario 2. The solid line depicts the learning curve of QL. The dashed line depicts the learning curve of DQL, and the flat dotted line depicts the objective value of WSAP. The learning curves were obtained by averaging the value of ρ of five learning runs with different random seeds. We set the length of the training episode to 10,000 to compare the performance of DQL and QL in detail. Considering the experiment for the flight schedule of a Korean domestic airline, QL suffered from the convergence problem due to the overestimation bias. However, DQL alleviated challenges of convergence compared to QL. In Case B, DQL, QL, and WSAP had the same result of objective value, one, and DQL converged faster than QL. However, in Cases A and C, QL could not learn a policy that improves the value of ρ continuously until the end of training episodes, and the value of ρ diverged. On the other hand, DQL showed steady improvement and learned a policy that led to the best solution quality among the comparison algorithms. Also, in contrast with QL, the ρ of DQL converged to the lowest value.

6.3. 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 three DQL with different reward functions. Throughout

Table 3
Information of disruption scenarios.

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

Table 4
Comparison of algorithms for the flight schedule of a Korean domestic airline.

Scenario	Method	Case A			Case B			Case C		
		OBJ	Conv ep	CPUs	OBJ	Conv ep	CPUs	OBJ	Conv ep	CPUs
1	DQL	120.00	3,000*	116.22	1.00	1,202.80	45.32	8.00	3,000*	117.38
	QL	119.00	3,000*	109.80	1.00	2,060.80	73.76	8.00	3,000*	109.32
	WSAP	135.00	1.00	0.05	1.00	1.00	0.05	10.00	1.00	0.05
2	DQL	154.20	2,981.20	116.97	1.00	1,202.80	45.37	11.00	3,000*	118.51
	QL	161.00	3,000*	110.06	1.00	2,060.80	73.84	11.80	3,000*	109.73
	WSAP	160.00	1.00	0.05	1.00	1.00	0.05	13.00	1.00	0.05
3	DQL	1,128.00	1,351.20	47.43	16.00	2,571.60	92.67	24.00	3,000*	109.07
	QL	1,128.00	1,264.20	41.78	16.00	3,000*	102.19	24.00	2,569.20	86.90
	WSAP	1,310.00	1.00	0.05	19.00	1.00	0.05	27.00	1.00	0.05
4	DQL	1,545.00	2,393.00	96.28	19.00	3,000*	121.62	29.00	3,000*	122.56
	QL	1,545.00	3,000*	114.56	19.00	3,000*	114.39	29.00	3,000*	115.02
	WSAP	1,765.00	1.00	0.05	22.00	1.00	0.05	32.00	1.00	0.05
5	DQL	145.60	2,737.80	95.69	2.00	1,739.40	59.53	8.00	3,000*	105.56
	QL	142.00	3,000*	98.16	2.00	576.20	17.92	8.00	3,000*	98.60
	WSAP	385.00	1.00	0.05	6.00	1.00	0.05	12.00	1.00	0.05
6	DQL	459.20	3,000*	114.72	6.00	3,000*	114.53	12.00	3,000*	114.94
	QL	481.00	3,000*	107.15	6.00	673.60	22.96	12.60	3,000*	107.30
	WSAP	695.00	1.00	0.05	10.00	1.00	0.05	16.00	1.00	0.05
7	DQL	888.00	3,000*	114.27	7.00	3,000*	114.25	19.00	3,000*	114.74
	QL	918.00	3,000*	107.07	7.80	3,000*	106.57	19.00	3,000*	107.03
	WSAP	1,235.00	1.00	0.05	12.00	1.00	0.05	20.00	1.00	0.05
8	DQL	90.00	3,000*	99.35	0.00	3,000*	99.02	7.00	2,935.60	97.04
	QL	90.00	3,000*	92.65	0.00	640.00	18.86	7.00	3,000*	92.57
	WSAP	140.00	1.00	0.00	0.00	1.00	0.05	11.00	1.00	0.05
9	DQL	147.00	2,919.40	105.81	1.00	3,000*	108.65	8.80	2,917.40	105.98
	QL	165.00	3,000*	101.88	1.00	698.40	22.76	9.00	3,000*	101.61
	WSAP	235.00	1.00	0.05	3.00	1.00	0.05	12.00	1.00	0.05
10	DQL	234.00	3,000*	116.04	1.60	3,000*	115.93	8.00	3,000*	116.29
	QL	237.00	3,000*	108.58	1.20	3,000*	108.74	10.00	3,000*	108.35
	WSAP	320.00	1.00	0.05	3.00	1.00	0.05	12.00	1.00	0.05

*End of the episode was reached.

this study, we use the terms 'DQL for Cases A, B, and C' to indicate DQL algorithm that applies appropriate reward functions for each objective. 'DQL for Case A' applies $\min\{t_f^d - \tilde{t}_f^d, 0\}$ for reward function and the others apply different reward functions defined for each objective (see Section 4.3).

