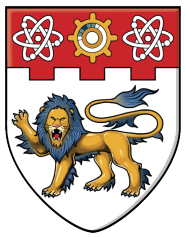


NANYANG TECHNOLOGICAL UNIVERSITY

REINFORCEMENT LEARNING FOR MULTI-AIRPORT SLOT SCHEDULING WITH FAIRNESS CONSIDERATION

NGUYEN DUY ANH

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

2025

REINFORCEMENT LEARNING FOR MULTI-AIRPORT SLOT SCHEDULING WITH FAIRNESS CONSIDERATION

NGUYEN DUY ANH

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2025

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

02/05/2025

Date

Nguyen Duy Anh

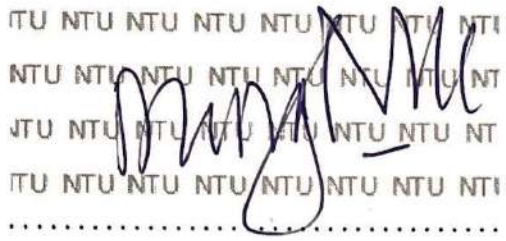
Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

02/05/2025

.....

Date



.....

Vu N. Duong

Authorship Attribution Statement

This thesis contains materials from three papers published in the following peer-reviewed conferences in which I am listed as an author.

Chapter 4 is published with material from:

(1) Anh Nguyen-Duy, Duc-Thinh Pham, Jian-Yi Lye, Duong Ta, "Reinforcement Learning for Strategic Airport Slot Scheduling: Analysis of State Observations and Reward Designs," *IEEE Conference on Artificial Intelligence (CAI)*, 2024.

The contributions of the authors are as follows:

- I proposed the research ideas, developed the methodologies, and implemented the models for the studies.
- Dr. Duc-Thinh Pham provided valuable guidance on model development, experimental design, and technical feedback.
- Jian-Yi Lye assisted in the execution of experiments for the study presented in Chapter 4.
- Professor Duong Ta contributed to discussions and gave feedback during the development phases for Chapter 4.
- I conducted the experiments, analyzed the results, and prepared the manuscripts for publication.
- Professor Vu N. Duong provided critical feedback and guidance, which served as the backbone for the paper.

Chapter 5 is published with material from:

(2) Anh Nguyen-Duy, Duc-Thinh Pham, "Reinforcement Learning for Collaborative Multi-Airport Slot Re-Allocation Under Reduced Capacity Scenarios," *2024 International Workshop on ATM/CNS (IWAC)*, 2024.

The contributions of the authors are as follows:

- I proposed the research ideas, developed the methodologies, and implemented the models for the studies.
- Dr. Duc-Thinh Pham provided valuable guidance on model development, experimental design, and technical feedback.
- I conducted the experiments, analyzed the results, and prepared the manuscripts for publication.
- Professor Vu N. Duong provided critical feedback and guidance, which served as the backbone for the paper.

Chapter 6 is published with material from:

(3) Anh Nguyen-Duy, Duc-Thinh Pham, "Reinforcement Learning for Airport Slot Allocation: Incorporating Fairness and Trade-off Analysis," *25th Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 2025.

The contributions of the authors are as follows:

- I proposed the research ideas, developed the methodologies, and implemented the models for the studies.
- Dr. Duc-Thinh Pham provided valuable guidance on model development, experimental design, and technical feedback.
- I conducted the experiments, analyzed the results, and prepared the manuscripts for publication.

- Professor Vu N. Duong provided critical feedback and guidance, which served as the backbone for the paper.

02/05/2025

Date

Nguyen Duy Anh

Acknowledgements

My deepest and sincerest thanks to:

- Professor Vu N. Duong, for giving me the opportunity to start my Ph.D. journey at the Air Traffic Management Research Institute (ATMRI), and for his patient guidance, unwavering support, and for consistently pushing me beyond my limits. Thank you for being a true role model in both research integrity and life ethics, to whom I am deeply grateful and full of admiration.

- Professor Sameer Alam, for agreeing to be my co-supervisor and for his invaluable help and motivation throughout this journey. He has been a shining example of ambition and hard work, serving as an inspiration and a figure of respect for a young researcher like myself.

- Dr. Thinh Pham, for mentoring me through thick and thin, and for his selfless care and thoughtful guidance. He has been not only a dear teacher but also a big brother throughout this journey.

- Professor Don Ta of the Singapore Management University (SMU), for his constant encouragement and wholehearted support.

- Soon-to-be-Dr. Nam, for his advice and emotional support. He has accompanied me through both hard and happy times, becoming another big brother for me at ATMRI.

- Jian-Yi, for helping me kick-start my first research paper.

- My labmates at ATMRI, Dr. Phu, big brother Hoang, big brother Phuoc, Dr. Yash, Dr. Ma, Dr. Ali, Dr. Arif, Dr. Cai, Dr. Kiran, Greg, Zhi Jun, Duy, Minh, Hoang, and many others, for all the fun gatherings, collaboration, and help along the way.
- My friends, Sonu, the two Vijays, Bernie, and many others in Singapore and Vietnam, for their companionship during both the hard and happy times.
- Ms. Suan, Ms. Yati, and Mr. Sunny, for their help and support.
- All my teachers before my Ph.D. journey, who helped me build a strong foundation and supported me in pursuing the Ph.D. opportunity.
- My family, for their unconditional love and support, which has been my anchor throughout this journey.
- Vingroup Scholarship Program for sponsoring my PhD study.
- And to all others who have supported me, whether in big or small ways. Thank you.

Contents

Abstract

The rapid growth of air traffic demand has intensified airport congestion, leading to significant economic and environmental impacts across the global air transport system. Airport slot scheduling, governed by the International Air Transport Association's Worldwide Airport Slot Guidelines (WASG), has emerged as a key mechanism for balancing supply and demand. However, current practices remain largely ad hoc, lacking optimization support tools and fairness considerations, thereby motivating the need for more effective and practical methodologies.

This thesis addresses the airport slot scheduling problem, beginning with a single-airport formulation as a foundational framework and extending to multi-airport perspectives. Recognizing the limitations of exact optimization and heuristic methods, particularly their scalability challenges and heavy reliance on expert knowledge, this research proposes the novel application of Reinforcement Learning (RL) for slot scheduling. RL offers a promising alternative by enabling adaptive, data-driven decision-making in complex, dynamic environments.

First, the thesis develops an RL-based framework for single-airport slot scheduling, analyzing different state observations, reward designs, and learning mechanisms to enhance training efficiency and solution quality. Building on this foundation, the study extends the framework to multi-airport systems, introducing a decentralized paradigm in which each airport operates autonomously without reliance on a central decision-making authority, especially relevant in regions such as ASEAN, where no centralized aviation governance exists.

A key contribution of this research is the integration of fairness considerations into the RL formulation. Recognizing fairness as critical for stakeholder acceptance and operational stability, three fairness incorporation strategies are investigated: the mini-max criterion, the

principle of proportionality, and constraint-based approaches. The thesis formulates reward designs that balance the trade-off between system efficiency and fairness, highlighting the risks associated with sparse and overly dense reward structures, and proposing strategies to optimize training convergence.

Experiments, including both training and testing on realistic airport scenarios derived from OAG flight data, demonstrate that RL agents can effectively learn slot scheduling policies that achieve high system efficiency while incorporating fairness. Results reveal that per-time-step reward signals are crucial for stable and efficient learning; that fairness-oriented reward components successfully promote equitable solutions; and that different fairness strategies offer distinct trade-offs between efficiency and fairness. In particular, while fairness improvements can lead to an increase in total delay, well-designed RL reward structures can mitigate this problem.

In conclusion, this thesis demonstrates the feasibility and advantages of applying RL to the multi-airport slot scheduling problem with fairness considerations. It offers a practical framework that addresses critical gaps in current practice. This research paves the way for future developments in autonomous air traffic management systems, contributing toward more efficient and equitable airport operations. Future work may explore multi-agent reinforcement learning, integrate stochastic operational uncertainties, and incorporate different operational constraints.

List of Figures

List of Tables

Chapter 1

Introduction

Air transport plays a vital role in the transportation sector. Despite the challenges posed by the COVID-19 pandemic, predictions indicate a significant recovery and increase in air travel demand over the coming year. However, the existing infrastructure is not sufficient to meet this rising demand. Restrictions on expanding capacity, such as constructing new airports, will lead to congestion, causing schedule disruptions and delays. These delays will impact individual airports and ripple throughout the airport network, resulting in widespread economic and environmental consequences for the entire air transport system.

To address this challenge, the International Air Transport Association (IATA) introduced the Worldwide Airport Slot Guidelines (WASG) to regulate and balance air transportation supply and demand. The WASG provides a standardized framework for flight scheduling across different airports and countries, aiming to proactively reduce delays through better coordination. However, the current scheduling practices are largely ad hoc and lack the capability to deliver optimal solutions. Limitations in existing slot scheduling methods in the literature further emphasize the need for exploring alternative approaches, which forms the core motivation of this thesis.

This chapter reviews the state of air transportation supply and demand, highlighting airport congestion and the role of slot scheduling as a key solution. The chapter identifies inefficiencies in current practices, presents the research motivation and objectives, and concludes with the thesis structure.

1.1 Airport Congestion Problem

1.1.1 The Growth in Air Traffic Demand

The aviation industry has undergone significant growth over the years. Pre-COVID-19 forecasts by the International Civil Aviation Organization (ICAO) projected that global passenger traffic would double by 2035, based on an anticipated annual growth rate of 4.3% (ICAO, 2007), as illustrated in Figure 1.1. In 2018 alone, the industry served 8 billion passengers and facilitated over 90 million flights worldwide.

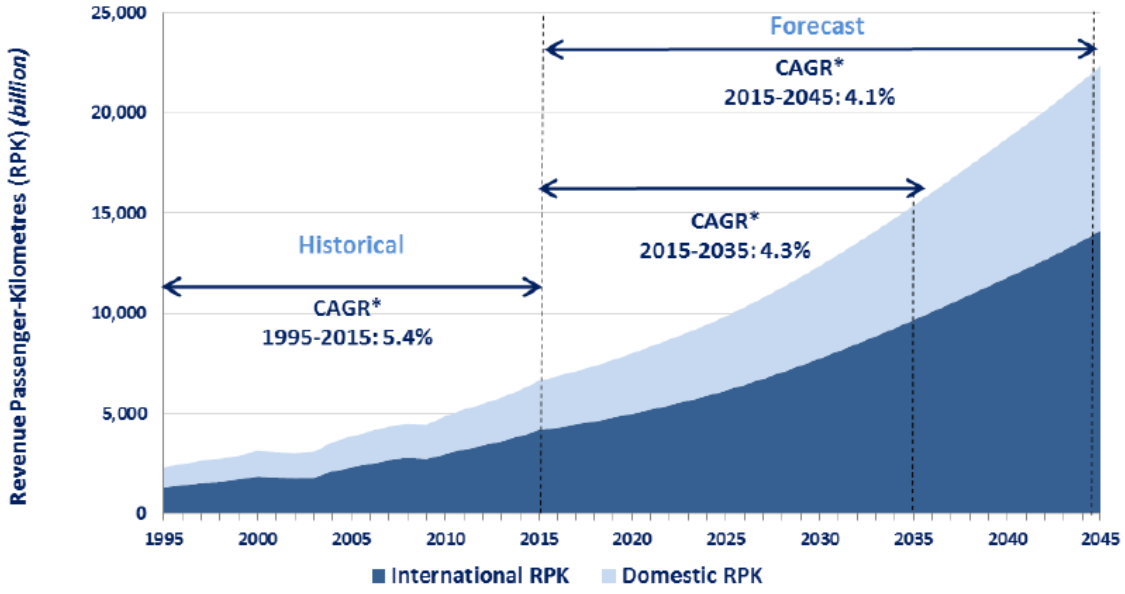


Figure 1.1: ICAO passenger traffic forecast in terms of Revenue Passenger-Kilometres (RPK). Figure obtained from "Global Air Transport Outlook to 2032 and Trends to 2042" Aviation (2025).

During the COVID-19 pandemic, the air travel sector encountered substantial challenges as governments implemented widespread travel restrictions. These measures led to severe disruptions in the aviation industry. By the end of the second quarter of 2020, the global airline industry experienced an operating loss of more than 70% of its revenue, as depicted in Figure 1.2.

However, following the ease of the pandemic, the airline industry is showing robust signs of recovery. According to IATA's updated forecast, Revenue Passenger Kilometers (RPKs)

Figure 1.2: Global airline industry annual revenue between 2015 and 2020. Figure obtained from "Outlook for the global airline industry april 2021 update" IATA (2021).

are projected to return to pre-COVID-19 levels by 2024 and maintain a consistent growth trend (IATA, 2021), as demonstrated in Figure 1.3. This indicates that, despite the severe disruptions caused by the pandemic, air traffic demand is expected to resume its upward trajectory, aligning with pre-COVID-19 projections.

1.1.2 Insufficient Infrastructure

The existing infrastructure, however, is unlikely to meet the increasing demand. Table 1.1 presents the three levels of coordinated airports defined by the International Air Transport Association (IATA). Each level reflects the airport's capability to manage air traffic demand:

- *Level 1:* Airports with sufficient capacity to meet demand. These airports do not require slot coordination as their capability to handle traffic is generally adequate.

- *Level 2:* Airports where congestion may occur during certain times of the day, week, or season. At these airports, schedule adjustments are made in coordination with airlines to manage demand effectively.

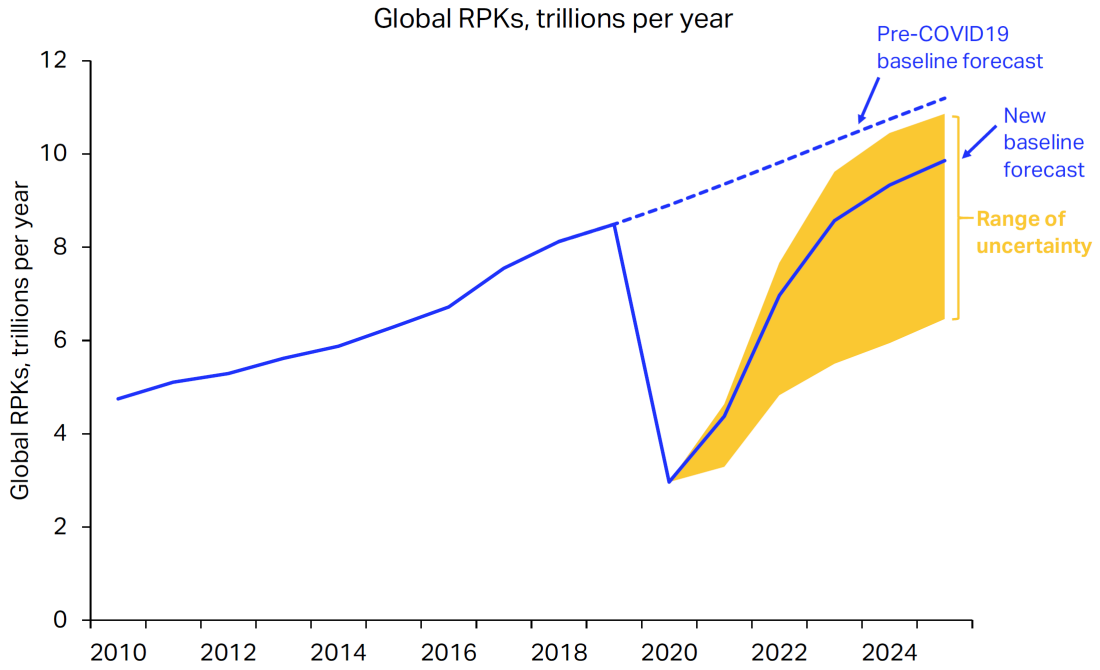


Figure 1.3: IATA's January 2021 Air Passenger Forecast. Figure obtained from "Outlook for the global airline industry april 2021 update" IATA (2021).

- *Level 3:* Airports where demand significantly exceeds available capacity. The slot allocation at these airports requires an independent slot coordinator. These airports require strict slot management to ensure operational efficiency and minimize congestion.

In the "Contact List for Level 2 and Level 3 Airports" published by IATA, airports across Europe, the Americas, and other global regions are classified under the IATA coordination levels, demonstrating that the IATA airport coordination framework is applied worldwide (IATA, 2023).

During the 2022 winter scheduling season, there were 153 Level 2 airports and 174 Level 3 airports globally. These numbers increased to 157 and 199, respectively, for the summer scheduling season of 2023. As of the time of writing this thesis, there are approximately 160+ Level 2 airports and 200+ Level 3 airports globally (IATA, 2022a). According to IATA, approximately 50% of all air passengers depart from Level 3 airports, and about 35% of all flights occur between two Level 3 airports. As air traffic recovered after the pandemic, flight punctuality declined again across both the U.S. and Europe. In the United States, arrival on-time performance consistently worsened from 2020 to mid-2023, falling below pre-

Level 1	Level 2	Level 3
• The airport capability is generally adequate to meet demand. • No airport coordination is required at a Level 1 airport.	• An airport where there is potential for congestion during some periods of the day, week, or season. • Schedule adjustments are mutually agreed with the airlines.	• An airport where demand significantly exceeds the airport capability. • It is necessary for all airlines and other aircraft operators to have a slot allocated by an independent slot coordinator to arrive or depart at the airport.

Table 1.1: Description of Airport Levels as per the International Air Transport Association (IATA)

pandemic (2019) levels. In Europe, punctuality declined moderately in 2021 but reached a historic low in the summer of 2022 (EUROCONTROL, 2024). These figures emphasize that the current infrastructure cannot keep up with the rising air traffic demand.

1.1.3 Impact of Airport Congestion

The mismatch between the handling capability of the infrastructure and the air traffic demand results in the airport congestion problem. In the first quarter of 2023, the average delay per flight reached a five-year high of 14.5 minutes, up from 9.5 minutes in the same period in 2022. This increase occurred alongside a 21% rise in the number of flights. Arrival punctuality also declined, with only 76.4% of flights arriving within 15 minutes of their scheduled arrival time (EUROCONTROL, 2023). Schedule disruptions, especially at crowded airports, create a ripple effect throughout the airport network.

Besides leading to operational bottlenecks and causing passenger dissatisfaction, flight delays cause detrimental economic and environmental impacts. For instance, inefficiencies in Air Traffic Management (ATM) within the European Union caused 10.8 million minutes of flight delays in 2012, leading to costs amounting to €4.5 billion for airspace users and €6.7 billion for passengers (Eurocontrol, 2014). In 2022, schedule disruptions in European air traffic incurred economic losses between $ 27 billion and $ 32 billion (AirHelp, 2022).

The delays contributed to 11.6 million tonnes of unnecessary CO2 emissions in 2022, a notable increase from 7.8 million tonnes in 2012 (IATA, 2014; EASA, 2022). The economic loss stems from increased operational costs for airlines and time lost by passengers. Holding patterns and diversions, induced by congestion, further elevate fuel consumption and emissions. Excessive fuel burn harms the aviation sector's environmental sustainability. In 2006, approximately 22% of U.S. flights experienced delays, marking the highest rate since 2000 (24%). The average aircraft load factor also reached 80%, the highest in U.S. aviation history at the time. Delay rates had been steadily increasing—rising from 16% in 2003 to 20% in 2004, 21% in 2005, and 22% in 2006, indicating a worsening congestion trend across the U.S. air transportation system (Donohue and III, 2008). In the Asia-Pacific (APAC) region, slot-constrained airports served roughly 2.6 billion passengers in 2019—approximately 28% of global passenger traffic, indicating intense demand pressures in APAC markets as well (APAC), 2023). In Southeast Asia specifically, slot allocation practices are increasingly relevant, with airports like Singapore Changi, Kuala Lumpur, and Bangkok experiencing frequent peak congestion.

1.2 Airport Slot Scheduling: A Solution for Airport Congestion Problem

With forecasts predicting a robust post-COVID-19 surge in air traffic demand, it is evident that the current infrastructure will be insufficient to accommodate long-term growth. This imbalance between supply and demand is expected to exacerbate airport congestion. Expanding infrastructure is costly and unfeasible in the short term, with constraints such as limited land availability further restricting the development of additional runways and airports. Consequently, there is a critical need to prioritize improving demand management strategies over relying on infrastructure expansion alone.

Most airports worldwide comply with IATA Worldwide Airport Slot Guidelines (WASG) to regulate airline demand, which is defined by movement requests, subject to airport ca-

pacity represented by slots, at Level-3 airports (Zografos et al., 2017). Alternative demand management methods include slot trading, auctions, and congestion/peak pricing; however, these options introduce costs and complexity, making them inefficient for broad applications and serving best as niche solutions in specific situations (IATA, 2022b).

Although the United States does not uniformly apply the IATA Worldwide Airport Slot Guidelines (WASG), international slot coordination between U.S. and non-U.S. airports is harmonized through IATA Slot Conferences and bilateral schedule facilitation processes. U.S. and foreign carriers coordinate slot requests for international routes via the IATA Slot Conference, ensuring schedule compatibility at IATA-coordinated airports abroad (IATA, 2024). On the U.S. side, several airports, including JFK, LGA, and DCA, are designated as slot-controlled and managed under the FAA's Slot Administration System, which allocates operating authorizations in accordance with FAA rules. The FAA explicitly states that it "generally follows the International Air Transport Association (IATA) Worldwide Slot Guidelines (WSG) to the extent they do not conflict with U.S. laws, rules, or procedures" (FAA, 2025). Thus, while the regulatory frameworks differ, international slot coordination is operationally aligned through reciprocal slot management and shared participation in IATA's global slot coordination process, ensuring cross-border schedule compatibility between FAA- and IATA-regulated systems.

Managing demand adhering to IATA principles involves improving airport capacity parameters utilizing declared capacity modeling or regulating the distribution of demand through airport slot scheduling. While declared capacity modeling provides the necessary constraint inputs, it still depends on established slot scheduling techniques for validation. Thus, slot scheduling serves as an effective approach to mitigate airport congestion.