Table 5 shows the performance of DQL for Cases A, B, and C in every objective. The objective values were obtained by averaging the results of five learning runs at the convergence episode. DQL with appropriate reward functions outperformed other reward functions for customized objectives in all but three experiments out of 30. In particular, DQL for Case A had the lowest objective value for all Scenarios for the objective of Case A. In Case B, DQL for Case B performed best in all Scenarios except for Scenario 10. DQL for Case C showed the best performance in all Scenarios except Scenarios 2 and 9 in terms of Case C.

For further comparison, we examined the number of the best performing DQL for Cases A, B, and C for every objective. Fig. 8 presents a comparative analysis of DQL with different reward functions for every objective. With the setup of the same random seed and scenario, we compared DQL for Cases A, B, and C, and the method obtained the best objective value was counted as the number of 'best performing' each objective. In the case in which several DQL algorithms obtained the same best objective value, all those algorithms were considered for the category of the best performing. If an algorithm showed the best performance for every learning run in ten scenarios, the best performing number was set at 50 (i.e., 5 × 10). As anticipated, our experiment showed that DQL with the appropriate reward function for each objective performed best for each objective. This result means that DQL with the appropriate reward function performed best on each

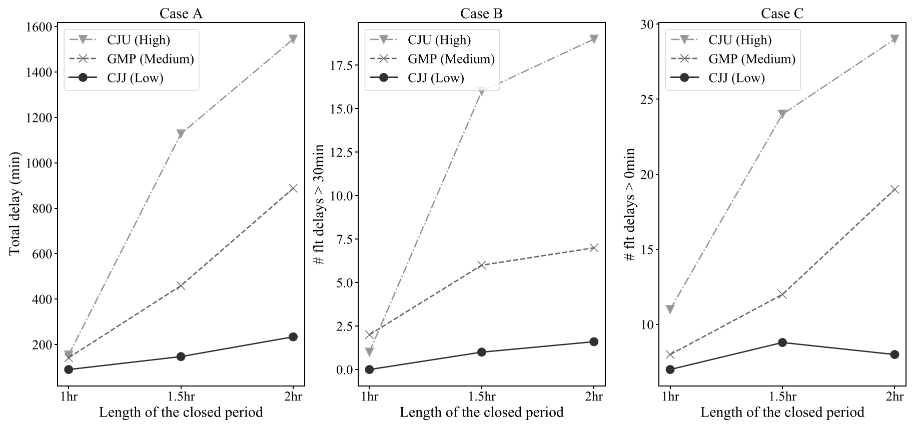


Fig. 6. Impacts of disruption depending on the traffic volume in the airport.

Table 5
Comparison of the performance of DQL with different reward functions for each objective.

Scenario	DQL for Case A			DQL for Case B			DQL for Case C		
	Case A	Case B	Case C	Case A	Case B	Case C	Case A	Case B	Case C
1	120.00	1.00	8.00	216.00	1.00	13.20	120.00	1.00	8.00
2	154.20	1.60	10.20	246.00	1.00	15.20	206.00	2.00	11.00
3	1,128.00	18.00	27.00	1,310.00	16.00	28.60	1,364.00	19.40	24.00
4	1,545.00	19.00	29.00	1,601.00	19.00	31.40	1,630.00	20.00	29.00
5	145.60	2.00	8.00	211.00	2.00	11.00	151.00	2.00	8.00
6	459.20	6.00	12.00	545.00	6.00	14.80	490.00	6.00	12.00
7	888.00	7.00	21.00	904.00	7.00	22.20	1,062.80	10.40	19.00
8	90.00	0.00	7.00	164.00	0.00	11.60	90.00	0.00	7.00
9	147.00	1.00	8.20	226.00	1.00	12.80	157.00	1.00	8.80
10	234.00	1.00	10.20	323.00	1.60	14.00	260.00	3.00	8.00

objective when trained specifically to optimize for the corresponding objective. Especially, DQL for Case A showed robustness in terms of performance for any type of objective.