Airport slot scheduling represents a significant stream of research due to its potential to deliver swift improvements in capacity utilization. Slot scheduling models can streamline and enhance the current IATA scheduling procedures, offering solutions that are straightforward to implement and do not require regulatory changes (Zografos et al., 2017). Furthermore, slot scheduling approaches demonstrate versatility, as they can be applied across different

stages of air traffic management, spanning from the strategic phase to the pre-tactical and tactical phases.

It is important to note that the term “slot” in this thesis refers to airport slot scheduling as defined by the International Air Transport Association (IATA) Worldwide Airport Slot Guidelines (WASG). Tactical Air Traffic Flow Management (ATFM) measures, such as Ground Delay Programs (GDPs) and Airspace Flow Programs (AFPs), are therefore outside the scope of this study.

1.2.1 Current Practice of Airport Slot Scheduling

To address the challenges of demand management at congested airports, the International Air Transport Association (IATA) has established regulations and guidelines for airport slot scheduling. This process involves allocating airlines' departure and arrival requests to limited airport resources, known as airport slots. A slot is defined as "the permission given to a carrier to use the full range of airport infrastructure necessary to operate at a slot-controlled airport on a specific date and time for landing and takeoff" (IATA, 2015).

More specifically, a slot is represented as a time interval, typically 5 minutes, on a specific date. At the strategic phase, IATA mandates that requests are allocated to a series of slots, which consists of at least five slots recurring on the same time interval and day of the week over a designated period of operation. Slot allocation follows a hierarchy: requests with historical (grandfathered) rights are given the highest priority, followed by new entrant requests, with all other requests receiving the lowest priority (IATA, 2024). A key requirement of this process is to ensure that allocations align as closely as possible with the requested time slots. We will later describe how this objective is conceptualized in the current literature on airport slot scheduling.

Figure 1.4 shows the IATA slot scheduling process timeline and the involved stakeholders at Level-3 airports (Ribeiro et al., 2019b). The IATA slot scheduling process operates on a biannual cycle, with separate scheduling for summer and winter seasons. For each season, the process consists of five main steps, which are outlined below:

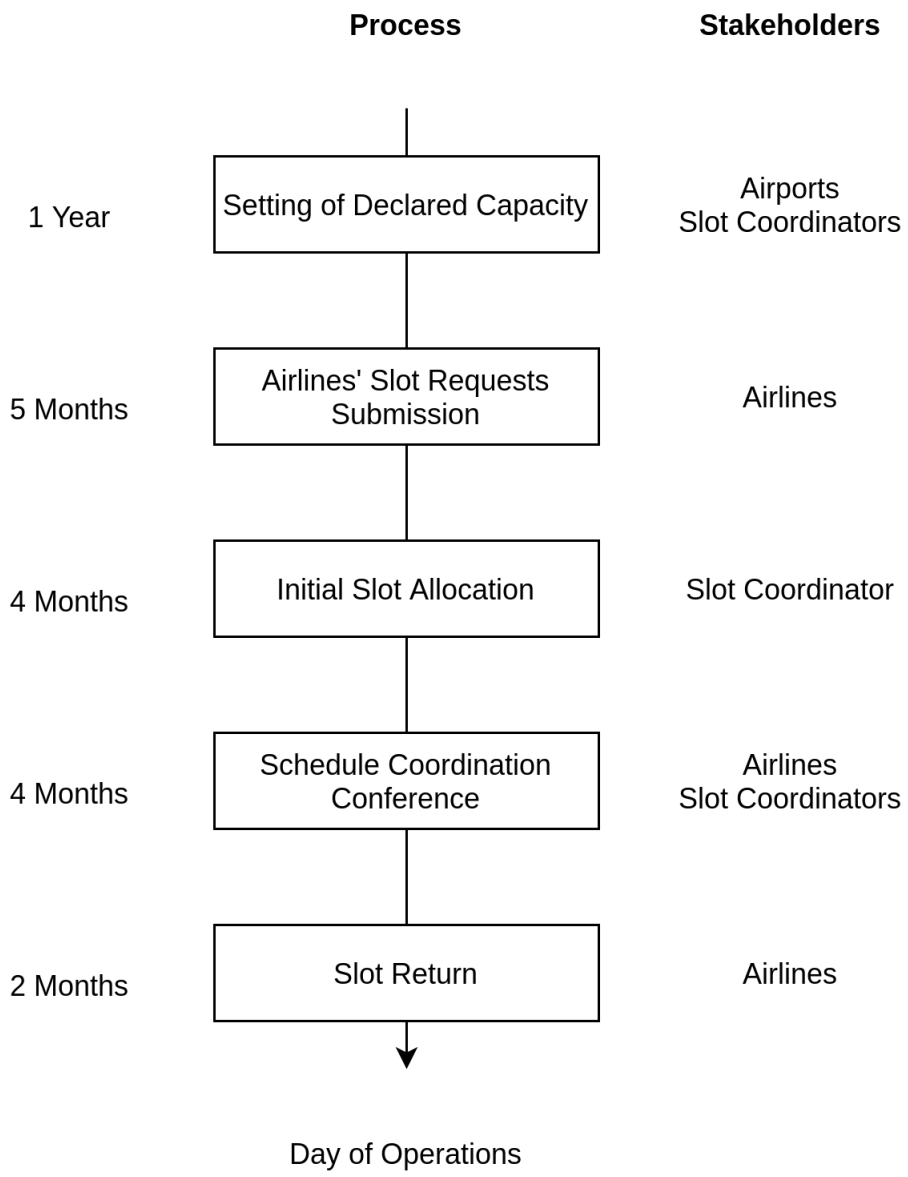


Figure 1.4: IATA Slot Scheduling Process: Timeline and Involved Stakeholders.

- *Step 1, Setting of Declared Capacity:* At coordinated airports, authorities assess slot allocation capacity, which specifies the number of available slots per time unit, twice a year in collaboration with key stakeholders, including Air Traffic Control (ATC), airlines, and airport coordinators. Each airport provides its "declared capacity," a figure outlining the number of slots or movements the airport can effectively handle during each time interval throughout the day.

- *Step 2, Airlines' Slot Requests Submission:* Airlines submit slot requests to the airport

coordinator, detailing their preferred time intervals for landing and take-off operations at the airport.

- *Step 3, Initial Slot Allocation:* Once receiving requests from airlines, the slot coordinator at each airport performs the initial slot allocation in an "unbiased, transparent, and non-discriminatory" manner, as mandated by IATA guidelines. This process ensures no direct interaction between the coordinator and the airlines. The allocation adheres to established slot scheduling priorities and requirements, and at this stage, only series of slots are assigned.

- *Step 4, Schedule Coordination Conference:* Representatives from airlines, slot coordinators, airport authorities, and other key stakeholders convene at the semi-annual IATA Slot Conference. This meeting serves as a platform to resolve conflicts arising from allocated slot timings, address disputes among airlines, and make necessary adjustments to ensure alignment with operational and scheduling requirements across various airports.

- *Step 5, Slot Return:* Airlines may return slots they do not intend to utilize and request schedule modifications to better align with their operational objectives. These changes are subject to review and approval by the slot coordinator to ensure compliance with capacity constraints and scheduling guidelines.

Slot coordinators must consider both the primary criteria established by the IATA WSG and the secondary criteria, as well as local guidelines and regulations, during Steps 3 and 4 of the process (IATA, 2024).

1.2.2 Inefficiencies of The Current Practice

Despite having standardized procedures, the current practice has two major inefficiencies that drive the motivation for this thesis:

- *Lack of Optimization Support Tools:* The initial slot allocation conducted by slot coordinators is largely empirical and lacks the support of robust optimization tools (Ribeiro

et al., 2018). In practice, slot requests are processed sequentially and often on an ad hoc basis, leading to inefficiencies. While tools like PDC SCORE exist to assist in current slot allocation procedures, they do not provide optimization capabilities, potentially resulting in suboptimal solutions (Ribeiro et al., 2018; Benlic, 2018). This gap highlights the need for advanced optimization tools to support and enhance the slot scheduling process.

- *Neglect of Fairness Consideration:* Fairness, a critical aspect of resource allocation, has been insufficiently addressed in resolving airport congestion (Bertsimas et al., 2011a). Although the IATA WASG emphasizes that slot allocation should be conducted in an "unbiased, transparent, and non-discriminatory" manner, they do not provide a strict metric or framework for evaluating fairness in the process. In regions lacking a centralized authority, ensuring fairness is essential for fostering consensus among diverse stakeholders. Without equitable slot allocation practices, disagreements among airlines, airports, and regulatory bodies may arise, undermining the effectiveness of slot scheduling solutions. Addressing fairness is essential to develop solutions that are widely accepted and promote cooperation among stakeholders.

1.2.3 Reinforcement Learning as a Suitable Optimization Approach

While existing optimization and heuristic methods have made progress in airport slot scheduling, they continue to struggle with the scale, uncertainty, and dynamic interactions that characterize real-world operations (Zografos et al., 2017; Zhang et al., 2020b,a; Wang et al., 2019b). Reinforcement Learning (RL) offers a promising alternative, as it enables an agent to learn and continuously update allocation strategies through experience and feedback, rather than relying on predefined rules or static models (Zhang et al., 2020b; Mazyavkina et al., 2021). RL has been successfully applied to related problems, such as resource-constrained scheduling and air traffic management, demonstrating strong potential for application to the slot scheduling problem (Zhang et al., 2020a; Wang et al., 2019b; Khadilkar, 2019). Moreover, RL can operate in a decentralized manner, making it suitable for regions without a central-

ized scheduling authority, where each airport must maintain operational autonomy (Duong et al., 2019). Finally, RL supports multi-objective optimization, which is essential for incorporating fairness considerations (Wang et al., 2019b; Cui et al., 2015). These features make RL suitable for developing data-driven, scalable, and practical slot allocation frameworks.

Most existing approaches in the literature adopt a centralized optimization framework, where all airports are solved simultaneously to obtain a single global solution. While this offers theoretical optimality, such centralized models are not operationally feasible in regions lacking a central coordinating authority. Moreover, centralized optimization requires recomputation of the entire network whenever schedule updates are needed, often leading to drastic changes that undermine previous planning efforts.

1.3 Thesis Contributions

To overcome the above limitations, this thesis develops a Reinforcement Learning (RL)-based framework for airport slot scheduling that enhances operational efficiency, supports decentralized coordination among airports, and incorporates fairness considerations into the scheduling process. The study progresses from a single-airport formulation to a multi-airport coordination framework, and finally to a fairness-integrated model, addressing key practical and methodological gaps in existing slot scheduling research.

The key contributions of this thesis are as follows:

- Formulate the airport slot scheduling task as a Reinforcement Learning problem and design a learning environment for agent training and evaluation.
- Develop and evaluate a multi-airport RL framework that enables decentralized coordination among airports without reliance on a central authority.
- Integrate fairness considerations into the RL formulation and analyze the trade-off between system efficiency and fairness.
- Identify limitations and propose a roadmap for scaling the proposed frameworks toward

real-world implementation.

The subsequent chapters of this thesis are organized as follows:

- *Chapter 2: Literature Review*

  This chapter examines the literature on airport slot scheduling, focusing on both single-airport and network-level (multi-airport) contexts, as well as fairness metrics in slot allocation. The chapter identifies key research gaps and outlines the research questions that this study aims to address.

- *Chapter 3: Methodology*

  This chapter provides an overview of the proposed methodology for addressing the identified research questions. It justifies the suitability of Reinforcement Learning as a decision-making mechanism and outlines the concept for applying Reinforcement Learning to the airport slot scheduling problem.

- *Chapter 4: Reinforcement Learning for Single-Airport Slot Scheduling*

  This chapter describes the methodology for formulating the airport slot scheduling problem using Reinforcement Learning (RL), providing a foundational framework for extending RL-based slot scheduling to address diverse objectives and constraints.

- *Chapter 5: Reinforcement Learning in a Multi-Airport Slot Scheduling System*

  This chapter examines the application of Reinforcement Learning (RL) to slot scheduling within a multi-airport system. The chapter focuses on formulating and evaluating the performance of an RL agent operating as a node within an airport hub. The study provides insights into leveraging RL to address the multi-airport slot scheduling problem.

- *Chapter 6: Incorporating Fairness into Reinforcement Learning for Airport Slot Scheduling*

This chapter explores different approaches to incorporating fairness using Reinforcement Learning (RL) for airport slot scheduling. The chapter examines methods for integrating fairness into RL formulations and analyzes the trade-offs between minimizing delays and ensuring fairness.

- *Chapter 7: Conclusions and Future Work*

  This chapter concludes the thesis by summarizing the key research outcomes, outlining the limitations identified throughout the study, and presenting a roadmap for translating the proposed Reinforcement Learning (RL)-based frameworks into real-world applications.

Chapter 2

Literature Review

The current airport slot scheduling practices exhibit inefficiencies that must be addressed. To the extent of our literature search and discussion with operational experts, we could not find any practical applications inspired by existing works in the literature.

In this chapter, we delve into the literature on airport slot scheduling to explore the reasons behind the lack of practical implementation. Based on the literature review, we highlight the limitations of existing methodologies and identify the key research gaps along with the research questions.

Given the critical role of fairness in resource allocation problems, particularly in the context of airport slot scheduling, it is essential to examine how fairness has been addressed in existing research. We dedicate a section to reviewing how fairness has been incorporated into the slot scheduling problem. This exploration highlights the importance of fairness in slot scheduling and provides insights into how fairness can be incorporated into future proposed methodologies.

2.1 Overview

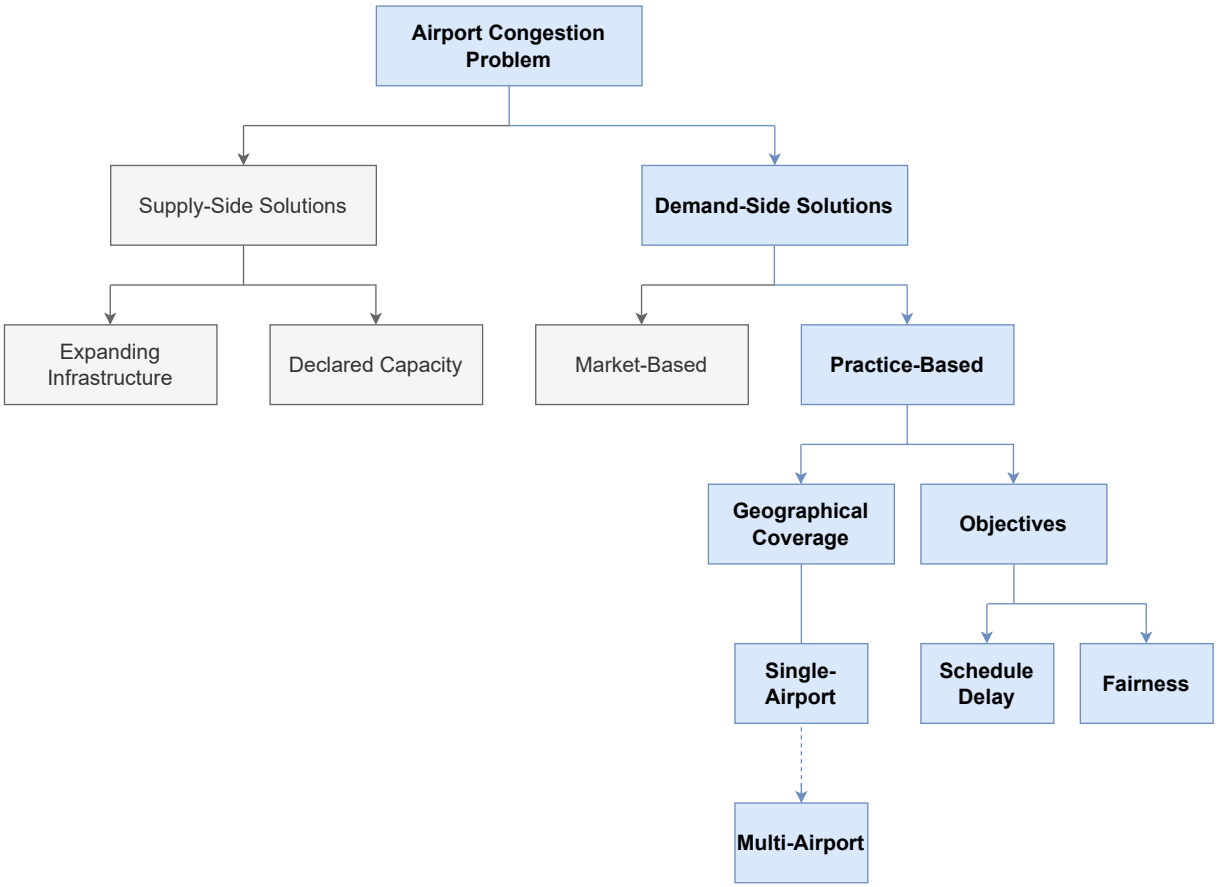


Figure 2.1: Scope of the Literature Review.

Figure 2.1 shows the focus of this literature review. The IATA slot scheduling process involves setting declared capacity and allocating slots based on predefined capacity constraints. Resolving airport congestion inherently relates to improving the accuracy of declared capacity specifications and optimizing airport slot allocation. As the declared capacity is determined six months prior to the day of operation, the actual capacity may vary on the operational day. Enhancing the accuracy of declared capacity estimation can mitigate the need for schedule adjustments, thereby improving operational stability and efficiency. In the context of airport slot allocation optimization, consider Airport A, where the traffic demand for the 0800-0805 time slot is seven flights, while the available capacity for this five-minute interval is limited to five flights. Optimizing slot allocation involves managing flight demand by reallocating flights to adjacent time slots to ensure compliance with capacity constraints. The key objective of

slot allocation is minimizing time displacement to reduce the deviation between requested and assigned time slots, thereby enhancing airline acceptability. Additionally, maintaining fairness is crucial, ensuring equitable distribution of decisions across a population of candidates, such as flights, airlines, or airports.

In addition to expanding infrastructure, a supply-side approach aims to refine declared capacity specifications through advanced analysis and modeling (Zografos et al., 2017). Declared airport capacity functions as a "control valve", imposing constraints on the slot allocation process. However, current practices for determining declared capacity at slot-coordinated airports have drawn substantial criticism (Ball et al., 2007; Odoni and Morisset, 2010). Major concerns include (i) excessive scope for localized interpretations and customizations, (ii) conservatively set capacity levels as a percentage of Instrument Flight Rules (IFR) capacity, (iii) reliance on empirical or ad hoc approaches, and (iv) limited involvement of key stakeholders in the process. Addressing these issues requires the development of a harmonized and universally agreed-upon framework for defining declared capacity and its underlying assumptions. Thus, refining the declared capacity specifications will not have an immediate impact as it involves changing the current practice of setting declared capacity in airport slot scheduling.

A demand-side approach, optimizing airport slot allocation, aims to assign requested movements, landings and takeoffs, to coordination time intervals within a designated scheduling period, typically finalized six months before operations and adjusted up to the day of operation. This process must consider a range of constraints, including declared airport capacity, priorities from the established slot allocation framework (e.g., historical slot rights), and operational requirements of airlines and airports, such as turnaround times, ground handling, and flight connectivity. Additional constraints may stem from Air Traffic Flow Management (ATFM), such as airspace sector capacity and the specific criteria for evaluating slot requests. In the literature, such optimization methods are often categorized under (airport) slot scheduling models or (airport) demand and capacity balancing models.

In this thesis, we focus on optimizing airport slot scheduling to address the congestion problem for the following reasons:

- *Direct Impact on Operational Efficiency:* Slot scheduling directly addresses the core problem of balancing demand and capacity. By optimizing slot allocation, airports can effectively utilize the declared capacity, subject to any capacity determinations. Slot scheduling serves as the key mechanism for managing demand distribution to align with capacity, enabling the industry to address capacity bottlenecks more effectively than simply refining capacity estimates alone.

- *Alignment with Existing Frameworks:* The International Air Transport Association's (IATA) Worldwide Slot Guidelines (WSG) provide a globally accepted framework for slot allocation. Slot scheduling models can complement or enhance existing IATA scheduling procedures, offering significant potential for achieving rapid improvements in capacity utilization. By integrating with the current framework, slot scheduling models provide a straightforward, immediately implementable solution that avoids the need for extensive organizational or regulatory changes. In contrast, improving declared capacity modeling often requires significant foundational changes and involves extensive stakeholder agreement, which can delay implementation and disrupt established systems.

In the broader literature on airport slot scheduling, there are two primary categories: market-based approaches and practice-based approaches. Drawing on economic theories, research on market-based approaches proposes mechanisms such as auctions, trading, or congestion-based pricing to allocate airport capacity among competing users (MacDonald, 2006; Madas and Zografos, 2006; Pellegrini et al., 2012b; Ranieri et al., 2013). Practice-based approaches, on the other hand, aim to refine or enhance the slot allocation practice, aligning closely with practical operational frameworks (Zografos et al., 2012; Ribeiro et al., 2019b; Zografos and Jiang, 2019; Benlic, 2018). They focus on improving efficiency and equity within the constraints of the established IATA guidelines. Market-based approaches are outside the scope of this study due to their reliance on economic theory. This thesis focuses on practice-based approaches, specifically optimizing airport slot allocation in accordance with IATA guidelines. These approaches have the potential for immediate improvements

in slot allocation efficiency and contribute directly to the context of the IATA framework. Furthermore, with advancements in technology, particularly in Artificial Intelligence (AI) and Machine Learning (ML), the potential for optimization has significantly increased. These technologies pave the way for new approaches to slot allocation, enhancing efficiency and decision-making in complex scheduling environments.

Table 2.1: Methodologies for Airport Slot Scheduling

Slot Scheduling	Relevant Studies	Method
Single-Airport	(Zografos et al., 2012)	Heuristics
	(Zografos et al., 2017)	Exact Methods, Heuristics
	(Androutsopoulos et al., 2020)	Heuristics
	(Ribeiro et al., 2018)	Exact Methods, Heuristics
	(Wang et al., 2019a)	Heuristics
	(Jacquillat and Odoni, 2015)	Exact Methods
	(Zografos et al., 2018)	Exact Methods
	(Shone et al., 2021)	Heuristics
	(Ribeiro et al., 2019b)	Heuristics
	(Fernández et al., 2017)	Exact Methods
	(Bertsimas et al., 2011b)	Exact Methods
	(Zografos and Jiang, 2019)	Exact Methods
	(Jacquillat and Vaze, 2018)	Exact Methods
	(Fairbrother et al., 2020)	Exact Methods, Heuristics
Network-Level	(Castelli et al., 2012)	Exact Methods
	(Bertsimas et al., 2011a)	Exact Methods
	(Pellegrini et al., 2012a)	Exact Methods, Heuristics
	(Corolli et al., 2014)	Exact Methods
	(Pellegrini et al., 2017)	Exact Methods
	(Benlic, 2018)	Heuristics
	(Keskina and Zografos, 2022)	Exact Methods
	(Keskina and Zografos, 2022)	Exact Methods

Next, we provide a comprehensive literature review on airport slot scheduling, focusing on both single-airport and network-level (multi-airport) perspectives. We examine fairness approaches in airport slot scheduling, including the mini-max criterion, the principle of proportionality, and constraint-based methods. Tables 2.1 and 2.2 summarize the methodologies discussed in the literature review, along with the corresponding relevant studies.

Table 2.2: Fairness in Airport Slot Scheduling

Fairness	Relevant Studies	Method
Mini-Max Criterion	(Jacquillat and Odoni, 2015)	Exact Methods
	(Zografos et al., 2018)	Exact Methods
	(Ribeiro et al., 2018)	Exact Methods, Heuristics
	(Ribeiro et al., 2019b)	Heuristics
	(Jacquillat and Vaze, 2018)	Exact Methods
	(Jiang and Zografos, 2021)	Exact Methods
	(Pellegrini et al., 2017)	Exact Methods
Principle of Proportionality	(Zografos and Jiang, 2016)	Exact Methods
	(Zografos and Jiang, 2019)	Exact Methods
	(Manley and Sherry, 2010)	Exact Methods
	(Fairbrother et al., 2020)	Exact Methods, Heuristics
Constrained-Based	(Jacquillat and Odoni, 2015)	Exact Methods

2.2 Airport Slot Scheduling

2.2.1 Single-Airport Slot Scheduling

The Single-Airport Slot Scheduling Problem focuses on Step 3 of the IATA slot scheduling process, optimizing slot allocation at individual airports without accounting for network-level interactions. While network-level models address the interdependency among airports, single-airport models can be integrated into the existing IATA framework. In practice, slot coordinators often perform initial allocations in an ad hoc manner, emphasizing the need for effective optimization tools to support their decision-making at the single-airport level.

The airport slot scheduling problem falls within the category of time-dependent resource-constrained scheduling problems, which are classified as NP-hard, making it impractical to develop solution algorithms with polynomial complexity (Garey, 1979). Due to this computational challenge, especially when addressing real-world instances, (Zografos et al., 2017) calls for the use of search-based approaches to tackle the problem.

(Zografos et al., 2012) introduces a heuristic algorithm to address the slot scheduling problem, adhering to the IATA's priority requirements. Slot requests are processed in a hierarchical manner, with capacity being updated after each priority group is allocated. The

algorithm was evaluated using data from three Greek international airports, with the largest test case involving 1,087 slot requests over a period of 197 days, equivalent to approximately 30,000 movements.

(Androutsopoulos et al., 2020) formulates the slot scheduling problem as a bi-objective Resource-Constrained Project Scheduling Problem (RCPSP) with partially renewable resources and non-regular objective functions. The first objective, referred to as "aggregate displacement" or "aggregate delay," was introduced by (Ribeiro et al., 2018). It represents the product of the displacement assigned to a request and the number of operational days for that request, termed here as the Total Frequency-Weighted Schedule Delay. The second objective is the "squared aggregate displacement metric," designed to mitigate extreme slot displacements by serving as an acceptability measure. The approach employs a heuristic method, the Feasibility Pump-Guided Hybrid Algorithm (FPHA), which provides a fairly accurate approximation of the problem's Pareto optimal solutions within a computationally feasible time frame. However, (Androutsopoulos et al., 2020) tests this model with only 735 slot requests, a relatively small-scale instance.

(Ribeiro et al., 2018) introduces the Priority-based Slot Allocation Model (PSAM), which incorporates multiple objectives such as minimizing the total number of rejected slots, the maximum displacement (considered a "guaranteed service level" for any slot), the total schedule delay across all flights, and the number of displaced slots, all while adhering to IATA's priority guidelines. The study demonstrates that PSAM ensures computational feasibility for mid-sized airports like Porto, Portugal, with approximately 100,000 annual movements. However, (Ribeiro et al., 2018) emphasizes the need for further development to address scalability challenges for larger airports. For example, Changi Airport, which recorded over 386,000 movements in 2018, highlights the significant scalability challenge for current methods (Changi, 2022).

(Wang et al., 2019a) proposes a heuristic approach based on PSAM to address the scalability limitations of exact methods, enabling the application to larger instances. The solution involves a two-stage process: first, determining the order of requests for allocation, and sec-

ond, performing the allocation itself. The request order is determined using three distinct ordering heuristics: static deterministic ordering (non-increasing order by the number of requested operation days), dynamic deterministic ordering (based on criteria such as most conflicts, least residual degree, or largest displacement cost), and dynamic stochastic ordering (prioritizing requests with the most operation days and randomly scheduling those with similar days). Unscheduled requests are randomly ordered at subsequent steps. Results demonstrate that prioritizing requests with the most operation days yields better outcomes. This approach provides valuable insights into sequential allocation strategies and highlights potential improvements for future search algorithms.

The aforementioned studies highlight the advantages of search-based methods over exact approaches, particularly in addressing the scalability challenges associated with large instances. The following parts deliver various extensions of the single-airport slot scheduling problem, offering a comprehensive perspective on the existing body of literature.

(Jacquillat and Odoni, 2015) proposes an integrated approach that optimizes both the strategic flight schedule and the tactical utilization of airport capacity, considering scenarios of capacity constraints and delay reduction. This model, following the U.S. scheduling practice, builds on the strategic scheduling framework developed by (Pyrgiotis and Odoni, 2016). The approach involves a two-step optimization: first determining the optimal maximum displacement, δ, and subsequently calculating the optimal total displacement, Δ, based on the derived δ. Applied to John F. Kennedy (JFK) International Airport, the study demonstrates that moderate rescheduling, involving 75–90% of flights, can substantially alleviate congestion at high-traffic U.S. airports.

(Zografos et al., 2018) explores the trade-off between scheduling efficiency and the acceptability of slot schedules by introducing two bi-objective scheduling models, extending the framework developed by (Zografos et al., 2012). In addition to schedule delay, which serves as the metric for efficiency, the models incorporate slot acceptability as a second objective. The underlying rationale is that airlines are generally resistant to significant displacements of their slot requests. Therefore, the models are designed to produce schedules that are

more likely to be accepted. The first model minimizes the maximum displacement, while the second focuses on a "violated slot assignment" objective to evaluate the threshold for unacceptable slot displacements. The findings reveal that significant improvements in schedule acceptability can be achieved with only minimal compromises in scheduling efficiency.

The slot scheduling problem can also be distinguished by the assumptions applied to constraints, which significantly affect the problem's complexity. For instance, declared airport capacity can be assumed as either deterministic or stochastic to evaluate solution feasibility during the tactical phase (Shone et al., 2021). Additionally, capacity may be defined on a rolling basis. For example, a declared capacity of 20 movements per hour on a 5-minute rolling window ensures that airports do not exceed 20 movements within overlapping intervals, such as 10:00–11:00, 10:05–11:05, 10:10–11:10, and so forth (Ribeiro et al., 2019a). These rolling declared capacity constraints introduce an additional layer of complexity to the model.

Other critical constraints include flight connectivity and turnaround time, which are essential for constructing realistic models (Zografos et al., 2017). The series of slots constraint is also significant, requiring at least five slots at the same time and day of the week over a specified period. This constraint differentiates strategic-level slot scheduling, which considers half-year scheduling periods, from pre-tactical and tactical Demand and Capacity Balancing (DCB) problems, where approaches focus solely on slot allocation for the day of operation (Kravaris et al., 2017, 2018, 2019; Fernández et al., 2017).

Minimizing schedule delay, defined as the absolute difference between requested and allocated slots, is the most commonly addressed objective in slot scheduling research. However, focusing solely on schedule delay often fails to yield practically implementable solutions. Fairness and equity are increasingly recognized as critical objectives for generating schedules that are more acceptable in real-world applications (Bertsimas et al., 2011a).

(Zografos and Jiang, 2019) introduces a bi-objective optimization model that simultaneously considers scheduling efficiency and fairness by proposing a novel fairness metric. The study explores the trade-off between fairness and efficiency under two scenarios: (1) adhering to IATA's hierarchical slot allocation framework and (2) without the hierarchical require-

ment. The results reveal that in scenario (1), new entrants and other requests bear a higher "price of fairness" compared to grandfathered slots. In scenario (2), removing the hierarchical allocation improves both schedule delay and fairness. In both cases, fairness can be enhanced with only a minimal sacrifice in scheduling efficiency.

Other notable works addressing fairness in single-airport slot scheduling include (Jacquillat and Vaze, 2018), which focuses on inter-airline equity, and (Fairbrother et al., 2020), which incorporates demand-based fairness. A deeper examination of current fairness metrics in slot scheduling will be presented in a subsequent section.

Summary

- *Preference for Search-Based Methods:*

  Search-based methods are favored over exact optimization techniques due to the computational challenges posed by the NP-hard nature of the slot scheduling problem. Exact methods, while capable of providing optimal solutions, are computationally intensive and impractical, especially for realistic-sized instances, making search-based approaches more suitable.

- *Increasing Complexity with Additional Objectives and Constraints:*

  The inclusion of multiple objectives, such as minimizing schedule delay and enhancing fairness, alongside operational constraints like rolling declared capacity, turnaround time, and flight connectivity, significantly increases the problem's complexity. This growing complexity further highlights the necessity of employing search-based methods.

- *Importance of Fairness:*

  Fairness has emerged as an essential consideration in generating implementable slot schedules. Studies, such as (Zografos and Jiang, 2019), highlight the trade-offs between efficiency and fairness, emphasizing the importance of equitable slot allocation. Addressing fairness alongside efficiency is essential for achieving consensus among stakeholders and ensuring practical implementation.

In the next section, we will explore the current literature on network-level (multi-airport) slot scheduling, which addresses the interdependency between multiple airports in a connected airport system.

2.2.2 Network-Level Slot Scheduling

A strong line of criticism against the current IATA slot scheduling practice is its failure to address the interdependency between multiple airports. This interdependence arises from the need to ensure that a departure slot at one airport aligns with a corresponding arrival slot at the destination airport, taking flight times into consideration. The current approaches of independently scheduling slots at individual airports fail to guarantee a feasible schedule across the entire network of interconnected airports (Corolli et al., 2014; Koesters, 2007; Jones et al., 2004). Single-airport models primarily address Step 3 of the IATA slot scheduling process, focusing exclusively on individual airport operations, which limits their ability to capture network-wide dynamics. To overcome this limitation, a growing body of research in "Network-Level Slot Scheduling" has emerged, aiming to address the complexity and interdependency of slot scheduling across multiple interconnected airports.

(Castelli et al., 2012) develops a network-level slot scheduling model based on the Air Traffic Flow Management (ATFM) framework introduced by (Bertsimas et al., 2011b). The objective of the model is to minimize a "shift cost," which accounts for arrival times and implicitly reflects departure times through flight durations. The model simultaneously assigns departure and arrival slots for each flight while determining the flight routes. These routes consist of a sequence of airspace sectors connecting origin and destination airports, enabling the estimation of flight times and ensuring alignment between allocated departure and arrival slots. An additional cost is incurred if the assigned route is longer than the desired route. The model creates a final network-wide schedule by solving all slot requests simultaneously. However, this approach is computationally intensive. Using the XPRESS optimizer, an Integer Programming (IP) solver, the model requires 30 minutes of CPU time to handle 2200 flights across a network of 60 airports over a 5-hour scheduling horizon. While effective for

small-scale scenarios, scaling the model to realistic instances with a longer time horizon and additional constraints would make the problem computationally infeasible. This example further emphasizes the need for search-based methods to address the challenge of multi-airport slot scheduling.

To handle realistic instances of the network-level slot scheduling problem, (Pellegrini et al., 2012a) proposes metaheuristic methods based on Iterated Local Search and Variable Neighborhood Search algorithms. These techniques aim to enhance computational efficiency and scalability, making them suitable for larger networks. However, the metaheuristics in (Pellegrini et al., 2012a), like many heuristic approaches, rely significantly on expert domain knowledge to structure the search process, which can lead to sub-optimal solutions (Pellegrini et al., 2017; Zhang et al., 2020b,a; Wang et al., 2019b).

(Corolli et al., 2014) extends the network-level slot scheduling problem by incorporating stochastic, time-dependent airport capacity to address variability and reduce operational delays. Building on (Zografos et al., 2012), the model aims to minimize total schedule delays and expected operational delays. It assumes fixed flight times, linking slots at origin and destination airports, and uses a two-stage approach: generating a network schedule and estimating delays across capacity scenarios. The model is solved using the Sample Average Approximation (SAA) method, requiring up to 40 hours of computation for networks of 5-10 airports with 6520 to 9267 flight requests.

(Pellegrini et al., 2017) extends the model of (Zografos et al., 2012) to incorporate network effects under various capacity constraints while explicitly considering turnaround time constraints. The model, formulated as an integer linear program, is solved using the CPLEX IP solver and tested on a dataset of 32,665 movements (arrivals and departures) across 152 airports for a single day of operations. While the model achieves optimal solutions, it is limited to relatively small instances and does not address fairness considerations.

(Benlic, 2018) enhances the model proposed in (Pellegrini et al., 2017) by addressing a relaxed slot coherency assumption. It ensures strict adherence to the requirement that the arrival time at the destination airport equals the departure time at the origin airport plus

the flight time between them. The study evaluates more extensive scenarios, with the largest instance involving 4,620,854 movements across 100 airports over 181 days. Given the size of these instances, exact methods are computationally impractical. To address this challenge, (Benlic, 2018) introduces a two-phase heuristic approach called the Strategic Slot Allocation at Network-Level (SSANL). However, the dataset used in the study is synthetic, generated with simple distributions, rather than real-world data, limiting the practical applicability of the findings.

Existing network-level approaches predominantly adopt a centralized paradigm, solving all initial requests from all airports simultaneously to produce a final network schedule. This centralized approach overlooks the distinct characteristics and operational differences of individual airports, as incorporating all their different objectives and constraints would create an overly complex model with numerous objectives and constraints.

(Keskina and Zografos, 2022) tackles the network-level slot scheduling problem by utilizing initial slot allocations from individual airports as inputs to create a feasible network-wide schedule. The model is formulated as an integer linear program and focuses on minimizing total frequency-weighted schedule delay. Additionally, it incorporates metrics to evaluate the importance of airports, highlighting the identification of "critical" airports within the network. Objectives such as maximum displacement and inter-airline fairness are also considered, though they are transformed into constraints using the ε-constraint method.

While (Keskina and Zografos, 2022) suggests that initial schedules implicitly account for individual airport differences, the lack of explicit integration of the different objectives and constraints may compromise their preservation in the final network schedule. The model is tested on a relatively small instance comprising 3,006 requested series across 16 coordinated airports, totaling 156,645 movements. However, due to its reliance on traditional optimization (CPLEX solver), the model's scalability to larger networks remains computationally infeasible.

Summary

- *Centralized Framework Limitation:*

  Current network-level slot scheduling approaches predominantly adopt a centralized framework, where all initial requests from all airports are solved simultaneously to produce the final network schedule. While this paradigm simplifies the optimization process, it overlooks the unique characteristics of individual airports. Accounting for these individual airport requirements would create highly complex models with multiple objectives and constraints, making them computationally impractical for large-scale networks.

- *Limited Attention to Fairness in Network-Level Models:*

  Fairness, a critical consideration in resource allocation, has not been widely addressed in network-level slot scheduling due to its complexity and scale. As network-level models involve numerous interconnected airports and large datasets, integrating fairness becomes challenging. This gap highlights the need for scalable methods that can effectively address fairness in a multi-airport system.

The literature review examines various methods to enhance the multi-airport slot scheduling process. However, focusing solely on system efficiency, such as minimizing total schedule delay, is insufficient for developing implementable solutions. Fairness plays a critical role in ensuring that the solutions are acceptable to all relevant stakeholders (Bertsimas et al., 2011a). In the next section, we explore existing fairness approaches and examine how fairness is addressed in both single-airport and network-level (multi-airport) contexts.

2.3 Fairness in Airport Slot Scheduling

Although various models in the literature address fairness in different forms, incorporating fairness into the airport slot scheduling problem primarily follows three key approaches: (1) the mini-max criterion, (2) the principle of proportionality, and (3) constraint-based methods.

2.3.1 Mini-Max Criterion

The mini-max criterion aims to minimize the maximum displacement encountered by the most disadvantaged slot request. While the concept originates from game theory, it was influentially formulated and firmly embedded in political philosophy by John Rawls through his maximin principle in "A Theory of Justice" (Rawls, 1971). Slot displacement is defined as the absolute difference between the requested and the allocated time slot of a specific movement. The corresponding formula is presented below:

$$\text{Minimize} \quad \max_{m \in M} |t_m - t'_m| \,, \tag{2.1}$$

where t_m is the requested time slot and t'_m is the allocated time slot of the movement m.

By minimizing the maximum displacement across (flight) movements, the mini-max criterion indicates inter-flight fairness by ensuring that the worst-off flight experiences the smallest possible displacement (Jacquillat and Odoni, 2015; Zografos et al., 2018; Ribeiro et al., 2018, 2019b).

(Jacquillat and Odoni, 2015) introduced a bi-objective slot allocation model that jointly optimizes the mini-max displacement (fairness) and total displacement (efficiency). The model is solved lexicographically, where the mini-max objective is optimized first, and its value is then used as a constraint to minimize the total displacement. This approach was integrated into a framework for optimizing strategic slot allocation decisions and mitigating airport congestion at the tactical level.

Building on this concept, (Zografos et al., 2018) proposed two formulations within the IATA slot scheduling process. The first formulation focuses on minimizing both the maximum displacement (inter-flight fairness) and total displacement across all slot requests, while the second formulation seeks to minimize the total displacement and the number of violated slots. Both models employed the ϵ-constraint method to explore the trade-off between fairness and efficiency by varying the ϵ value, which defines the maximum acceptable displacement.

(Ribeiro et al., 2018, 2019b) extended these approaches by developing a multi-objective model to minimize total displacement, maximum displacement, number of unassigned slots, and number of displaced slots. A weighted method was used to optimize these objectives, with priorities assigned to minimizing the number of unassigned slots, followed by maximum displacement, total displacement, and displaced slots. This model highlighted the interrelationship between fairness and efficiency in the allocation process.

These studies demonstrate the application of the mini-max criterion for fairness in airport slot allocation. By minimizing disparities among flights, this method promotes inter-flight fairness to analyze the trade-off between fairness and efficiency in slot scheduling.

(Jacquillat and Vaze, 2018) applied the min-max criterion to address inter-airline fairness in slot scheduling by focusing on minimizing two fairness-related objectives: the maximum cost of missed allocations and the maximum total cost across all airlines. Their approach introduced the concept of airline disutility, defined as the weighted average of per-flight displacements for each airline, which quantified the cost of displacement on an airline's schedule. To consider inter-airline equity, the model employed a lexicographic optimization strategy. This method systematically minimized disparities by prioritizing the reduction of the largest airline disutility first, followed by the second largest, and so on. This hierarchical approach allowed the model to analyze trade-offs between efficiency and fairness, providing a structured framework for addressing inter-airline fairness. The results show that neglecting inter-airline equity in favor of purely efficiency-based objectives could lead to significantly inequitable outcomes.

(Jiang and Zografos, 2021), in addition to the fairness measure that minimizes the maximum distance from average fairness proposed by (Zografos and Jiang, 2019), introduce two additional inter-airline fairness objectives: minimizing the maximum distance from absolute fairness and minimizing the Gini index. These objectives are incorporated individually into a bi-objective efficiency-fairness model to generate the corresponding efficient frontiers.

To the extent of our literature search, only one study addresses fairness in the network-level (multi-airport) context of slot scheduling. (Pellegrini et al., 2017) propose two variants

to incorporate fairness at the network level. Both variants aim to minimize the maximum cost of missed allocations per airline and the maximum total displacement costs across all airlines and airports. The difference between the two variants lies in the consideration of airline size in the second approach, where the objectives are normalized by dividing the costs by the number of requests made by each airline. This adjustment ensures that fairness is evaluated relative to the scale of each airline's operations. The proposed model is formulated as an integer linear programming model and solved using the CPLEX IP solver. It is tested on a dataset involving 32,665 movements, which represents a relatively small-scale instance. Consequently, the approach lacks scalability and is not suitable for handling large-sized instances.

2.3.2 Principal of Proportionality

(Zografos and Jiang, 2016, 2019) developed models for inter-airline fairness in the IATA context, employing the principle of proportionality. This approach allocates total displacement proportionally to the number of slot requests submitted by each airline, ensuring a fair distribution of delays. They introduced a fairness objective to minimize the maximum absolute deviation of an airline's displacement from the average displacement metric across all airlines. (Zografos and Jiang, 2019) demonstrates a trade-off between system efficiency and fairness within the same hierarchy level of slot requests. Results show that the extra delay required to achieve a specific level of fairness varies across different hierarchy levels. The fairness objective in this context is quantified using a fairness index:

$$\mathrm{p_a} = \frac{\frac{\mathrm{d_a}}{D}}{\frac{|\mathrm{M_a}|}{\sum_{a \in A} |\mathrm{M_a}|}}, \forall a \in A \tag{2.2}$$

The fairness index, denoted as $\mathrm{p_a}$, is calculated as the ratio of two components: the numerator represents the proportion of schedule delays assigned to an airline, while the denominator corresponds to the proportion of slots requested by the airline. The underlying principle of this index is that the allocated schedule delays should be proportional to the

airline's share of requested slots. A value of $p_a = 1$ indicates absolute fairness, where the airline is treated equitably, while deviations from 1 imply that the airline either benefits or is disadvantaged compared to the norm.