6.4. Managerial insights

This study 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 RL 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 framework modeled with the RL approach could help AOCC, the final user of the proposed algorithms, make better decisions.
(2) QL 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 RL, QL could show poor performance because of the overestimation bias. In such cases, therefore, we showed that utilizing DQL would be an effective strategy.
(3) In this study, we suggested three objectives and defined reward functions for each objective. The RL approach is configurable to different objectives. Based on the conducted experiments, DQL with an appropriate reward function for each objective outperforms other reward functions. Therefore, if airline operations controllers want to meet the objectives suggested by our study, they could utilize the proposed appropriate reward functions. On the other hand, if airline controllers want to implement aircraft recovery to perform well for overall objectives, we recommended that adopting the reward function for minimizing total delays would be an effective strategy.

7. Conclusions

Implementing aircraft recovery to minimize the damage from disruptions is significant for an airline's bottom line, in terms of both customer satisfaction and operations costs. In the aircraft recovery process, 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 RL, specifically QL and DQL, for the ARP. Among various types of air transportation disruption, we concentrated on airport closures, which affect many flight operations. Note that our proposed methods can be applied to other disruption types, such as aircraft unavailability, flight delay, and crew and/or passenger caused delays, by revising the disruption scenarios and inputting them to the developed environment.

We developed the environment of daily flight schedules and defined the states, actions, and reward functions of MDP for implementing the proposed RL algorithms. In particular, the three different reward functions were defined for each objective. We evaluated the performances of QL and DQL, and compared the results with TLN, MMGA, and WSAP. Through conducting computational experiments, we could observe the outperformance of the proposed RL algorithms compared to other methods in terms of computation time. Furthermore, in the real-world flight schedule of one of the domestic airlines in South Korea, we validated the advantages of utilizing DQL, which alleviates the

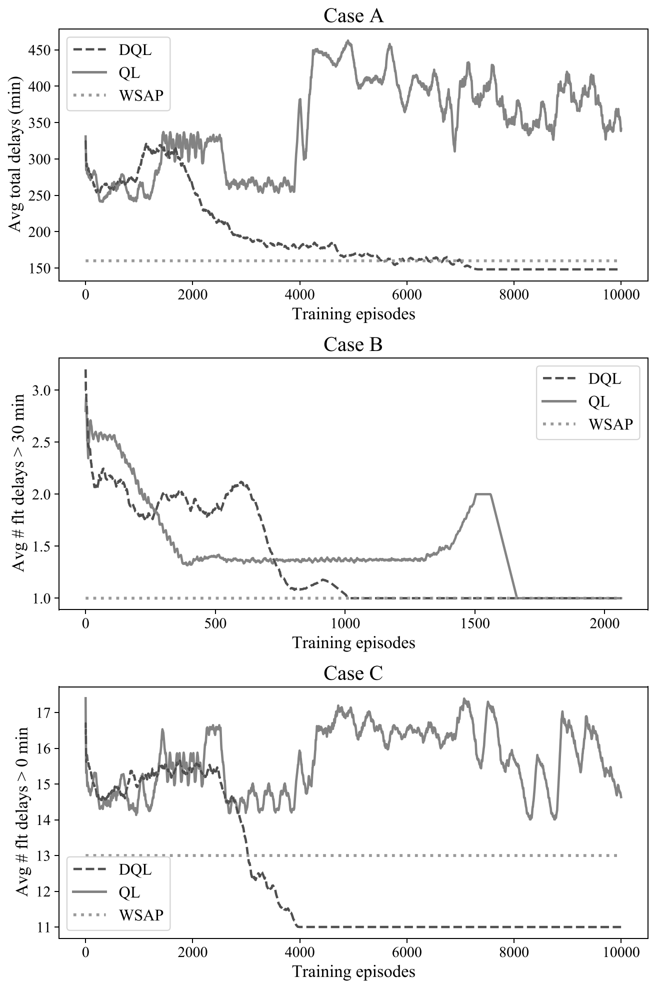


Fig. 7. Learning curves of algorithms in Scenario 2.