The fairness objective aims to minimize the maximum distance between the fairness index of an airline and the average fairness indexes across all airlines:

$$Min\{Max\{|p_a - \frac{\sum_{a \in A} p_a}{|A|}, \forall a \in A|\}\} \tag{2.3}$$

The ideal scenario of absolute fairness is achieved when the variance of the fairness index is zero, meaning that the fairness indexes of all airlines are equal to one. In this case, all airlines experience proportional delays relative to their requested slots. However, achieving this perfect scenario is infeasible due to inherent operational and capacity constraints. (Zografos and Jiang, 2016) extends the fairness metric by incorporating weighted delays. These weights account for the aircraft's size and the flight length associated with the slot, providing a more nuanced representation of fairness in slot scheduling.

In the context of the Ground Delay Program (GDP), a U.S.-based practice, (Manley and Sherry, 2010) implemented an equity rule to allocate delays among airlines. This rule ensured the proportional distribution of delays based on the number of flights operated by each airline, promoting fairness in delay management.

A notable criticism of the above approaches is their equal treatment of all slot requests, regardless of whether they pertain to peak or off-peak periods. This uniform handling can lead to unnecessary adjustments for off-peak requests, which do not significantly contribute to scheduling infeasibility. To address this issue, (Fairbrother et al., 2020) introduced a demand-based fairness metric that explicitly differentiates between peak and off-peak requests, avoiding penalties for the latter. Building on the proportionality metric, this demand-based fairness approach emphasizes equitable allocation of displacement, specifically during high-demand periods.

2.3.3 Constrained-Based Approaches

Approaches that consider fairness using constraint-based methods incorporate fairness into the model as strict constraints rather than as an optimized objective. For instance, (Jacquillat and Odoni, 2015) incorporate two constraints into the scheduling model to define the maximum displacement per flight and the total flight displacement:

$$|u_i| \leq \delta, \quad \forall i \in \mathcal{F}, \tag{2.4}$$

$$\sum_{i \in \mathcal{F}} |u_i| = \Delta, \tag{2.5}$$

where u_i is the displacement of flight i, which cannot exceed the predefinded value δ, and the total displacement across all flights cannot exceed Δ.

The maximum displacement, often considered a "guaranteed service level," acts as a fairness indicator by enforcing a constraint that limits the schedule delay allocated to any flight to a specific threshold. This approach establishes a baseline level of fairness by reducing extreme disparities in displacement. However, constraint-based methods do not explicitly address the variations in schedule delays assigned to individual flights, which may still result in uneven treatment among flights and airlines. Whether it is more effective to explicitly address these disparities or to rely on a defined threshold remains an open question. Explicitly addressing uneven treatment could potentially lead to a reduction in overall system efficiency, raising further considerations about the trade-off between fairness and efficiency.

2.4 Research Gaps

Numerous approaches have been proposed in the literature to improve airport slot scheduling, ranging from exact optimization methods to search-based approaches. However, despite these extensive efforts, practical implementations remain largely absent in real-world applications.

Based on our literature review, several key research gaps have been identified, highlighting the need for new approaches to address the limitations of current methodologies.

2.4.1 Limitations of Exact Methods and Heuristics

Exact methods, which are traditional optimization techniques designed to find optimal solutions, face significant limitations when applied to slot scheduling problems, especially at the network level, due to scalability issues. As a result, there is a growing need to consider alternative search-based approaches, such as heuristics and learning-based methods, rather than relying on exact solutions (Zografos et al., 2017).

Heuristics are rule-based methods derived from expert domain knowledge. These approaches are typically tailored to specific cases, limiting their ability to generalize to broader or varied contexts. As heuristics rely on predefined and rigid sets of rules, they often struggle to adapt to dynamic environments or unforeseen circumstances. Consequently, they fail to achieve optimal solutions in such situations (Wang et al., 2019b; Zhang et al., 2020b).

2.4.2 Unrealistic Centralized Framework

Existing approaches in the literature adopt a centralized framework that processes all initial slot requests from multiple airports to generate a global network schedule. While theoretically appealing, this assumption is impractical in regions lacking a central governing body, such as Association of Southeast Asian Nations (ASEAN), where airports reserve autonomy in making decisions and operate under different objectives, constraints, and local guidelines. The absence of a unified authority makes it challenging to enforce centralized solutions effectively (Duong et al., 2019).

Furthermore, the complexity of simultaneously addressing the diverse objectives and constraints of numerous airports leads to computational infeasibility. This emphasizes the need to investigate alternative frameworks capable of addressing multi-airport slot scheduling while respecting the autonomy of individual airports.

2.5 Research Questions

The current slot scheduling methods discussed in the literature are not practically applicable due to the aforementioned limitations. A practical solution must effectively address the slot scheduling problem in a multi-airport system. Exact methods in the literature face computational infeasibility when applied to multiple airports. Heuristic methods, being highly case-specific and reliant on expert knowledge, are less effective in addressing the dynamic and complex nature of multi-airport systems. Centralized frameworks, while easier for optimization, are unsuitable for regions lacking a central authority, such as ASEAN. Solving all airports in a network simultaneously leads to an overly large and computationally challenging problem, emphasizing the need to explore alternative frameworks. Lastly, for any approach to be implementable, fairness must be incorporated. To address these challenges, this thesis targets the following research question:

1) If solving all requests simultaneously across a multi-airport system results in a large joint problem, is there an alternative paradigm that can effectively scale down the problem?

2) Is there an optimization mechanism, other than exact methods and heuristics, that can effectively integrate into the new paradigm while addressing the challenges of multi-airport slot scheduling?

3) Can the new method incorporate fairness considerations? What is the trade-off between fairness and efficiency associated with the new method?

2.6 Motivation for Adopting Reinforcement Learning

Although heuristic methods have demonstrated the ability to manage the complexity of airport slot scheduling, they are not designed for decentralized decision-making. Applying heuristics to decentralized settings would require developing multiple custom heuristics for different airports with varying objectives and constraints, leading to poor scalability and limited coordination capability. In contrast, Reinforcement Learning (RL) is capable of learning decision policies through interaction and feedback, making it well-suited for decentralized co-

ordination among airports. We outline below the key motivations for adopting Reinforcement Learning (RL) as the core methodology of this study:

An alternative to heuristics:

Heuristic approaches rely heavily on hand-crafted rules, domain expertise, and trial-and-error tuning (Zhang et al., 2020a; Wang et al., 2019b; Mazyavkina et al., 2021). RL, on the other hand, automates policy discovery and leverages historical data to continuously improve decision quality over time. This allows the learning agent to optimize slot allocation through experience rather than explicit modeling (Sutton and Barto, 2018). Modern RL methods, such as Deep Q-Networks (DQN) (Mnih et al., 2015) and AlphaGo (Silver et al., 2017), have demonstrated the ability to solve complex, large-scale decision-making problems, underscoring their relevance to airport slot scheduling.

Applicability to Resource-Constrained Scheduling Problems (RCSPs):

The slot scheduling problem shares structural similarities with RCSPs, which involve assigning tasks under capacity and precedence constraints. In this analogy, slot requests correspond to activities, airport capacity represents the limited resource, and turnaround or connectivity constraints mirror precedence relationships. RL methods have achieved competitive results in solving RCSPs with earliness–tardiness objectives (Wang et al., 2019b; Luo, 2020). RL has also been successfully applied to related air traffic management problems (Spatharis et al., 2019; Kravaris et al., 2018), demonstrating its adaptability to dynamic and capacity-constrained scheduling contexts.

Relevance to decentralized coordination:

In practice, slot allocation involves local decision-making at each airport. RL has an established and growing literature in multi-agent and decentralized learning, where independent agents coordinate through interaction rather than centralized control (Gronauer and Diepold, 2022). Such frameworks enable scalable coordination among airports without re-

quiring global re-optimization of the entire network, making them ideal for regions without centralized scheduling authorities.

Multi-objective optimization and fairness:

Beyond efficiency, slot scheduling must balance multiple objectives, such as minimizing delay while maintaining fairness across flights, airlines, and airports. RL supports multi-objective reward formulations, enabling trade-offs between efficiency and fairness (Cui et al., 2015; Sheikh and Bölöni, 2020). Recent studies have also shown how fairness can be integrated into multi-agent RL through explicit or implicit reward design (Chen et al., 2021; Jiang and Lu, 2019).

In summary, RL offers a scalable and decentralized solution to airport slot scheduling, addressing the major limitations of heuristic and centralized methods. It provides a unified, data-driven framework capable of learning from experience, coordinating across airports, and incorporating fairness, justifying the adoption of Reinforcement Learning as the core methodology in this study.

Chapter 3

Methodology

Improving the current ad-hoc slot scheduling process is essential for mitigating airport congestion. A practical approach for slot scheduling must address the problem in a multi-airport context while considering fairness. Ensuring an equitable distribution of decisions enhances stakeholder acceptance and fosters greater collaboration in the decision-making process.

Because of the large number of flights, exact methods are computationally infeasible for large-scale multi-airport systems. Heuristics, on the other hand, rely heavily on expert knowledge and struggle to adapt to the interconnected nature of a multi-airport system, where decisions made by individual stakeholders can have system-wide effects. Centralized approaches, although theoretically appealing, are impractical in regions without a central authority, such as ASEAN airport aviation authorities. Moreover, addressing multiple airports simultaneously leads to excessively large joint problems. For example, formulating a linear programming model for a system with three airports, each handling approximately 1,000 flights and operating across 288 five-minute time slots, would result in more than 1.8 million binary variables. Therefore, alternative frameworks and suitable optimization mechanisms must be investigated.

This chapter presents a framework for addressing the multi-airport slot scheduling problem using Reinforcement Learning (RL). We demonstrate why RL is a suitable solution and detail the steps involved in applying RL to airport slot scheduling. The chapter concludes with an overview of the data scope and rationales of this research.

3.1 Decentralized Framework for Multi-Airport Slot Scheduling

Centralized approaches, which solve all initial flight requests (movements) simultaneously for multiple airports, face significant challenges due to the exponential nature of problem size with the number of requests and airports involved. This makes it sensible to explore a decentralized approach, where the multi-airport problem is divided into smaller and manageable sub-problems. Figure 3.1 illustrates the centralized and decentralized paradigms for multi-airport slot scheduling. It also demonstrates the two levels of division within the decentralized paradigm, division based on initial requests and division based on initial slot allocations, as follows:

- *Division Based on Initial Requests:*

  This approach creates sub-problems by matching departure and arrival slots for individual requests. However, it leads to an overwhelming number of sub-problems, as it scales directly with the number of flights. The number of sub-problems can be at least equal to the total number of flights plus the number of required adjustments to flight schedules. Furthermore, optimizing global objectives, such as minimizing total schedule delay, becomes highly suboptimal due to the excessive fragmentation of the sub-problems.

- *Division Based on Initial Slot Allocations:*

  In the division based on the initial slot allocations (airport level), initial schedules are first obtained locally at individual airports. The decentralized problem-solving then focuses solely on adjusting non-coherent requests from these initial schedules. This approach aligns with the current IATA practice, where flight slots are initially allocated at the individual airport level and subsequently adjusted at the network (multi-airport) level to ensure overall system efficiency. This paradigm offers several advantages. By first obtaining schedules locally, airport-specific objectives and constraints are opti-

mized to some extent, and adjusting only the non-coherent requests afterward preserves both local and global requirements. At the individual level, the number of flights subject to specific airport objectives and constraints is smaller, making the problem more manageable compared to optimizing all flights across the entire airport system under joint objectives and constraints. Adjusting only non-coherent requests significantly reduces the number of sub-problems. Most importantly, this paradigm aligns with Steps 3 and 4 of the IATA slot scheduling process, ensuring consistency with the current practice.

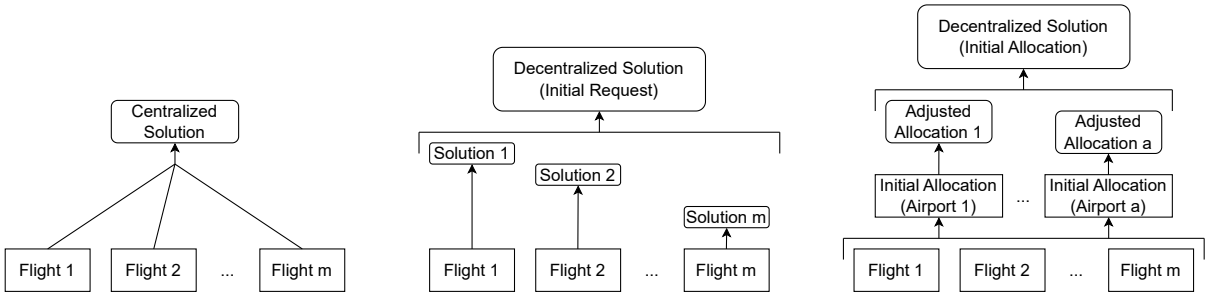


Figure 3.1: Centralized paradigm, decentralized paradigm (initial request division), and decentralized paradigm (initial slot allocation division).

The decentralized approach, based on the division of initial slot allocations, effectively balances optimizing individual airport schedules while achieving global network-level objectives. This paradigm provides a scalable and practical framework for tackling multi-airport slot scheduling challenges.

Figure 3.2 provides a detailed representation of the proposed framework. At the individual airport level, slot coordinators handle the initial slot allocation tasks based on the corresponding requests and capacity. This step is conducted independently for each airport, adhering to the airport-specific objectives and constraints, resulting in an initial slot allocation (initial schedule). These initial schedules are then utilized as inputs to formulate a cohesive global network schedule. The entire process mimics the IATA scheduling practice (Steps 3 and 4), ensuring compatibility and seamless integration into existing procedures.

The next challenge lies in determining how to perform schedule adjustments at the network level. In regions without a central authority, airports maintain autonomy in their

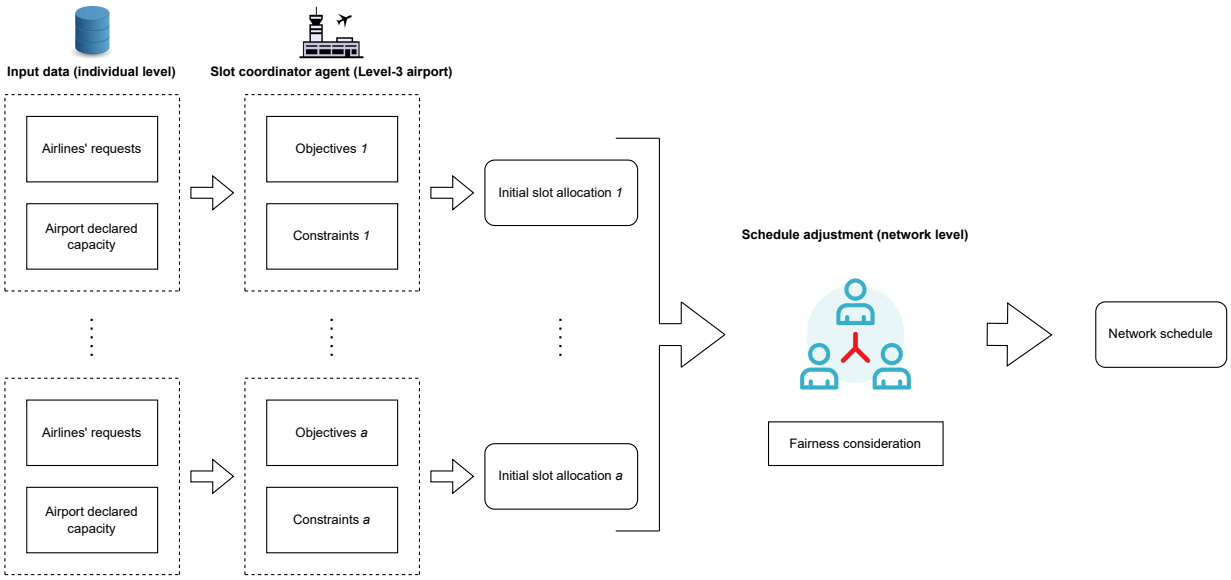


Figure 3.2: Concept diagram for the decentralized paradigm.

decision-making processes, making a centralized framework for combining initial schedules impractical. Unaligned stakeholders will lead to backtracking and altering the centralized global solution.

Figure 3.3 illustrates a proposed protocol for airport agents (slot coordinators) to handle non-coherent requests. Under this protocol, the adjustment process takes place between the departure and arrival airports associated with the non-coherent request. Initially, the departure airport agent proposes an adjustment based on both departure and arrival information. The arrival airport agent evaluates the proposal and decides whether to accept or reject it. If accepted, the adjustment is finalized. If rejected, the arrival airport agent updates the arrival information and sends it back to the departure airport agent, enabling the generation of revised adjustment proposals. This protocol ensures that adjustments are made in a decentralized manner, respecting the autonomy of individual airports while avoiding the computational challenges associated with a centralized framework due to the explosion of a joint problem.

Following this protocol, traditional optimization methods and heuristics are not suitable as decision-making mechanisms because updated information requires modifications to the optimization problem. Traditional optimization methods would necessitate changes to the

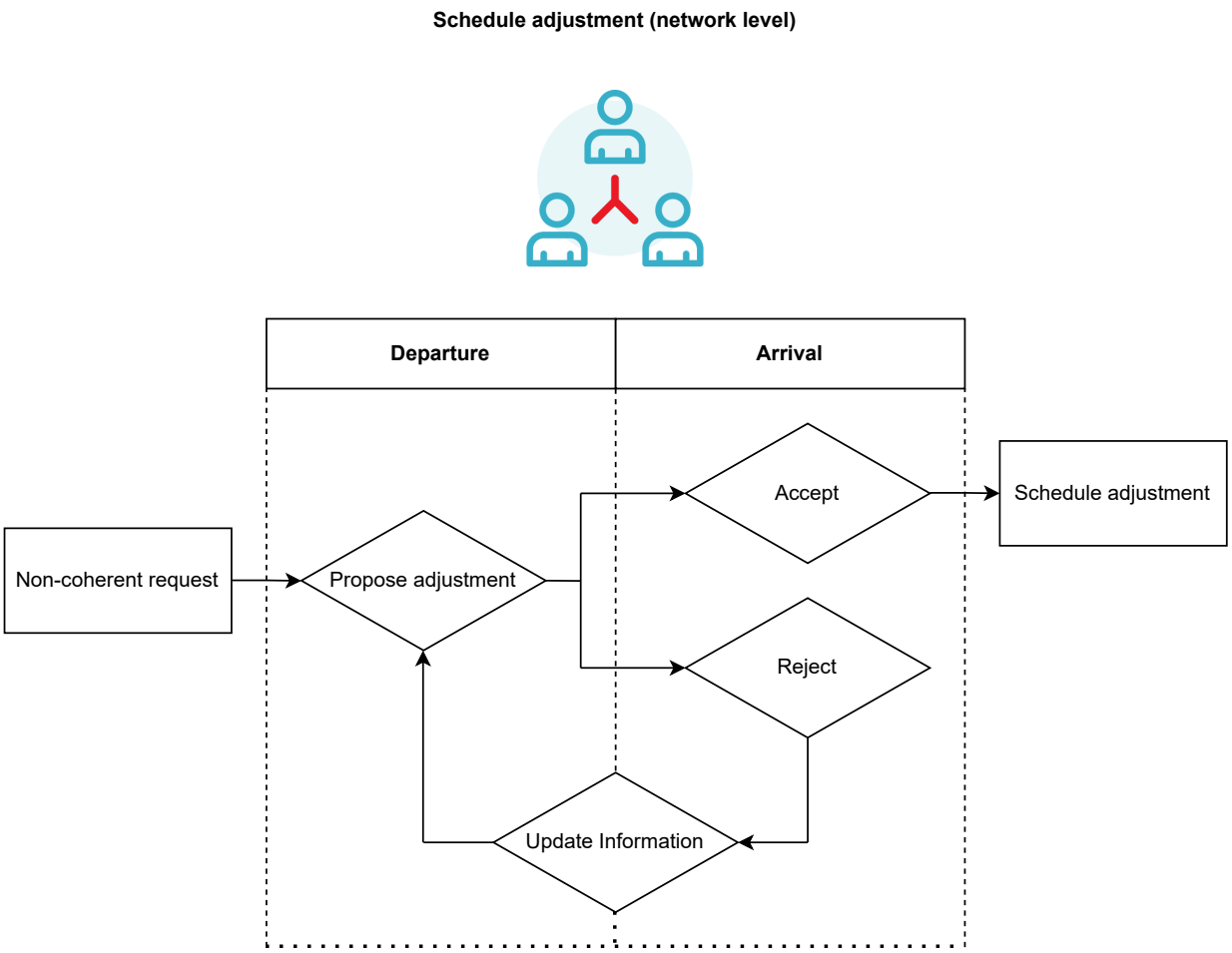


Figure 3.3: Adjustment protocol between the departure and arrival airports.

mathematical formulation, while heuristics, being a fixed set of rules, are less adaptable and may struggle in dynamic settings.

In the following section, we propose Reinforcement Learning as a decision-making approach, supported by existing literature that justifies its suitability for the problem.

3.2 Reinforcement Learning for Airport Slot Scheduling

3.2.1 Justifications for Adopting Reinforcement Learning

Following the proposed protocol, exact methods and heuristics are not suitable due to several reasons. Exact methods, which are designed to solve problems globally and simultaneously, are infeasible in decentralized systems where each airport independently adjusts non-coherent

requests. They are also rigid and lack the flexibility to adapt to incremental changes inherent in the iterative updates of the protocol. Heuristics rely heavily on predefined rules and domain expertise, which do not generalize well across airports with unique objectives and constraints. Heuristics also struggle in dynamic and decentralized environments, making them inefficient in responding to iterative changes in the problem context, such as updates from the arrival airport. Given these limitations, learning-based approaches like Reinforcement Learning (RL) are more suitable for the following reasons:

- *Enhancing Search Automation and Generalization:*

  Reinforcement Learning (RL) emerges as an alternative to heuristic methods and demonstrates significant potential in addressing the complexities of the airport slot scheduling problem. Unlike heuristics, RL automates the search process and relies less on predefined rules and extensive domain knowledge. This generalization capability of RL allows it to adapt more effectively to complex and dynamic airport environments, where each airport may have unique objectives, constraints, and operational requirements. While heuristics often require significant manual intervention and tailoring to specific scenarios, RL provides a more flexible framework that can dynamically learn and adjust based on the problem context (Zhang et al., 2020b,a; Wang et al., 2019b).

  By leveraging historical data, RL offers a distinct advantage in slot scheduling, where airlines' slot requests often exhibit recurring seasonal patterns. This capability makes RL not only efficient in handling repeated patterns but also robust in adapting to new or evolving scheduling challenges. Its ability to generalize enables consistent performance across different airports within a network, making it particularly effective for dynamic and iterative problem-solving in complex multi-airport scheduling environments (Mazyavkina et al., 2021).

- *Applicability Proven in Related Problems:*

  The slot scheduling problem is a type of Resource-Constrained Scheduling Problem (RCSP), which involves scheduling activities while adhering to resource and other prece-

dence constraints. In the slot scheduling context, each slot request is treated as a project activity, subject to resource constraints like airport capacity (Androutsopoulos et al., 2020).

Reinforcement Learning (RL) has demonstrated promising results in various RCSP contexts, especially for earliness-tardiness optimization objectives, which closely resemble the schedule delay minimization goals in slot scheduling (Zhang et al., 2020b; Wang, 2020; Luo, 2020). RL-based algorithms, such as Deep Q-Networks (DQN), have shown an ability to surpass heuristic methods by effectively managing dynamic changes and uncertainty (Waschneck et al., 2018; Palombarini and Martínez, 2019).

The pre-tactical Demand and Capacity Balancing (DCB) problem, which shares similarities with slot scheduling, also aims to minimize schedule delays while adhering to capacity constraints. RL has been successfully applied to pre-tactical DCB scenarios, where agents resolve congestion through delay management decisions (Spatharis et al., 2021; Kravaris et al., 2019). These successes highlight the potential of RL in addressing slot scheduling challenges.

- *Ability to Address Multi-Objective Problems:*

  Reinforcement Learning (RL) provides robust capabilities for addressing multi-objective problems, making it highly suitable for airport slot scheduling with fairness consideration. RL has been effectively applied in multi-objective project scheduling problems (Wang et al., 2019b; Cui et al., 2015; Sheikh and Bölöni, 2020).

  RL can incorporate multiple objectives into a unified reward function. For example, in Air Traffic Flow Management (ATFM), RL approaches employ reward functions with components addressing different priorities: local rewards minimize individual flight delays, social rewards prevent sector congestion, and global rewards aim to minimize overall delays (Chen et al., 2021). In this work, fairness is incorporated within the reward structure, where a coefficient λ controls the balance between minimizing total system delays and reducing individual delays. The study shows that fairness can be

achieved using an independent Deep Q-Network (DQN) approach, demonstrating the flexibility of reward design to balance fairness and efficiency.

The ability of Reinforcement Learning (RL) to address multi-objective problems enables the incorporation of both fairness and efficiency, making it a potential approach for developing practical and implementable solutions to the slot scheduling problem. In a later section, we will elaborate on incorporating fairness into RL formulations and further highlight its applicability.

The novelty of this study lies in developing a unified Reinforcement Learning (RL) framework that integrates three key aspects: IATA-aligned operational design, decentralized coordination, and fairness integration. The framework follows current IATA slot coordination practices, evolving from single-airport allocation to multi-airport coordination and fairness-aware scheduling. It introduces a decentralized architecture, where each airport operates as an independent RL agent, enabling coordinated decision-making in regions without a central scheduling authority. Fairness is incorporated directly into the RL formulation, allowing the model to balance between system efficiency and fairness. In contrast, existing approaches rely on centralized frameworks, which limit practical applicability in decentralized operational contexts and require full re-optimization whenever schedule updates occur.

In the next section, we provide the background of Reinforcement Learning to establish a foundation for its application in the airport slot scheduling problem.

3.2.2 Reinforcement Learning Background

Traditional RL relies on tabular-based methods that estimate the values of all possible state-action pairs, representing every potential interaction between the agent and the environment (Sutton and Barto, 2018). However, these approaches' scalability and generalization capabilities are limited, making them unsuitable for large and complex problems.

To address these limitations, (Mnih et al., 2015) introduced a combination of deep learning and RL, achieving human-level control in the game Atari. Similarly, Deep Reinforcement Learning (DRL) demonstrated superhuman performance with AlphaGo, a program

that defeated Go champion Lee Sedol in March 2016 (Silver et al., 2017). The game of Go is renowned for its complexity, with approximately 10^{172} possible board positions. These successes highlight the potential of RL, particularly DRL, in solving highly complex and large-scale problems.

Reinforcement Learning (RL) operates on the principle of trial and error, where an agent interacts with a learning environment and evaluates its actions based on feedback, or reward signals, provided by the environment. This iterative process is modeled as a Markov Decision Process (MDP). The main components of an MDP include:

- *States (S):* These represent the various statuses the agent (decision-maker) can occupy. In the context of airport slot scheduling, the states might include the set of requests that need to be allocated and the capacity information of the involved airports.

- *Actions (A):* These are the choices available to the agent that determine transitions between states. For airport slot scheduling, actions could represent assigning a specific time slot to a request.

- *Transition Probabilities (P):* These define the probability of transitioning to a new state s' given the current state s and the action taken a. Transition probabilities encapsulate the dynamics of the environment as it responds to the agent's decisions.

- *Rewards (R):* Rewards quantify the feedback received based on the actions performed. The agent's objective is to maximize the cumulative reward over time, balancing immediate and long-term gains.

- *Discount Factor (γ):* The discount factor determines the relative importance of future rewards compared to immediate rewards. A higher γ values long-term rewards, while a lower γ prioritizes immediate gains.

An MDP is formally represented as $m = (S, A, P, R, \gamma)$. A policy π is a mapping from states s to actions a. It can be deterministic, where a fixed action a is specified for each state s, or stochastic, where a probability distribution over all possible actions is defined for each

state. The agent's ultimate goal is to learn a policy that maximizes the expected cumulative reward over time.

The state value $V_\pi(x)$ represents the expected average reward an agent would receive by visiting a specific state x repeatedly under a given policy π. Similarly, the state's action value $Q_\pi(x, u)$ denotes the average reward the agent would obtain by performing a specific action u in state x, following the same policy. These value functions, which are known as Bellman equations, can be mathematically calculated using Equation 3.1 and Equation 3.2 (Sutton and Barto, 2018).

$$V_\pi(x) = \sum_u \pi(u|x) \sum_{x',r} p(x', r|x, u) \left[r + \gamma V_\pi(x')\right] \tag{3.1}$$

$$Q_\pi(x, u) = \sum_{x',r} p(x', r|x, u) \left[r + \gamma \sum_{u'} \pi(u'|x') Q_\pi(x', u')\right] \tag{3.2}$$

where x is the *current state*, x' is the *next state*, u is the *current action*, u' is the *next action*, r is the *immediate reward* received after taking action u in state x, and $p(x', r|x, u)$ represents the *joint probability* of transitioning to state x' and receiving reward r given the current state-action pair (x, u).

For finite Markov Decision Processes (MDPs), the optimal policy can be determined by utilizing a recursive relationship between the value function, or the state-action value function, of the current and successor states. The choice of using $V_\pi(x)$ or $Q_\pi(x, u)$ depends on the specific algorithm employed. Equation 3.3 and Equation 3.4 illustrate the process of deriving the optimal policy π^* by leveraging the state-action value function $Q^*(x, u)$:

$$Q^*(x, u) = \max_\pi Q_\pi(x, u) = \sum_{x',r} p(x', r|x, u) \left[r + \gamma \max_{u'} Q^*(x', u')\right] \tag{3.3}$$

$$\pi^*(x) = \arg\max_u Q^*(x,u) = \arg\max_u \sum_{x',r} p(x',r|x,u)\left[r + \gamma Q^*(x',u')\right] \tag{3.4}$$

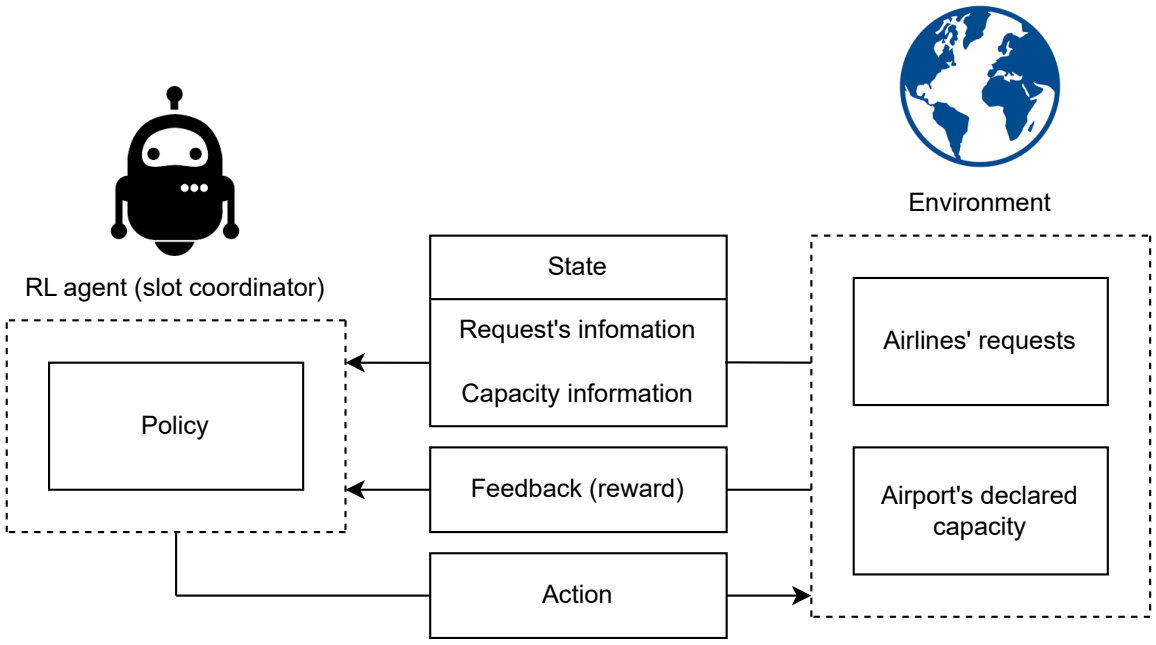


Figure 3.4: Interaction between the RL agent and the environment.

Figure 3.4 illustrates the fundamental interactions between the learning environment and the agent. The learning environment generates scenarios and encapsulates relevant information within the state observation, which is provided to the agent. Based on this observation, the agent takes an action that modifies the state of the environment. The environment then updates the state and delivers a reward signal to the agent. This reward is used by the agent to refine its decision-making policy. Unlike rule-based or traditional optimization algorithms, RL agents adapt and refine their policies based on ongoing feedback during these interactions. This capability enables RL to continuously improve and update its policy when new data is integrated into the environment.

3.2.3 Applying Reinforcement Learning

Following the proposed framework 3.2, the application of Reinforcement Learning (RL) to airport slot scheduling with fairness consideration requires the development of three essential components. The first component, "Reinforcement Learning for Single-Airport Slot Scheduling", focuses on utilizing RL to optimize the initial slot allocations at individual airports. The second component, "Reinforcement Learning in a Multi-Airport Slot Scheduling System", evaluates the performance of an RL agent within a multi-airport system. Finally, the third component, "Incorporating Fairness into Reinforcement Learning for Airport Slot Scheduling", integrates fairness considerations into the RL framework, analyzing the trade-off between efficiency and fairness.

Reinforcement Learning for Single-Airport Slot Scheduling

In order to apply Reinforcement Learning (RL) to any problem, it is essential to establish a well-defined RL formulation. This involves designing the Markov Decision Process (MDP), which encapsulates the learning environment, and constructing the architecture of the RL model. The RL model comprises the action space, state observation (inputs for decision-making), reward design, and algorithm selection. In this module, we aim to develop an RL formulation that provides a foundational basis for further applications of RL in the airport slot scheduling context.

Figure 3.5 illustrates the overall concept for the first module. It begins with data generation, where scenarios are created based on predefined distributions of airlines' requests and airport capacities. Each scenario initiates an initial slot allocation process. The state observations provided to the agent can either be full state observations or local state observations. While full state observations encompass all the information, their large size can hinder the learning process. It is essential to identify an effective state observation.

Reward structures play a vital role in guiding the agent. In this component, we focus on two key objectives: minimizing total schedule delay (efficiency) and minimizing the maximum delay across requests (fairness). Rewards are instrumental in ensuring stable training

and achieving optimal performance. Different combinations of state observations and reward structures lead to distinct models. These models are then analyzed for training efficiency, specifically their ability to achieve convergence and maintain decision-making stability. Models that fail to converge exhibit instability and poor predictive performance. Successfully converged models are further evaluated and compared to identify the most effective model.

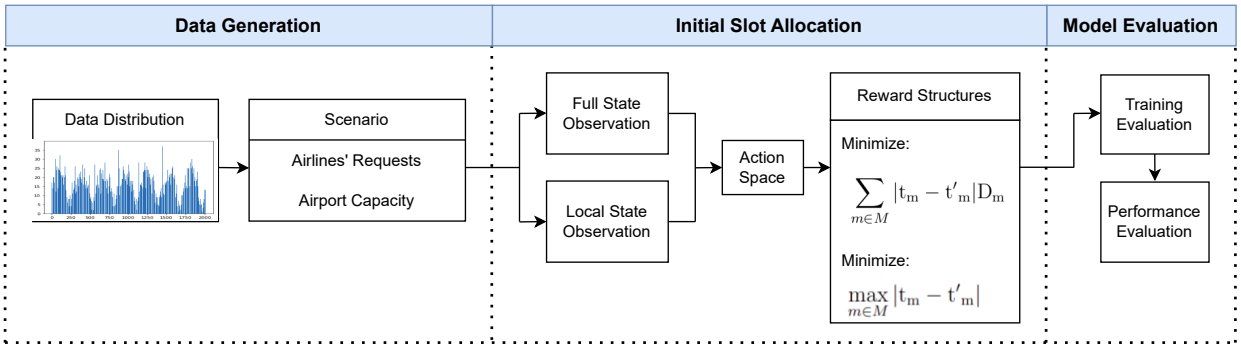


Figure 3.5: Concept diagram of component "Reinforcement Learning for Single-Airport Slot Scheduling".

Reinforcement Learning in a Multi-Airport Slot Scheduling System

Building on the foundation established in the first component, this second component focuses on analyzing how the Reinforcement Learning (RL) agent performs within a multi-airport system. While the RL model's architecture is based on the foundation developed in the first component, it requires enhancements, particularly in the information provided to the agent. In a multi-airport system, the RL agent must interact with other airport agents to adjust schedules. Therefore, this component aims to analyze an enhanced version of the state observation that integrates information from the associated arrival airport. Furthermore, the inherent nature of a multi-airport system allows each airport to follow its own decision-making policy, as long as it adheres to the slot scheduling guidelines. We want to evaluate whether the RL agent can effectively utilize these interactions between airports to learn and capture the underlying scheduling policies of other airports, ultimately leading to improved allocation decisions.

Figure 3.6 illustrates the concept diagram for the second component. In this component, we use the OAG data, which reflects historical scheduled flights. Since these flights already

satisfy airport capacity constraints, we lower the capacity to generate more requests that require adjustment. For the multi-airport level, we assume that a Reinforcement Learning (RL) agent interacts with other airports adopting specific policies. The RL agent resolves non-coherent departure movements following the proposed protocol in Figure 3.3. Once training is complete, the model undergoes a similar evaluation process as in the first component. We further assess the RL agent's performance with a baseline method, which allocates slots to the nearest available departure-arrival slot pair.

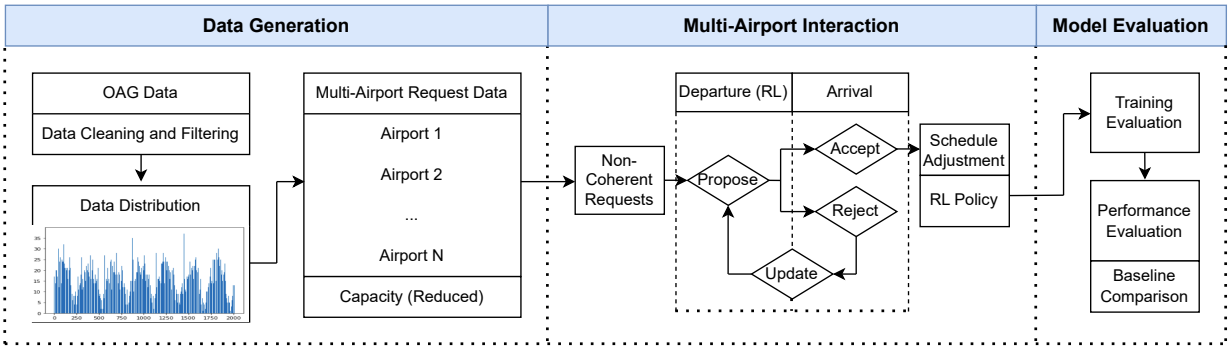


Figure 3.6: Concept diagram of component "Reinforcement Learning in a Multi-Airport Slot Scheduling System".

Incorporating Fairness into Reinforcement Learning for Airport Slot Scheduling

There are three primary approaches to incorporating fairness in airport slot scheduling: the mini-max criterion, the principle of proportionality, and the constrained-based approach. Fairness can be integrated through the first two approaches by designing reward structures. The third approach models fairness as constraints, which can be achieved by limiting the number of actions available to the agent. In this component, we analyze the effectiveness of the approaches for embedding fairness into the RL formulation.

Figure 3.7 illustrates the concept diagram for this component. Different RL formulations are trained, each representing different approaches to fairness. This results in multiple policies, which are evaluated for both training efficiency and performance. By comparing these models, we aim to assess the effectiveness of each approach in incorporating fairness. We further examine the trade-off between fairness and efficiency by comparing the obtained models with a baseline, where no fairness is considered.

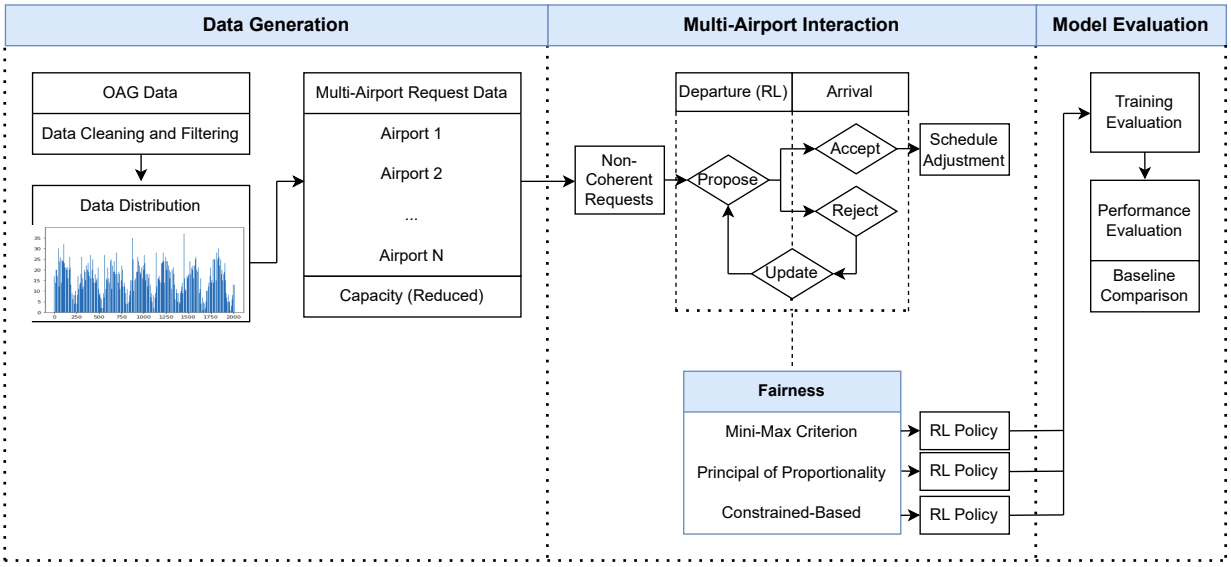


Figure 3.7: Concept diagram of component "Incorporating Fairness into Reinforcement Learning for Airport Slot Scheduling".

3.3 Summary

Figure 3.8 summarizes the research scope of this thesis. Based on the literature, we justify focusing on the practice-based airport slot scheduling approach to address the airport congestion problem. This thesis explores the application of Reinforcement Learning (RL) for airport slot scheduling, encompassing both single-airport and multi-airport perspectives. For the multi-airport context, we specifically consider applications in regions without a central authority, such as ASEAN. To ensure a practical and implementable solution, the study incorporates two key objectives: minimizing schedule delay (system efficiency) and maintaining fairness.