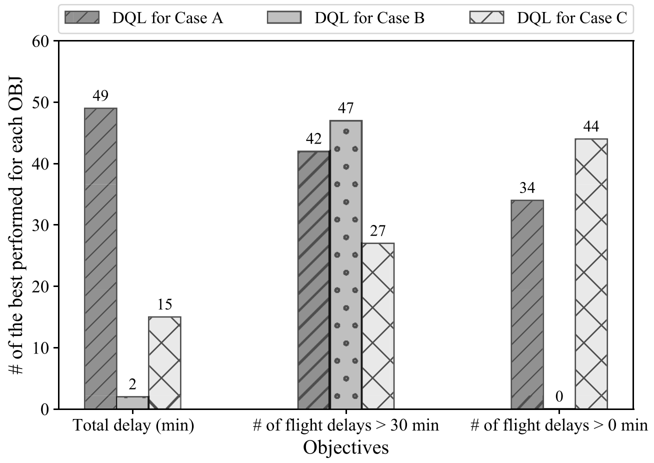


Fig. 8. Performance for different objectives.

overestimation bias of QL. We analyzed the impact of disruption depending on the traffic volume in the airport. In addition, our computational experiments in this study showed that DQL with appropriate reward functions outperformed other reward functions in customized objectives.

To the best of our knowledge, this study is a first step to applying the RL approach to the ARP, since most existing studies adopted operations research methods instead. The presented framework with the RL approach has three distinctive features that differentiate it from previous studies. First, re-timing certain flights and swapping aircraft can be implemented by the RL approach without relying on copies of flight arcs. Second, by just modifying reward functions, the RL agent can flexibly adapt to various objectives. Third, because the RL algorithms are carried out through the simulation (i.e., environment), complex real-world conditions can be handled without heavily relying on mathematical models.

The present findings suggest several managerial implications for the AOCC staff who manage disruptions of airline flight schedules. Along with existing optimization-based tools and the practical knowledge of AOCC staff, the proposed RL framework could help the final user to make better decisions for aircraft recovery. We expect our approach to serve as a bridge for utilizing artificial intelligence technology, specifically RL, in airline disruption management.

The RL framework developed in this study has several limitations to address in future research, as follows:

- An important issue to resolve for future studies is to derive general policy for various airline disruptions. Because the states and actions in the RL framework were defined based on aircraft routes, it was difficult to reuse the obtained policy for new problem instances. By redefining states and actions and by utilizing suitable functions (e.g., a neural network) to approximate the action-value function, the agent could learn a general policy that can be utilized for new problem instances. In this manner, the RL approach may derive general policy, which could be applied to many disruption instances.
- The current study was not designed to prevent unnecessary swapping of aircraft. However, an airline's engaging in non-profitable duty swaps for flights could cause trouble in changing crews and could consume other resources. Unnecessary swaps, therefore, might be regarded as measures of deviation from original flight schedules. Future studies on the current topic are therefore suggested in order to revise reward functions that take into account the additional cost of swapping aircraft.
- The present study has only considered aircraft among many airline resources (e.g., crews and passengers) for recovery of schedules reacting to airline disruptions. The proposed approach can be extended to the integrated recovery of aircraft, crews, and passengers. An advanced artificial environment should be developed to incorporate the complex operations of these resources.

CRediT authorship contribution statement

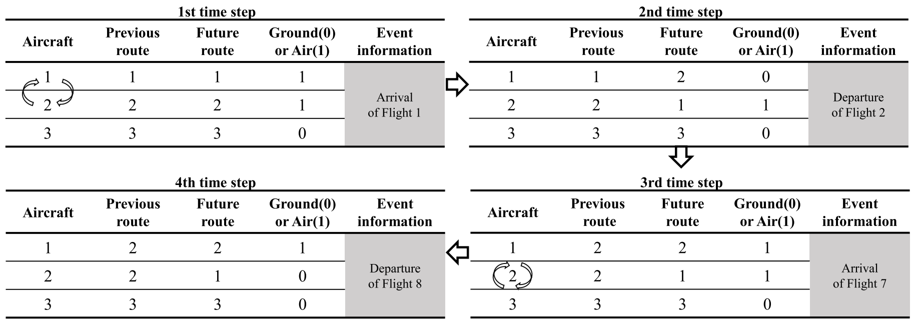
Junhyeok Lee: Conceptualization, Methodology, Investigation, Software, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Kyungsik Lee:** Conceptualization, Methodology, Writing – review & editing. **Ilkyeong Moon:** Conceptualization, Validation, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.