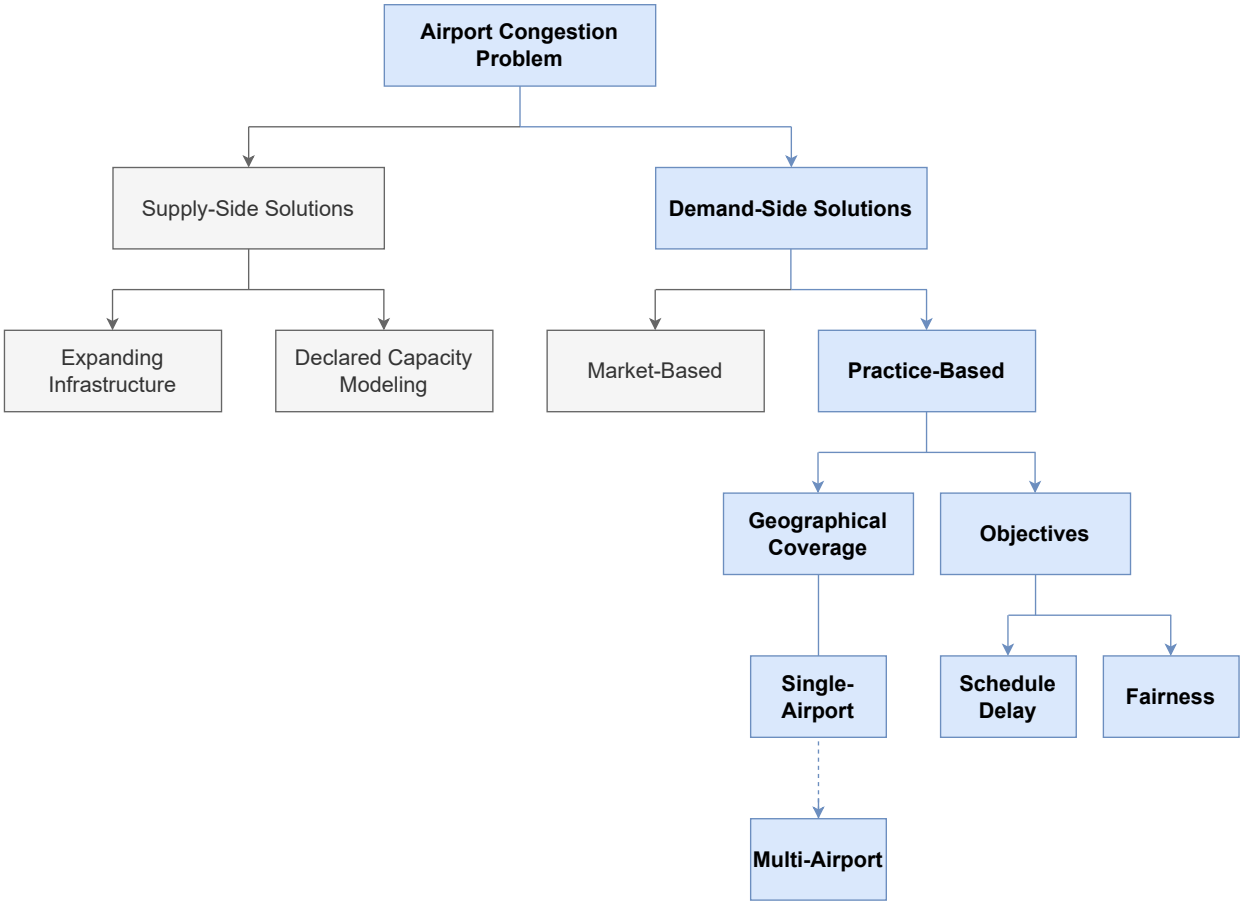


Figure 3.8: Scope of research.

Chapter 4

Reinforcement Learning for Single-Airport Slot Scheduling

Approaches for Single-Airport Slot Scheduling focus on optimizing the slot allocation process at the individual airport level. At this level, slot coordinators allocate slots based on their respective airport capacities and airline requests, without considering network-level effects. Network-Level (Multi-Airport) Slot Scheduling approaches build upon the foundation and findings of Single-Airport Slot Scheduling approaches. Unless a network-level approach can comprehensively address all objectives and constraints across all airports in the network, there remains significant research potential in Single-Airport Slot Scheduling to develop assisting tools for slot coordinators in the initial slot allocation process.

The slot scheduling problem is classified as a Resource-Constrained Scheduling Problem (RCSP). As evident from the literature, Reinforcement Learning (RL) approaches for RCSPs have shown competitive performance, making them a promising method for addressing the slot scheduling problem. Reinforcement Learning has been extensively applied in the Air Traffic Management (ATM) domain, further highlighting its potential in this context.

In this chapter, we formulate the Single-Airport Slot Scheduling problem using Reinforcement Learning. This work serves as a foundation for extending RL-based slot scheduling to various extensions. The content of this chapter is based on our published work, "Reinforcement Learning for Strategic Airport Slot Scheduling: Analysis of State Observations and

Reward Designs", presented at the IEEE Conference on Artificial Intelligence (CAI), 2024.

4.1 Overview

This chapter explores the application of Reinforcement Learning to airport slot scheduling. The problem is formulated as a Markov Decision Process (MDP), and we adopt a reward structure commonly used in pre-tactical airport slot allocation (Yutong et al., 2023; Chen et al., 2021; Tang and Xu, 2021) and build upon it with further extensions. We propose that including a positive reward signal for addressing unaccommodated requests can enhance training stability. To validate this, we train models with two reward designs: Reward Design A, which includes the positive reward signal, and Reward Design B, which does not. Additionally, we examine the impact of using full state observations versus local (partial) state observations to evaluate their effectiveness. Performance is assessed based on total schedule delay, maximum displacement of requests, and the number of unaccommodated requests. To evaluate the models' generalization to unseen scenarios, statistical measures such as t-statistic and p-value are employed.

Formulating a problem using Reinforcement Learning and evaluating the performance of the approach requires the following components:

- *Learning Environment:*

  The learning environment is a critical component with which the RL agent interacts to develop its decision-making policy. The environment must be capable of generating scenarios that represent the nature of the problem and specifying the rules governing state transitions based on the agent's actions. These rules, commonly referred to as the *Learning Mechanism*, define how the environment evolves in response to the agent's decisions.

- *Reinforcement Learning Model:*

  The Reinforcement Learning model comprises several key components. The action space represents the set of possible actions available to the agent at each step, and

its design has a significant impact on training efficiency and the agent's overall performance. State observation encapsulates information that partially or fully describes the environment at any given time, ensuring that the agent has sufficient input to make informed decisions. Reward design evaluates the favorability of the agent's actions in achieving the task's objectives, and it must be carefully structured to align with the problem's goals.

- *Experiments:*

  Based on the defined Reinforcement Learning formulations, the experiments are conducted, comprising training and testing phases. The training configurations specify the details of the generated scenarios, including traffic density, the number of slot requests, and airport capacity constraints. These configurations also outline the training settings, which consist of the RL formulations and the scenarios used to train the agent. Learning algorithms, such as DQN and PPO, define how the agent updates its policy, while hyper-parameters, including the learning rate, discount factor, exploration rate, and algorithm-specific and model-specific parameters, guide the training process. Convergence metrics are employed to determine whether the agent has achieved convergence during training, and performance metrics evaluate the agent's effectiveness during testing. After completing the experiments, a summary of the training and testing results is provided, accompanied by a discussion of the findings.

After completing the experiments, we provide a summary of the *Training* and *Testing Results*, along with the *Discussion*.

4.2 Learning Environment

4.2.1 Generating Scenarios

Figure 4.1 illustrates the overall conceptual framework of this chapter. The learning environment begins by generating a scenario that includes both the airlines' requests and the airports'

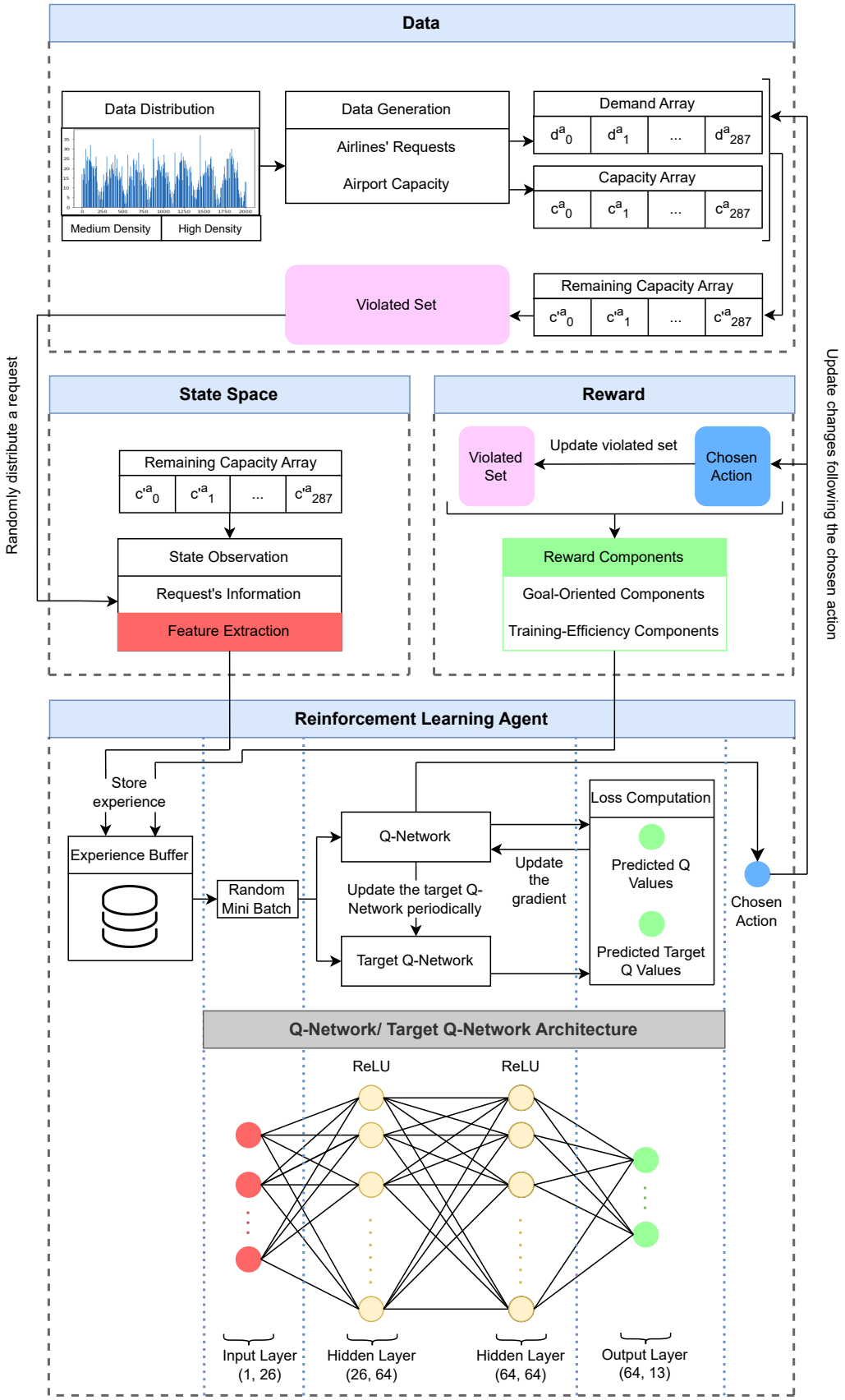


Figure 4.1: The overall concept diagram of Chapter 4.

capacity constraints. Each airline's request comprises two movements: departure and arrival. A request $r \in R$ is represented as a tuple $(a_{dep}, t_{dep}, a_{arv}, t_{arv})$, where $a_{dep}, a_{arv} \in A$ represent the departure and arrival airports, and $t_{dep}, t_{arv} \in T$ denote the corresponding time slots.

For each departure or arrival movement at an airport $a \in A$ associated with a time slot $t \in T$, the current demand $d_t^a \in D_T^A$ is incremented by one. The current demand d_t^a represents the total number of movements associated with a particular time slot t at airport a. The capacity of time slot t at airport a is denoted by c_t^a. The set of time slots where the demand exceeds capacity is represented as $T_{violated} = \{t_a | d_t^a > c_t^a\}$. If a request r includes either t_{dep} or $t_{arv} \in T_{violated}$, it is added to the violated set V.

In this chapter, we focus on scenarios involving two airports, tested under medium and high traffic density levels, to evaluate the RL agent's ability to generalize to unseen conditions. A detailed configuration of these scenarios will be provided in subsequent sections. Table 4.1 summarizes the notations used in this chapter.

Table 4.1: Summary of Notations

Notation	**Description**
$r \in R$	Set of airline requests: $r = (a_{dep}, t_{dep}, a_{arv}, t_{arv})$ (departure/arrival airports and time slots).
A	Set of airports: $A = \{0, 1\}$, where 0 and 1 represent the two airports.
T	Set of time slots: $T = \{0, 1, \ldots, 287\}$ (5-minute intervals).
C_T^A	Capacity constraints: c_t^a for time slot t at airport a.
D_T^A	Current demands: d_t^a for time slot t at airport a.
$T_{violated}$	Violated time slots: $T_{violated} = \{t \mid d_t^a > c_t^a\}$.
V	Unaccommodated requests: Requests with t_{dep} or t_{arv} in $T_{violated}$.

4.2.2 Learning Mechanism

At the start of each episode, the learning environment generates a scenario containing the airline requests and the airports' capacity. The data is encapsulated in arrays, each of size (1x288), where 288 corresponds to the number of 5-minute time slots in a day. Each airport a is associated with a capacity array $Capacity\_Array^a = [c_0^a, c_1^a, ..., c_{287}^a]$ and a current demand array $Current\_Demand\_Array^a = [d_0^a, d_1^a, ..., d_{287}^a]$. From these arrays, the remaining capacity array $Remaining\_Capacity\_Array^a = [\mathrm{c'}_0^a, \mathrm{c'}_1^a, ..., \mathrm{c'}_{287}^a]$ is derived, where $\mathrm{c'_t} = \mathrm{c_t} - \mathrm{d_t}$. The remaining capacity arrays are used to identify the current unaccommodated requests.

The process of resolving unaccommodated requests sequentially has been commonly adopted in heuristic methods (Ribeiro et al., 2019a; Wang et al., 2019a). Similarly, the learning environment selects a request from the violated set at each step and distributes it to the RL agent. The order in which the unaccommodated requests are resolved can affect the final results. Previous studies (Wang et al., 2019a) have shown that prioritizing requests with the highest number of spanning days can lead to better results. However, since the scenarios in this chapter consider a single-day scheduling horizon, we let the requests from the violated set be randomly selected at each step.

The information of the selected request is encapsulated as state observations, which are given to the RL agent as inputs for decision-making. Once the agent takes an action, the current demand arrays, remaining capacity arrays, and violated set are updated accordingly based on the action taken. The environment provides feedback in the form of a reward, enabling the agent to assess the effectiveness of its action within the given state. The reward design plays a crucial role in guiding the agent toward achieving its objectives.

One of the RL agent's primary objectives is to reassign time slots to the requests in V such that no time slots remain in $T_{violated}$. Resolving the requests in V is formulated as a minimization objective rather than enforced as a hard constraint. This approach offers two key advantages. First, enforcing the aforementioned task as a hard constraint may make the problem unsolvable. Second, formulating the task as a minimization objective allows for greater flexibility in balancing different conflicting objectives: minimizing the total schedule delay,

reducing the number of unaccommodated requests, and maintaining schedule acceptability. The approach ensures the trade-off between these objectives is appropriately managed. The maximum displacement per request is considered an important factor in maintaining schedule acceptability(Zografos et al., 2018) and can be integrated as a minimization objective.

The episode continues until all the unaccommodated requests are cleared from the violated set or the predefined maximum number of steps for the episode is reached.

In the next section, we present the action space, state observation, and reward design, each carefully formulated to reflect the nature of the problem while enhancing training efficiency and the agent's performance. Additionally, we provide a thorough explanation and justification for the formulation, emphasizing its importance in achieving convergence.

4.3 Reinforcement Learning Model

4.3.1 Action Space

Figure 4.2 demonstrates the design of the action space. One of the objectives of slot scheduling is to minimize the discrepancy between the requested and assigned time slots while complying with airport capacity constraints. To meet the objective, the assigned time slots should vary around the requested slots. Therefore, we design the agent's actions to be shifting a request to either an earlier or later time slot.

An intuitive approach might be to define the action space as all 288 time slots, allowing the agent to directly choose the specific time slot for allocation. However, this method has two significant drawbacks. First, a large action space can negatively affect the agent's training efficiency (Yehudai et al., 2022). Second, it is unrealistic to shift a request to a time slot too far from the originally requested time, such as moving a request from 6 a.m. to 6 p.m. Therefore, it is practical to limit the action space by limiting the maximum time slot displacement.

We design the agent's action space to include moving a request forward by n time slots $(+n)$, backward by n time slots $(-n)$, or keeping the request at its current time slot, as

shown in Figure 4.2. Limiting the displacement to a maximum of n time slots per step does not imply that a request cannot be shifted beyond n time slots. If a request is displaced by $k \leq n$ time slots and remains in the violated set, it may be displaced further in subsequent steps.

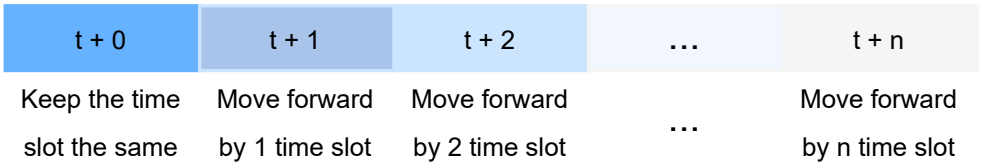


Figure 4.2: Action space of the Reinforcement Learning agent.

4.3.2 State Observation

To make decisions, the agent needs information about the current demand and the capacity of the time slots. While the capacity is assumed to be fixed across all time slots, the demand can change depending on the agent's actions. Since the capacity remains unchanged, information on demand and capacity can be encapsulated in the remaining capacity c'_t.

Although it might seem intuitive to provide the agent with the remaining capacity of all 288 time slots, redundant irrelevant information could lead to inefficient training. The agent may struggle to identify which pieces of information are most relevant to the decision-making process. For instance, the agent does not necessarily need to know the remaining capacity of time slot 200 when dealing with a request currently assigned to time slot 100. To validate our claim, we perform training experiments with two variations of the state observation:

- *Full State Observation*: An array of size 578, derived as $2(288 + 1)$, representing the current time slot (1) and remaining capacities for 288 time slots at both the departure and arrival airports.

- *Local State Observation*: An array of size $2(2n + 1)$, which consists of the remaining capacity of the n time slots preceding the current time slot, the remaining capacity of the current time slot, and the remaining capacity of the n time slots following the current time slot at the corresponding departure and arrival airports.

4.3.3 Reward Design

The reward design plays a crucial role in the training efficiency of the Reinforcement Learning agent (Dayal et al., 2022). A straightforward approach to designing the reward for the slot scheduling problem is to penalize request displacements and assign a penalty for the total number of unaccommodated requests. This aligns with the problem's objectives of minimizing the total schedule displacement and minimizing the number of unaccommodated requests. Thus, we define the first two components of the reward as follows:

- $R_{local} = z(-|t - t_m|)$, where t is the newly assigned time slot for the request, and t_m is the originally requested time slot. This component penalizes the displacement, resembling the widely adopted cost function in the slot scheduling problem. The constant z normalizes the reward to a smaller value, enhancing training efficiency. This reward is given to the agent at every time step.

- $R_{global} = -u$ or $R_{global} = v$, where u represents the total number of unaccommodated requests at the end of an episode. This component penalizes the agent based on the number of remaining unaccommodated requests. If all requests are resolved by the end of an episode, the agent receives a positive constant reward v. R_{global} is only given at the end of each episode.

To encourage the agent to solve the problem faster and prevent undesirable behavior, such as unnecessarily extending the episode, a third component is incorporated into the reward structure:

- $R_{time\_step} = -p$, where p is a constant number. This reward component is given to the agent at every time step.

In the literature, the reward design consisting of the above components is commonly used in pre-tactical airport slot reallocation problems (Yutong et al., 2023), (Chen et al., 2021), (Tang and Xu, 2021). However, a limitation of this design is that at each time step, apart from the penalty per time step, the agent either receives a reward of 0 for keeping the current

time slot or a negative reward (penalty) for shifting the request. The larger the displacement, the greater the penalty. This discourages the agent from attempting to resolve the request and instead motivates it to maintain the current time slot. Since the agent must learn to clear requests from the violated set, we introduce an additional reward component to address this issue:

- $R_{solving} = +s$, where s is a constant number. This reward component is given to the agent when the agent successfully clears the considered request from the violated set. The component encourages the agent to resolve requests in the violated set. It is important to balance between $R_{solving}$ and R_{local}. Any imbalance between the two components may lead the agent to intentionally create more unaccommodated requests and prolong the episode to gain additional $R_{solving}$. To prevent reward exploitation, $R_{solving}$ was limited to a value smaller than the total displacement cost plus the per-time-step reward. This ensures that resolving violated requests remains beneficial, while discouraging the agent from deliberately creating additional unaccommodated requests. To validate the effectiveness of $R_{solving}$ in improving training efficiency, we perform experiments with two reward designs: Reward Design A (which includes $R_{solving}$) and Reward Design B (which excludes $R_{solving}$).

Figure 4.3 illustrates the two state observation designs, Full State Observation and Local State Observation, along with the two reward designs, Reward Design A and Reward Design B. We conduct experiments using various combinations of state observation and reward design.