1st time step

Aircraft	Previous route	Future route	Ground(0) or Air(1)	Event information
1	1	1	1	Arrival of Flight 1
2	2	2	1	
3	3	3	0	

2nd time step

Aircraft	Previous route	Future route	Ground(0) or Air(1)	Event information
1	1	2	0	Departure of Flight 2
2	2	1	1	
3	3	3	0	

3rd time step

Aircraft	Previous route	Future route	Ground(0) or Air(1)	Event information
1	2	2	1	Arrival of Flight 7
2	2	1	1	
3	3	3	0	

4th time step

Aircraft	Previous route	Future route	Ground(0) or Air(1)	Event information
1	2	2	1	Departure of Flight 8
2	2	1	0	
3	3	3	0	

Fig. B.9. Example of states changed by the actions of the agent.

Acknowledgments

The authors are grateful for the valuable comments from the associate editor and three anonymous reviewers. This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning [Grant no. NRF-2021R1A4A1029124].

Appendix A. Generation of list of aircraft satisfying the swappable condition

Let mta_c denote the minimum TAT 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 3 Generation of the list of swappable aircraft.

```
$S \leftarrow$ empty list
$c \leftarrow$ an aircraft 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
    $mta_i \leftarrow$ minimum TAT of aircraft $i$
    if i is in the air then
        if $\max\{\textit{planned arrival time of } i + mta_i, dt_1\} < dt_2$ then
            $S.append(i)$
        end
    else
        if $\max\{\textit{the time } i \textit{ has arrived} + mta_i, dt_1\} < dt_2$ then
            $S.append(i)$
        end
    end
end
```

Appendix B. Example of states changed by the actions

Fig. B.9 depicts the small example of how states are changed by swapping actions. At the first time step, Aircraft 1 is assigned to the arrival of Flight 1 (i.e., arriving aircraft will be Aircraft 1). The AOCC decided to swap Aircraft 1 and Aircraft 2, which are planned to arrive at the same airport. Assuming that Aircraft 3 does not satisfy the swappable condition, Aircraft 1 could not swap with Aircraft 3. At the second time step, the future route states of Aircraft 1 and Aircraft 2 were changed, and the binary parameter state of Aircraft 1 was changed to zero due to the previous time step. At the third time step, because the departure of Flight 2 was assigned to Aircraft 1 at the second time step, the previous route and binary parameter state of Aircraft 1 was changed. The arrival of Flight 7 was assigned to Aircraft 2, and the AOCC decided not to swap. Therefore, only the binary parameter state of Aircraft 2 was changed to zero at the fourth time step.

Appendix C. Without swapping strategy

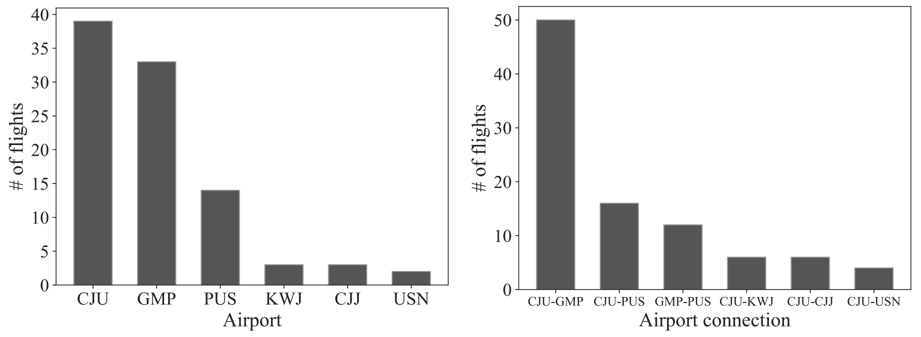
For validating the performance of RL approach, we proposed WSAP for comparison. In real practice, flight schedulers build extra buffer time between consecutive flights for absorbing unpredictable flight delays. Therefore, without implementing additional recovery options, just delaying flights' departure times could be an effective strategy. For implementing the above strategy, we utilized the environment proposed in Section 4.2. In contrast with RL approach, there was only one valid action (e.g., initial aircraft route) in every time step when carrying out WSAP. Thus, implementing only one episode is required for WSAP. The pseudocode of WSAP is as follows:

Algorithm 4 WSAP.

```
$at_c \leftarrow 0, \forall c \in C$
$t_f^d \geq mta_c, \forall f \in F, \forall c \in C$
$E \leftarrow$ 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 TAT of aircraft $c$
    if e is departure event 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
```

Appendix D. Detailed description of the domestic flight schedule in South Korea

In South Korea, Jeju Island is the main low-cost carrier (LCC) market and is considered one of the best vacation spots. Air transport is the fastest way to travel to Jeju Island; thus, over 90% of travelers use airline service [61]. Hence, domestic airline flights in South Korea are concentrated on the CJU. However, owing to this reason, CJU had the highest average delay time and incurred a very large impact on domestic propagated delays [62]. If an extreme air transport disruption, such as a temporary airport closure, occurs, a large number of flights will be affected by delay



(a) The number of flights connected to each airport (b) The number of flights belonging to each flight route

Fig. D.10. Flight information in terms of airports.

propagation. Therefore, an effective ARP strategy is necessary for dealing with the concentration of domestic airline flights on the CJU, which is why we implemented a case study for South Korea.

We analyzed the input data set, which is a domestic flight schedule from one of the airlines in South Korea. Fig. D.10 shows the flight information regarding airports. As anticipated, airline flights are concentrated on the CJU. The CJU is connected to all five airports. The GMP is also connected to many flights. The CJU-GMP flight route is the busiest in the domestic flight schedule. The PUS is connected to CJU and GMP, and is connected with fourteen flights. However, the KWJ, CJJ, and USN are only connected to three or two flights.