4.4 Experiments

4.4.1 Training Configurations

Figure 4.4 illustrates the flow diagram of the experiment process from the training phase to the testing phase. The data consists of 1000 requests, equivalent to the daily number of

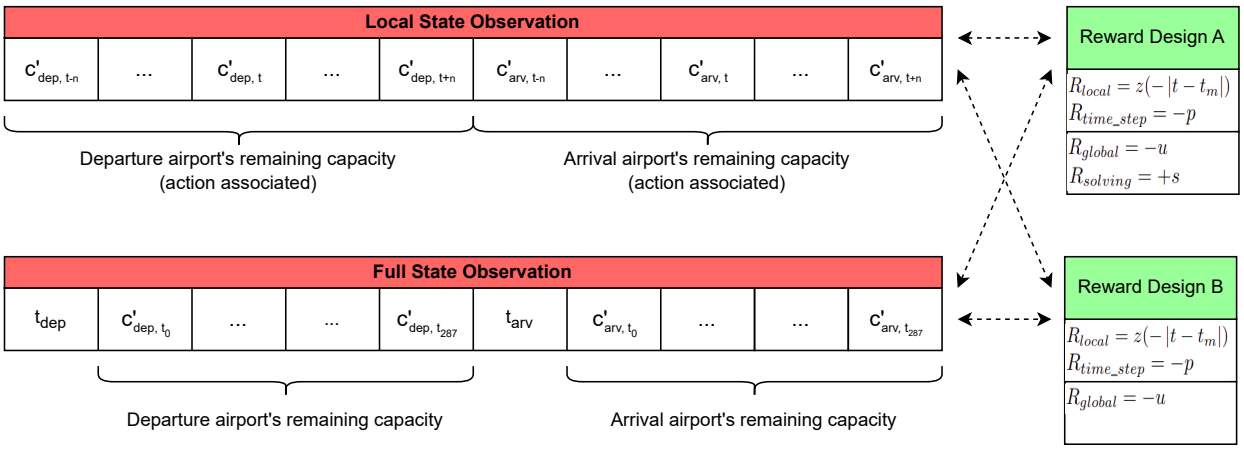


Figure 4.3: The combinations of the Full State Observation/Local State Observation and the Reward Design A/Reward Design B.

movements at Changi Airport. Since the real airline request data is unavailable during this experiment, we assume two hypothetical distributions for the requested time slots. Specifically, we consider two peak demand periods: time slot 72 (0600–0605) and time slot 216 (1800–1805). The first distribution is modeled as $X \sim \mathcal{N}(\mu = 72, \sigma)$, and the second as $Y \sim \mathcal{N}(\mu = 216, \sigma)$. To generate medium and high-density scenarios, the standard deviations are set to $\sigma = 60$ and $\sigma = 36$, respectively. The airport capacity is assumed to be fixed, with $c_t^a = 6$, $\forall a \in A$, $\forall t \in T$. Table 4.2 summarizes the data configuration for the training and testing phases.

Table 4.2: Summary of Data Configuration for Training and Testing Phases

Parameter	**Configuration**
Number of Requests	1000 requests
Request Distributions	$X \sim \mathcal{N}(\mu = 72, \sigma)$ $Y \sim \mathcal{N}(\mu = 216, \sigma)$
Scenario Density	Medium-Density: $\sigma = 60$ High-Density: $\sigma = 36$
Airport Capacity	$c_t^a = 6$, $\forall a \in A$, $\forall t \in T$

Settings 1, 3, 4, and 5 represent four combinations of Full State Observation/Local State

Observation and Reward Design A/Reward Design B, all trained with high-density scenarios. As only Setting 1 achieves convergence, we introduce an additional Setting 2, which uses the same configuration as Setting 1 but is trained with medium-density scenarios. Table 4.3 summarizes the five training settings.

Table 4.3: Summary of Experimental Settings

Setting	Reward Design	State Observation	Scenario Density (σ)
1	A	Local	High ($\sigma = 36$)
2	A	Local	Medium ($\sigma = 60$)
3	B	Local	High ($\sigma = 36$)
4	A	Full	High ($\sigma = 36$)
5	B	Full	High ($\sigma = 36$)

Justification of data configuration and assumptions

Figure 4.5, extracted from Table D4 of the ICAO Asia/Pacific Seamless Air Navigation Services Plan, Version 3, presents a summary of runway and airspace capacity expectations reported by various States, including Singapore. In our study, an airport capacity of 6 movements per 5-minute slot is adopted for Singapore Changi Airport. This assumption aligns with the APAC Seamless ANS Plan (Version 3), which specifies a capacity of 72 aircraft movements per hour for Changi's two-runway configuration (ICAO, 2022). This value corresponds to 36 movements per hour per runway, or 1 movement every 100 seconds per runway. Accordingly, each runway can accommodate 3 movements within a 5-minute interval, resulting in a combined total of 6 movements per 5-minute slot for the dual-runway system.

At the time this study was conducted, Singapore Changi Airport had three runways in its infrastructure, but only two were in active use for commercial operations. The central runway (02C/20C) was closed during this period due to ongoing construction works related to the Changi East and Terminal 5 development, as indicated by official closure notices (CAAS, 2021). Commercial flight operations were limited to the northern (02L/20R) and southern (02R/20L) runways. Therefore, it is credible to apply the two-runway operational capacity

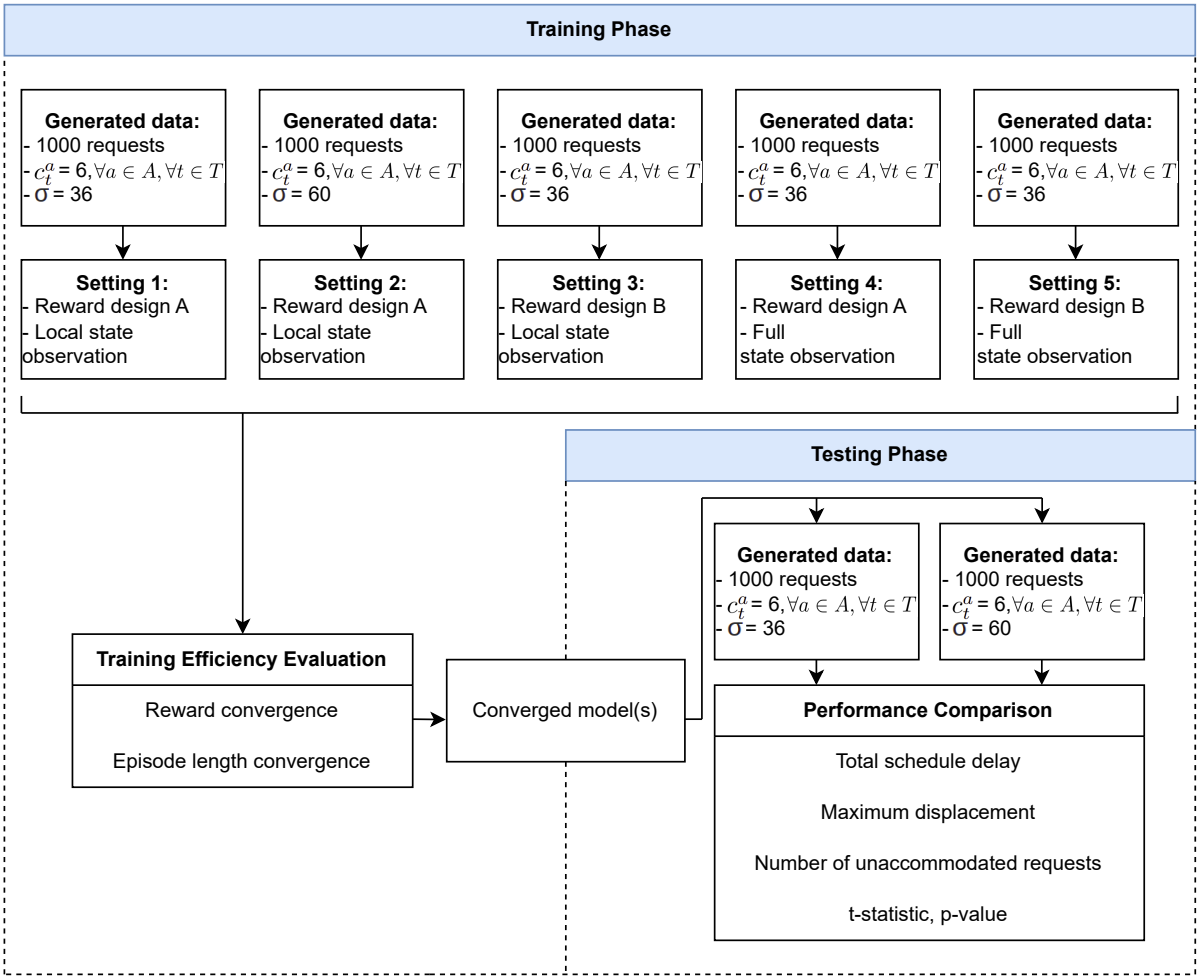


Figure 4.4: The training phase and the testing phase flow diagram.

figure in this study.

	Parallel or Near Parallel Runway Capacity					ATC Sector Capacity		
	1	2	3	4	5	T	S	R
Australia	40-48							
Japan		56-64		74				
NZ	32-40					12	15	15
Singapore	30	72				14	18	
Thailand	34							
USA	**61**	**95**	**150**	**177**	**211**	12-**18**	16-**20**	17-**24**
Doc 9971 Simplified Table Comparison						15	18	18

Figure 4.5: Summary of runway and airspace capacity expectations (Extracted from ICAO APAC Seamless ANS Plan v3, Table D4).

Another study utilizing data from Shanghai Pudong International Airport (PVG) also assumes a comparable operational scale and runway capacity (Wang et al., 2024). In the study, the number of daily aircraft movements in the dataset ranges from 1,049 to 1,068, which is comparable with the approximately 1,000 daily movements at Singapore Changi Airport during the period when the experiments in this research were conducted. The authors assume a declared capacity of 78 movements per hour, equivalent to 6 to 7 movements per 5-minute slot. These figures demonstrate a close alignment in both operational scale and airport capacity between Shanghai Pudong International Airport and Singapore Changi Airport.

In practice, the declared capacity of an airport is typically set at 85–90% of its maximum throughput. Although declared capacity is often represented as a constant number of slots per time unit, uniformly applied across all time slots, it may also be defined as time-varying in more advanced scheduling frameworks (Zografos et al., 2017). In this study, a fixed capacity is adopted across all time slots. This assumption allows the study to focus on evaluating the learning capability of the Reinforcement Learning agent in allocating slots under a consistent capacity constraint. Moreover, since the study focuses on the strategic and pre-tactical phase, where short-term capacity variations are less critical, the fixed-capacity assumption is feasible.

Figure 4.6 illustrates the relationship between declared capacity and maximum throughput, demonstrating that airport capacity can be applied uniformly across all time slots (Zo-

grafos et al., 2017). Figure 4.7 presents the case of Shanghai Pudong International Airport (PVG), where the declared capacity is set at 78 aircraft movements per hour throughout the day, further supporting the feasibility of assuming a fixed capacity across all time slots (Wang et al., 2024).

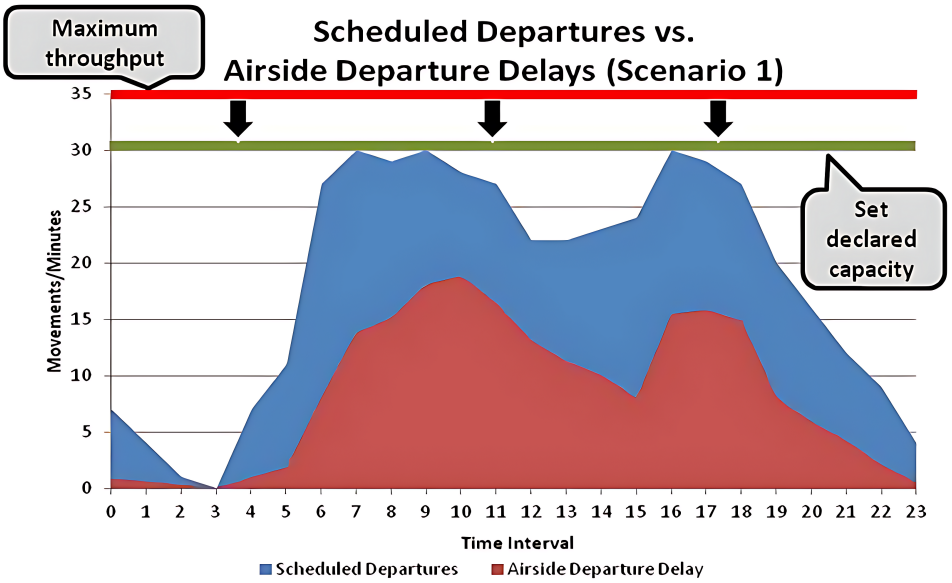


Figure 4.6: Setting of declared capacity based on maximum throughput (Extracted from Zografos et al. (2017)).

Table 4.4 presents the annual aircraft movements and corresponding average daily movements at Changi airport from 2016 to 2019. The figures consistently range around 1000 daily movements, supporting the validity of our assumption of 1000 movements per day in this study. The data before the COVID-19 period are used to represent Changi's normal operational conditions.

Table 4.4: Annual aircraft movements and average daily movements at Changi airport (2016–2019).

Year	Number of Movements	Average Daily Movements
2016	343,776	942
2017	365,305	1,001
2018	371,467	1,018
2019	351,164	962

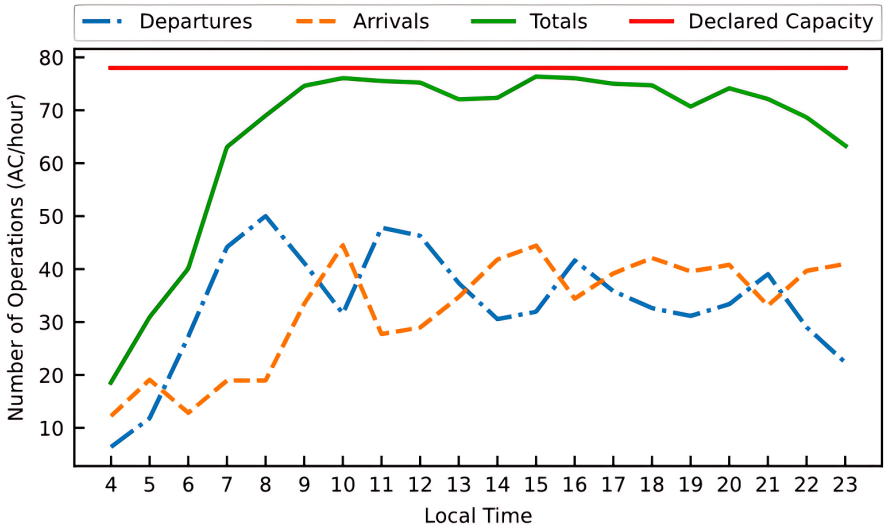


Figure 4.7: Declared capacity and average hourly demand of PVG (Extracted from Wang et al. (2024)).

4.4.2 Learning Algorithms and Hyper-Parameters

We adopt the Deep Q-Network (DQN) algorithm to derive decision-making policies across all five settings. The algorithm is particularly suitable for discrete-action problems. To enhance data efficiency and ensure training stability, we utilize two techniques: experience replay and target networks (Mnih et al., 2015). The algorithm's pseudocode is provided in Algorithm 1.

Table 4.5 summarizes the hyper-parameters used for training the DQN. The number of actions n is defined as 6, corresponding to a maximum displacement of 30 minutes per time step. The penalty p in R_{time_step} is chosen as 0.8 to maintain a balance with the average reward obtained after resolving a request in each step, which is the sum of R_{local} and $R_{solving}$. Balancing these reward components is essential to prevent the agent from exploiting the system by generating additional unaccommodated requests to increase its overall rewards.

Since the DQN algorithm successfully converges in Setting 1 and Setting 2, we further train the Proximal Policy Optimization (PPO) algorithm in these two settings to compare its performance with the DQN. Although the PPO algorithm is primarily designed for problems with continuous action spaces, continuous actions can be discretized into bins, making the PPO applicable to our problem. To improve data efficiency and maintain training stability

Algorithm 1 DQN with Experience Replay and Target Network

Setting Configuration:
$dst \leftarrow medium$ or $high$
$obs\_type \leftarrow local$ or $global$
$rw\_design \leftarrow A$ or B
Initialization:
Randomly initialize the Q-Network parameters θ.
Initialize the Target Q-Network parameters θ_{target} by $\theta_{\text{target}} \leftarrow \theta$.
Initialize the replay buffer D.
Training:
while $T \leq total\_training\_timesteps$ **do**
    if $episode\_end =$ True **then**
        Environment generates a new scenario S_{dst}.
        Receive observation $o_1^{obs\_type}$ from a randomly selected request in the violated set $V_{S_{dst}}$.
    end if
    for $t = 1$ to $max\_timesteps$ **do**
        Select action a_t based on current policy $\mu(o_t^{obs\_type}|\theta)$ and exploration value ϵ.
        Execute action a_t, receive reward $r_t^{rw\_design}$, and observe $o_{t+1}^{obs\_type}$.
        Store experience $(o_t^{obs\_type}, a_t, r_t^{rw\_design}, o_{t+1}^{obs\_type})$ in replay buffer D.
        if $V_{S_{dst}} = \emptyset$ **or** $t = max\_timesteps$ **then**
            $episode\_end \leftarrow$ True
            break
        end if
        if $T \bmod K = 0$ **then**
            Parameters Updating:
            Randomly sample a minibatch from D.
            Update θ by minimizing $L(Q_\theta, Q_{\theta_{\text{target}}})$.
            Periodically update θ_{target} by $\theta_{\text{target}} \leftarrow \theta$.
        end if
    end for
end while

Table 4.5: DQN Training Hyper-Parameters

Parameter	Value	Description
Buffer Capacity	50,000	Maximum size of the replay buffer.
Batch Size	32	Number of samples per training update.
Discount Rate (γ)	0.99	Discount factor for future rewards.
Learning Rate	0.0001	Step size for updating network parameters.
Target Update Cycle	10,000	Frequency of updating the target network.
Training Steps	10,000,000	Total number of training steps.
Number of Actions (n)	6	Maximum displacement of 30 minutes per time step.
z in R_{local}	0.1	Scaling factor for local reward.
v in R_{global}	30	Positive reward for clearing all requests at the end of an episode.
p in $R_{\text{time\_step}}$	0.8	Penalty per time step to balance rewards.
s in R_{solving}	1	Positive reward for clearing a request at the end of a time step.

for the PPO, we adopt two techniques: advantage estimation and clipped surrogate objective (Schulman et al., 2017). The algorithm's pseudocode is provided in Algorithm 2.

Table 4.6 summarizes the hyper-parameters used for training the PPO. Besides the algorithm-specific hyper-parameters, the number of actions and the reward components are the same as in the DQN.

Table 4.6: PPO Training Hyper-Parameters

Parameter	Value	Description
Batch Size	64	Number of samples per training update.
Discount Rate (γ)	0.99	Discount factor for future rewards.
Learning Rate	0.0003	Step size for updating network parameters.
Number of Epochs	10	Number of passes over the sampled mini-batch per update.
Training Steps	10,000,000	Total number of training steps.
Number of Actions (n)	6	Maximum displacement of 30 minutes per time step.
z in R_{local}	0.1	Scaling factor for local reward.
v in R_{global}	30	Positive reward for clearing all requests at the end of an episode.
p in $R_{\text{time\_step}}$	0.8	Penalty per time step to balance rewards.
s in R_{solving}	1	Positive reward for clearing a request at the end of a time step.

We refer to the models DQN (Setting 1), DQN (Setting 2), PPO (Setting 1), and PPO

Algorithm 2 PPO with Advantage Estimation and Clipped Surrogate Objective

Setting Configuration:
$dst \leftarrow medium$ or $high$
$obs\_type \leftarrow local$ or $global$
$rw\_design \leftarrow A$ or B
Initialization:
Initialize policy network parameters θ.
Initialize value network parameters ϕ.
Initialize the replay buffer D.
Training:
while $T \leq total\_training\_timesteps$ **do**
  if $episode\_end =$ True **then**
    Environment generates a new scenario S_{dst}.
    Receive observation $o_1^{obs\_type}$ from a randomly selected request in the violated set $V_{S_{dst}}$.
  end if
  for $t = 1$ to $max\_timesteps$ **do**
    Select action $a_t \sim \pi_\theta(a_t|o_t^{obs\_type})$ based on the current policy.
    Execute action a_t, receive reward $r_t^{rw\_design}$, and observe $o_{t+1}^{obs\_type}$.
    Store experience $(o_t^{obs\_type}, a_t, r_t^{rw\_design}, o_{t+1}^{obs\_type})$ in replay buffer D.
    if $V_{S_{dst}} = \emptyset$ **or** $t = max\_timesteps$ **then**
      $episode\_end \leftarrow$ True
      break
    end if
    if $T \bmod K = 0$ **then**
      Policy and Value Update:
      Randomly sample a minibatch of experiences from D.
      Compute advantage estimates A_t using the value network V_ϕ.
      Update policy parameters θ by optimizing the clipped surrogate objective:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) A_t \right) \right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$.
      Update value network V_ϕ by minimizing the value loss:

$$L^{\text{value}}(\phi) = \mathbb{E}_t \left[(V_\phi(o_t) - R_t)^2 \right].$$

      Periodically update old policy parameters: $\theta_{\text{old}} \leftarrow \theta$.
    end if
  end for
end while

(Setting 2) as Model 1, Model 2, Model 3, and Model 4, respectively.

4.4.3 Convergence and Performance Metrics

After the training phase is complete, we evaluate the training efficiency by examining the convergence of both the reward and the episode length. While achieving a high reward is the agent's objective, it does not necessarily ensure alignment with the problem's goals. The agent may develop undesired behaviors, such as prolonging the episode by creating additional unaccommodated requests to gain higher rewards if balancing between the reward components is not well-managed. We summarize the two metrics used for convergence analysis as follows:

- *Average Episode Reward (AER)* $= \frac{\sum \text{Episode Rewards}}{\text{Number of Episodes}}$
- *Average Episode Length (AEL)* $= \frac{\sum \text{Episode Lengths}}{\text{Number of Episodes}}$

During the testing phase, we generate data for each run using either medium or high-density scenarios to evaluate the agent's ability to generalize to unseen scenarios. We compare the performance using the following metrics:

- *Total schedule delay* $M_1 = \sum_{m \in M} |t - t_m|$. This is the most commonly used performance metric for the airport slot scheduling problem.
- *Maximum displacement across all requests* $M_2 = \max_{m \in M} |t - t_m|$. This metric evaluates the feasibility of the final generated schedule, as airlines are unlikely to accept significant time displacement gaps.
- *Number of unaccommodated requests* $M_3 = u$. An important goal of the slot scheduling problem is to maximize the number of accommodated requests. Hence, it is essential to include this metric in the performance comparison.
- *Average displacement per request* $M_4 = M_1/(U - M_3)$, where U is the number of unaccommodated requests at the beginning of each episode. This metric provides more insight into the displacement distribution.