References

[1] D. Hummels, Transportation costs and international trade in the second era of globalization, J. Econ. Perspect. 21 (3) (2007) 131–154.
[2] K. Ng, C.K. Lee, F.T. Chan, Y. Lv, Review on meta-heuristics approaches for airside operation research, Appl. Soft Comput. 66 (2018) 104–133.
[3] E. Khanmirza, M. Nazarahari, M. Haghbeigi, A heuristic approach for optimal integrated airline schedule design and fleet assignment with demand recapture, Appl. Soft Comput. 96 (2020) 106681.
[4] J. Rapajic, Beyond Airline Disruptions, Ashgate Publishing, Ltd, 2009.
[5] N. Kohl, A. Larsen, J. Larsen, A. Ross, S. Tiourine, Airline disruption management—perspectives, experiences and outlook, J. Air Transp. Manag. 13 (3) (2007) 149–162.
[6] P. Belobaba, A. Odoni, C. Barnhart, The Global Airline Industry, John Wiley & Sons, 2015.
[7] J.D. Petersen, G. Sölveling, J.-P. Clarke, E.L. Johnson, S. Shebalov, An optimization approach to airline integrated recovery, Transp. Sci. 46 (4) (2012) 482–500.
[8] D. Zhang, C. Yu, J. Desai, H.H. Lau, A math-heuristic algorithm for the integrated air service recovery, Transp. Res. B 84 (2016) 211–236.
[9] Y.N. Yetimoğlu, M.S. Aktürk, Aircraft and passenger recovery during an aircraft's unexpected unavailability, J. Air Transp. Manag. 91 (2021) 101991.
[10] J. Evler, M. Lindner, H. Fricke, M. Schultz, Integration of turnaround and aircraft recovery to mitigate delay propagation in airline networks, Comput. Oper. Res. 138 (2022) 105602.
[11] Y. Hu, B. Xu, J.F. Bard, H. Chi, et al., Optimization of multi-fleet aircraft routing considering passenger transiting under airline disruption, Comput. Ind. Eng. 80 (2015) 132–144.
[12] B.G. Thengvall, J.F. Bard, G. Yu, Balancing user preferences for aircraft schedule recovery during irregular operations, Iie Trans. 32 (3) (2000) 181–193.
[13] S. Bratu, C. Barnhart, Flight operations recovery: New approaches considering passenger recovery, J. Sched. 9 (3) (2006) 279–298.
[14] A. Zhao, J.F. Bard, J.E. Bickel, A two-stage approach to aircraft recovery under uncertainty, SSRN Electron. J. (2022).
[15] Z. Huang, X. Luo, X. Jin, S. Karichery, An iterative cost-driven copy generation approach for aircraft recovery problem, European J. Oper. Res. 301 (1) (2022) 334–348.
[16] J. Clausen, A. Larsen, J. Larsen, N.J. Rezanova, Disruption management in the airline industry—concepts, models and methods, Comput. Oper. Res. 37 (5) (2010) 809–821.
[17] L. Hassan, B.F. Santos, J. Vink, Airline disruption management: A literature review and practical challenges, Comput. Oper. Res. 127 (2021) 105137.
[18] T.-K. Liu, C.-R. Jeng, Y.-H. Chang, Disruption management of an inequality-based multi-fleet airline schedule by a multi-objective genetic algorithm, Transp. Plann. Technol. 31 (6) (2008) 613–639.
[19] A. Gosavii, N. Bandla, T.K. Das, A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking, IIE Trans. 34 (9) (2002) 729–742.
[20] P. Balakrishna, R. Ganesan, L. Sherry, Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of tampa bay departures, Transp. Res. C 18 (6) (2010) 950–962.
[21] G. Hondet, L. Delgado, G. Gurtner, Airline disruption management with aircraft swapping and reinforcement learning, in: SESAR Innovation Days 2018, SID 2018, 2018.
[22] J. Ruan, Z. Wang, F.T. Chan, S. Patnaik, M.K. Tiwari, A reinforcement learning-based algorithm for the aircraft maintenance routing problem, Expert Syst. Appl. 169 (2021) 114399.
[23] Y. Hu, X. Miao, J. Zhang, J. Liu, E. Pan, Reinforcement learning-driven maintenance strategy: A novel solution for long-term aircraft maintenance decision optimization, Comput. Ind. Eng. 153 (2021) 107056.
[24] S.A. Shihab, P. Wei, A deep reinforcement learning approach to seat inventory control for airline revenue management, J. Revenue Pricing Manag. 21 (2) (2022) 183–199.
[25] T. Kravaris, K. Lentzos, G. Santipantakis, G.A. Vouros, G. Andrienko, N. Andrienko, I. Crook, J.M.C. Garcia, E.I. Martinez, Explaining deep reinforcement learning decisions in complex multiagent settings: towards enabling automation in air traffic flow management, Appl. Intell. (2022) 1–36.
[26] D.-T. Pham, P.N. Tran, S. Alam, V. Duong, D. Delahaye, Deep reinforcement learning based path stretch vector resolution in dense traffic with uncertainties, Transp. Res. C 135 (2022) 103463.
[27] D. Teodorović, S. Guberinić, Optimal dispatching strategy on an airline network after a schedule perturbation, European J. Oper. Res. 15 (2) (1984) 178–182.
[28] S. Yan, D.-H. Yang, A decision support framework for handling schedule perturbation, Transp. Res. B 30 (6) (1996) 405–419.
[29] M. Løve, K.R. Sørensen, J. Larsen, J. Clausen, Disruption management for an airline—rescheduling of aircraft, in: Workshops on Applications of Evolutionary Computation, 2002, pp. 315–324.
[30] J.M. Rosenberger, E.L. Johnson, G.L. Nemhauser, Rerouting aircraft for airline recovery, Transp. Sci. 37 (4) (2003) 408–421.
[31] O. Khaled, M. Minoux, V. Mousseau, S. Michel, X. Ceugniet, A multi-criteria repair/recovery framework for the tail assignment problem in airlines, J. Air Transp. Manag. 68 (2018) 137–151.
[32] Z. Liang, F. Xiao, X. Qian, L. Zhou, X. Jin, X. Lu, S. Karichery, A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility, Transp. Res. B 113 (2018) 70–90.
[33] J. Vink, B.F. Santos, W.J. Verhagen, I. Medeiros, et al., Dynamic aircraft recovery problem-an operational decision support framework, Comput. Oper. Res. 117 (2020) 104892.
[34] Y. Hu, P. Zhang, B. Fan, S. Zhang, J. Song, Integrated recovery of aircraft and passengers with passengers' willingness under various itinerary disruption situations, Comput. Ind. Eng. 161 (2021) 107664.
[35] S. Yan, C.-G. Lin, Airline scheduling for the temporary closure of airports, Transp. Sci. 31 (1) (1997) 72–82.
[36] B.G. Thengvall, G. Yu, J.F. Bard, Multiple fleet aircraft schedule recovery following hub closures, Transp. Res. A 35 (4) (2001) 289–308.
[37] D. Šemrov, R. Marsetič, M. Žura, L. Todorovski, A. Srdic, Reinforcement learning approach for train rescheduling on a single-track railway, Transp. Res. B 86 (2016) 250–267.

[38] S. Bratu, C. Barnhart, An analysis of passenger delays using flight operations and passenger booking data, Air Traffic Control Q. 13 (1) (2005) 1–27.
[39] T.-K. Liu, C.-H. Chen, J.-H. Chou, Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm, Expert Syst. Appl. 37 (3) (2010) 2307–2315.
[40] M.M. Mota, I.F. De La Mota, D.G. Serrano, Applied Simulation and Optimization: In Logistics, Industrial and Aeronautical Practice, Springer, 2015.
[41] H. Lin, Z. Wang, Flight scheduling for airport closure based on sequential decision, in: 2018 4th International Conference on Information Management, ICIM, IEEE, 2018, pp. 241–245.
[42] M.A. Jones, D.L. Mothersbaugh, S.E. Beatty, Why customers stay: measuring the underlying dimensions of services switching costs and managing their differential strategic outcomes, J. Bus. Res. 55 (6) (2002) 441–450.
[43] Z. Wu, Q. Gao, B. Li, C. Dang, F. Hu, A rapid solving method to large airline disruption problems caused by airports closure, IEEE Access 5 (2017) 26545–26555.
[44] M.R. Garey, D.S. Johnson, Computers and Intractability, vol. 174, freeman San Francisco, 1979.
[45] K.M. Hamdia, X. Zhuang, T. Rabczuk, An efficient optimization approach for designing machine learning models based on genetic algorithm, Neural Comput. Appl. 33 (6) (2021) 1923–1933.
[46] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A.M. Karimi-Mamaghan, E.-G. Talbi, Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art, European J. Oper. Res. 296 (2) (2022) 393–422.
[47] L. Meenachi, S. Ramakrishnan, Metaheuristic search based feature selection methods for classification of cancer, Pattern Recognit. 119 (2021) 108079.
[48] A. Gosavi, Reinforcement learning: A tutorial survey and recent advances, INFORMS J. Comput. 21 (2) (2009) 178–192.
[49] M. Nazari, A. Oroojlooy, L. Snyder, M. Takac, Reinforcement learning for solving the vehicle routing problem, in: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc, 2018.
[50] N. Mazyavkina, S. Sviridov, S. Ivanov, E. Burnaev, Reinforcement learning for combinatorial optimization: A survey, Comput. Oper. Res. 134 (2021) 105400.
[51] J.R. Vázquez-Canteli, Z. Nagy, Reinforcement learning for demand response: A review of algorithms and modeling techniques, Appl. Energy 235 (2019) 1072–1089.
[52] A. Haydari, Y. Yilmaz, Deep reinforcement learning for intelligent transportation systems: A survey, IEEE Trans. Intell. Transp. Syst. 55 (1) (2020) 11–32.
[53] C. Yu, J. Liu, S. Nemati, G. Yin, Reinforcement learning in healthcare: A survey, ACM Comput. Surv. 55 (1) (2021) 1–36.
[54] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[55] Y.-K. Gu, J. Zhang, Y.-J. Shen, C.-J. Fan, Fault tree analysis method based on probabilistic model checking and discrete time markov chain, J. Ind. Prod. Eng. 36 (3) (2019) 146–153.
[56] R. Bellman, Dynamic programming, Science 153 (3731) (1966) 34–37.
[57] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[58] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. (1951) 400–407.
[59] S. Thrun, 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.
[60] H.V. Hasselt, Double q-learning, in: Advances in Neural Information Processing Systems, 2010, pp. 2613–2621.
[61] J.Y. Chung, T. Whang, The impact of low cost carriers on Korean island tourism, J. Transp. Geogr. 19 (6) (2011) 1335–1340.
[62] M. Kim, S. Park, Airport and route classification by modelling flight delay propagation, J. Air Transp. Manag. 93 (2021) 102045.