- The t-statistic and p-value are used to test whether the results obtained from Model 1 and Model 2 differ significantly. If the p-value is not statistically significant, it indicates that the performance of the two models is comparable, suggesting that both models generalize well to unseen scenarios. Conversely, if the p-value is significant, it implies a notable difference between the models' results. The same procedure is applied to compare the performance of Model 3 and Model 4.

4.5 Results and Discussion

4.5.1 Training Results

Figure 4.8 presents the convergence analysis graphs for the 5 settings. We observe that the DQN algorithm achieves stable convergence with both Setting 1 and Setting 2. This is evident from the increasing trend in the average episode reward and the decreasing trend in the average episode length.

- *Model 1 (DQN with Setting 1)* achieves a reward of approximately 1.468 after convergence. This value is reasonable since:

  The reward per time step when solving a request is calculated as:

$$R_{\text{local}} + R_{\text{solving}} + R_{\text{time\_step}} = 0.1\,(-|t - t_m|) + 1 - 0.8, \tag{4.1}$$

  where t_m is the original time slot.

  Substituting the range for $|t - t_m| \in [1, 6]$, the possible reward per time step when solving a request belongs to the range:

$$\big[0.1(-6) + 1 - 0.8, 0.1(-1) + 1 - 0.8\big] = \big[-0.4, 0.1\big]. \tag{4.2}$$

  Given an average of 369.65 unaccommodated requests at the beginning of each episode and the reward for clearing all requests at the end of an episode (R_{global}) of 30, the

total reward can range between:

$$[(-0.4)369.65 + 30, (0.1)369.65 + 30] = [-117.86, 66.965]. \tag{4.3}$$

- *Model 2 (DQN with Setting 2)* achieves a reward of approximately 36.151 after convergence, which is also reasonable since:

  The average number of unaccommodated requests at the beginning of each episode is 249.80. Using the same reward calculations, the total reward range is:

$$\big[(-0.4)249.80 + 30, (0.1)249.80 + 30\big] = \big[-69.920, 54.980\big]. \tag{4.4}$$

For comparison, the PPO algorithm achieves similar convergence trends in both settings but with slightly lower results:

- *Model 3 (PPO with Setting 1)* achieves a reward of approximately -10.869 after convergence.
- *Model 4 (PPO with Setting 2)* achieves a reward of approximately 27.051 after convergence.

For each episode, the average number of unaccommodated requests is 249.80 for medium-density scenarios and 369.65 for high-density scenarios. Model 1 and Model 3 achieve average episode lengths of 399.81 and 412.89, respectively, which are comparable to the 369.65 figure of high-density scenarios. Similarly, Model 2 and Model 4 achieve average episode lengths of 256.72 and 262.51, respectively, aligning closely with the 249.80 figure of medium-density scenarios. These results indicate that the agent does not need to frequently revisit the unaccommodated requests during the episode.

Table 4.7 summarizes the training results for comprehensive reference and comparison.

Table 4.7: Summary of Training Results for Models 1, 2, 3, and 4.

Algo.	**Setting**	**Avg. Episode Reward**	**Avg. Episode Length**	**Reward Range**	**Unaccom. Requests**
DQN	1	1.468	399.81	$[-117.86, 66.965]$	369.65
DQN	2	36.151	256.72	$[-69.920, 54.980]$	249.80
PPO	1	-10.869	412.89	$[-117.86, 66.965]$	369.65
PPO	2	27.051	262.51	$[-69.920, 54.980]$	249.80

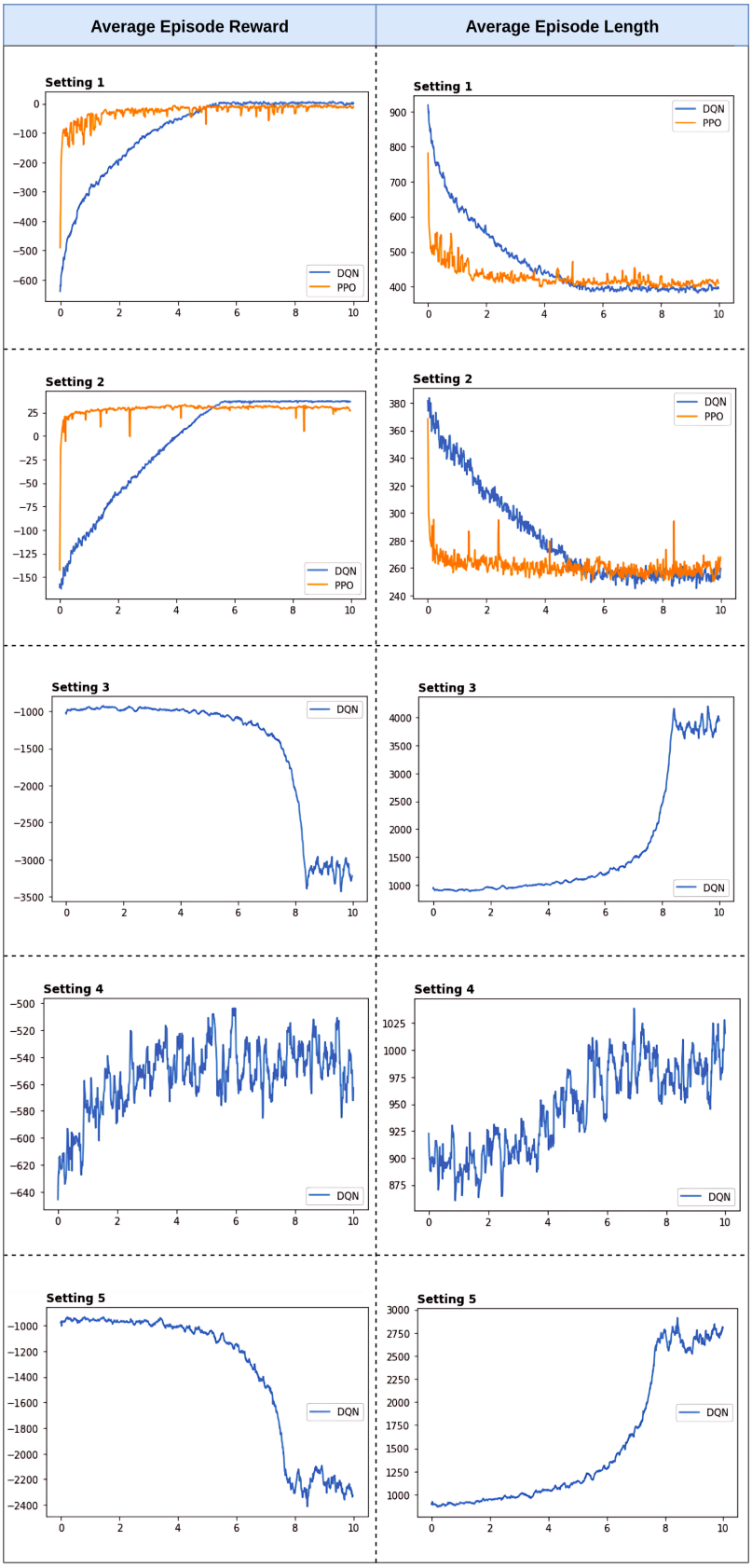


Figure 4.8: Convergence Analysis Graphs for Settings 1, 2, 3, 4, and 5.

4.5.2 Testing Results

Table 4.8 and Table 4.9 present the testing results of the four models on 50 medium-density and 50 high-density scenarios, respectively. M_3 is measured in *requests*, whereas all other metrics are measured in *time slots*. The scenarios are the same across all four models. During testing, there are scenarios where the models fail to resolve all unaccommodated requests, which can result in a misleading lower total schedule delay. While a lower total schedule delay is generally a positive performance indicator, achieving it with unaccommodated requests remaining is undesirable.

For an unbiased evaluation, we introduce a new metric, adjusted total schedule delay:

$$M_1^* = M_1 + M_3(M_2 + 4\sigma(M_2)). \tag{4.5}$$

While M_1 measures the total schedule delay, it does not account for unaccommodated requests. When certain requests remain unaccommodated, no delay is recorded for them, which can lead to an artificially lower M_1 despite an undesirable outcome. To address this limitation, M_1^* combines both the total schedule delay M_1 and the number of unaccommodated requests M_3 into a single unified metric, providing a more comprehensive evaluation of scheduling performance. $M_2 + 4\sigma(M_2)$ represents the furthest displacement based on the obtained statistics, correcting the total schedule delay when unaccommodated requests remain.

From the testing results, we provide highlights of the performance analysis:

- *Medium-Density Scenarios*:

  Model 2 demonstrates the best overall performance, achieving an average total schedule delay of 355.18 time slots while having no unaccommodated requests.

- *High-Density Scenarios*:

  Although Model 4 achieves the lowest average total schedule delay (729.58 time slots) and adjusted total schedule delay (746.67 time slots), it has a significantly high average number of unaccommodated requests compared to the other three models.

Among the remaining models, Model 1 performs best with the following results: average total schedule delay of 750.82 time slots (adjusted: 751.11 time slots), average maximum displacement of 7.28 time slots, average number of unaccommodated requests of 0.02, and average displacement per request of 1.99 time slots.

Overall, the DQN models outperform the PPO models in both medium-density and high-density scenarios. The reason is primarily that the slot scheduling problem involves discrete actions and deterministic transitions, which align better with DQN's structure. PPO's updates are more effective in continuous or stochastic environments. However, further validation using real-world data and additional model tuning is required to confirm the generalizability of this finding.

We compute the t-statistic and p-value for the total schedule delay across 50 runs for each scenario type to compare Model 1 and Model 2:

- *Medium-Density Scenarios*: t-statistic: 0.0810, p-value: 0.9356.
- *High-Density Scenarios*: t-statistic: -1.0016, p-value: 0.3190.

These results show no statistically significant difference between the two models' performance, as the p-values are much larger than the standard threshold ($p > 0.05$). This suggests that Model 1 and Model 2, based on the DQN algorithm, demonstrate consistent and comparable performance, indicating fair generalization ability across the two scenario types.

In contrast, when comparing Model 3 and Model 4:

- *Medium-Density Scenarios*: t-statistic: 0.0854, p-value: 0.9321.
- *High-Density Scenarios*: t-statistic: 3.4625, p-value: 0.0008.

For medium-density scenarios, no significant difference is observed ($p > 0.05$), similar to the results for Models 1 and 2. However, for high-density scenarios, the p-value of 0.0008 (significantly below 0.05) and the large positive t-statistic of 3.4625 indicate a statistically

significant difference in performance. This suggests that Model 3 and Model 4 fail to generalize effectively under high-density conditions, highlighting performance inconsistencies for these models in more demanding scenarios.

Table 4.8: Testing Results (Medium-Density)

	DQN (Setting 1)	**DQN (Setting 2)**	**PPO (Setting 1)**	**PPO (Setting 2)**
U	246.26 (25.99)			
M_1	355.84 (40.90)	355.18 (39.79)	392.72 (44.00)	391.94 (46.42)
M_1^*	355.99	355.18	392.72	391.94
M_2	4.30 (0.75)	4.36 (0.69)	5.44 (0.75)	6.18 (0.65)
M_3	0.02 (0.14)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
M_4	1.45 (0.08)	1.44 (0.07)	1.60 (0.07)	1.59 (0.08)

Each value cell indicates the mean and the standard deviation (in parentheses) of the corresponding metric.

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

M_3 is measured in *requests*, whereas all other metrics are measured in *time slots*.

Table 4.9: Testing Results (High-Density)

	DQN (Setting 1)	**DQN (Setting 2)**	**PPO (Setting 1)**	**PPO (Setting 2)**
U	377.32 (33.41)			
M_1	750.82 (81.17)	768.04 (88.85)	785.30 (88.59)	729.58 (76.61)
M_1^*	751.11	769.53	786.60	746.67
M_2	7.28 (1.78)	8.46 (2.53)	9.76 (1.63)	10.22 (2.53)
M_3	0.02 (0.14)	0.08 (0.34)	0.08 (0.39)	0.84 (0.99)
M_4	1.99 (0.10)	2.03 (0.14)	2.08 (0.13)	1.93 (0.11)

Each value cell indicates the mean and the standard deviation (in parentheses) of the corresponding metric.

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

M_3 is measured in *requests*, whereas all other metrics are measured in *time slots*.

4.6 Summary

In this chapter, we present a Reinforcement Learning formulation for addressing the airport slot scheduling problem. The training process investigates different combinations of state observations and reward designs. The results indicate that employing local state observations along with a positive reward signal leads to stable convergence. The trained DQN models demonstrate fair generalization on unseen scenarios and outperform PPO models overall.

This chapter establishes a foundational framework for applying Reinforcement Learning to the slot scheduling problem. Building upon this foundation, subsequent chapters will evaluate the performance of RL agents in a multi-airport system and incorporate fairness considerations. While real-world data was not utilized for the training and testing phases in this chapter, future experiments will leverage OAG data to enhance the applicability and robustness of the approach.

Chapter 5

Reinforcement Learning in a Multi-Airport Slot Scheduling System

The previous chapter establishes a foundation for formulating the slot scheduling problem using Reinforcement Learning. However, a key aspect of slot scheduling is the interdependence between airports, where the decisions of one airport must be coherent with others within the network.

Existing approaches in the literature assume a centralized framework, where a global solution is generated for all airports in the system. This assumption is impractical in regions like ASEAN, where no central authority governs the system. Instead, airports in different nations retain autonomy over their resources, provided they comply with the IATA Worldwide Slot Guidelines (WSGs) and local regulations.

This chapter explores the application of Reinforcement Learning to slot scheduling in a multi-airport system. We focus on formulating and analyzing the performance of a Reinforcement Learning agent acting as a node in an airport hub. This study offers insights into using Reinforcement Learning to address slot scheduling in a multi-airport system. The content of this chapter is based on our published work, "Reinforcement Learning for Collaborative Multi-Airport Slot Re-Allocation Under Reduced Capacity Scenarios", presented at the International Workshop on ATM/CNS (IWAC), 2024.

5.1 Overview

Airport Collaborative Decision Making (A-CDM) is being implemented to enable stakeholders within an interconnected multi-airport system to collaborate effectively and reach joint agreements while optimizing their diverse objectives (Corrigan et al., 2015). However, in regions like ASEAN, where no central authority governs operations, countries maintain autonomy over managing resources such as airport slot utilization (Duong et al., 2019).

The IATA Worldwide Airport Slot Guidelines (WASG) states the need to "ensure the most efficient declaration, allocation, and use of available airport capacity in order to optimize benefits to consumers, taking into account the interests of airports and airlines, while minimizing congestion and delays" (IATA, 2024). From these guidelines, various objectives can be inferred. Stakeholders under different authorities prioritize these objectives differently, based on their preferences and local guidelines. This diversity leads to various decision-making policies across airports (Katsigiannis and Zografos, 2024). Given the complexity of a multi-airport system, it is infeasible to develop a universal model capable of fully accommodating the preferences of all stakeholders (Katsigiannis and Zografos, 2024). Consequently, the decision-making policies and the selection of optimization algorithms depend on the individual preferences of each airport.

Studying the application of Reinforcement Learning (RL) in a multi-airport system can be categorized into three main approaches:

- *Centralized RL*: A centralized decision-making framework where Reinforcement Learning is used to manage the entire airport network.
- *Decentralized RL (Independent)*: A Reinforcement Learning agent operates independently within a multi-airport system.
- *Decentralized RL (Collaborative)*: Airports within a network adopt RL either homogeneously (using the same RL policies) or heterogeneously (using different RL policies) to collaborate.

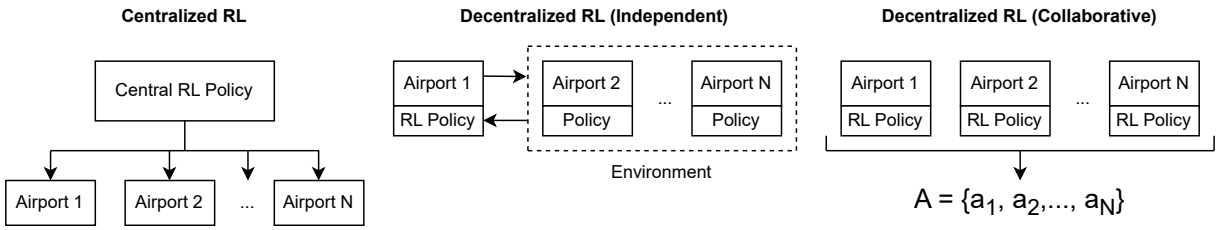


Figure 5.1: Agent Interaction Summary.

Figure 5.1 illustrates the interactions between the airport agents in a network under the three approaches: Centralized RL, Decentralized RL (Independent), and Decentralized RL (Collaborative). In the Centralized RL approach, a central system collects information from the airports and generates a network-wide solution. However, this approach is only applicable for regions with a central authority, such as EUROCONTROL in Europe or the FAA in the United States. In the Decentralized RL (Collaborative Network) approach, agents are required to adopt a collaborative RL framework to make joint decisions. This approach assumes that all airports in the network agree to adopt RL, which is an unlikely scenario in practice due to differing preferences and autonomy requirements.

Thus, this chapter focuses on the second approach, Decentralized RL (Independent Agent), where individual airports independently employ Reinforcement Learning as their decision-making mechanism, treating the remaining airports in the network, along with their respective policies, as part of the environment. Unlike the third approach, which assumes that all airports in the network adopt RL, the second approach provides flexibility, enabling each airport to independently form its policies while maintaining autonomy in decision-making.

This chapter aims to develop a Reinforcement Learning formulation specifically designed for the slot scheduling task in a multi-airport system. The rest of the chapter is organized as follows:

- *Learning Environment:*

  The learning environment introduced in Chapter 4 is extended to a multi-airport context. As a proof-of-concept, we focus on the Hong Kong-Singapore-Bangkok hub, enabling the exploration of interactions between airports within this network. In this chapter, we employ OAG data to simulate realistic scenarios. This data ensures that

the experiments closely reflect real-world conditions in airport slot scheduling.

- *Reinforcement Learning Model:*

  Building upon the Reinforcement Learning model developed in Chapter 4, we introduce several extensions and modifications to adapt the RL framework to the multi-airport context. One key adjustment involves refining the state representation to capture inter-airport interactions effectively. The adjustments aim to provide the RL agent with sufficient information to make more informed and effective decisions in a multi-airport context.

- *Experiments:*

  We outline the training configurations, learning algorithms, and hyper-parameters. Convergence analysis is performed, and test performance is compared with the Nearest Heuristics, which assigns the nearest available departure-arrival slot pair.

After completing the experiments, we provide a summary of the *Training* and *Testing Results*, along with the *Discussion*.

5.2 Learning Environment

5.2.1 Problem Overview

Current network-level slot allocation approaches assume a centralized optimization framework, in which a single decision-making authority aggregates flight requests and capacity data from all airports and solves for a global solution simultaneously (Zografos et al., 2017). While such centralized coordination provides theoretical optimality, it is not operationally feasible in regions where airports maintain decision autonomy and no central scheduling authority exists, such as within the ASEAN network (Duong et al., 2019). Furthermore, operational uncertainties (e.g., weather disruptions or temporary runway closures) frequently cause capacity reductions, requiring rapid schedule adjustments. Re-optimizing the entire

network under such conditions would require system-wide recomputation, leading to drastic schedule changes that undermine pre-established planning schedules and reduce operational stability.

Consequently, localized coordination among small groups of connected airports becomes essential. The airport congestion management must consider interactions among interconnected airports to achieve system-wide efficiency improvements. However, the scope and scale of such "connected" clusters, specifically, how many and which airports should be considered during local reallocation, remain largely unexplored in the literature (Jacquillat and Odoni, 2015).

In practice, airports and airlines engage in bilateral or multilateral coordination, particularly under capacity constraints, slot trading, or shared airspace dependencies (EUROCONTROL, 2023; Kargar et al., 2024). There is no established guidance on how local coordination structures should be defined or optimized under operational uncertainty. To address this gap, we adopt a two-airport formulation as an initial step toward understanding decentralized coordination mechanisms. This setup serves as a proof-of-concept for localized slot reallocation under capacity-reduction scenarios. We examine how decentralized agents adapt to disruptions before extending the framework to more complex multi-airport coordination environments.

To demonstrate the concept, a proof-of-concept case study is conducted using the Hong Kong–Singapore–Bangkok hub, designed to evaluate how a Reinforcement Learning (RL) agent performs within a multi-airport network. In this setup, the RL agent represents Changi Airport (Singapore), interacting with other airports, such as Hong Kong and Bangkok, which operate with fixed policies. The Hong Kong and Bangkok airports are modeled with fixed policies, meaning their decision rules remain constant throughout the simulation and do not adapt through learning. These airports follow a deterministic slot reallocation policy, where any flight affected by capacity reduction is reassigned to the nearest available time slot that does not violate capacity constraints. This setup allows the focus to be placed on evaluating the RL agent at Singapore (Changi Airport), which actively learns and adapts its policy in

response to the policies of other airports. The objective is to assess whether the RL agent can effectively leverage inter-airport interactions to infer the decision-making behaviors of other airports and develop an efficient slot reallocation strategy under reduced-capacity conditions.

The study considers scenarios in which a runway closure at Changi Airport reduces its capacity, creating an imbalance between flight demand and available time slots. Acting as the slot coordinator, the RL agent at Changi utilizes information from other airports to reallocate slots efficiently, while Hong Kong and Bangkok airports maintain fixed policies allowing shifting flights to the next available time slot when necessary.

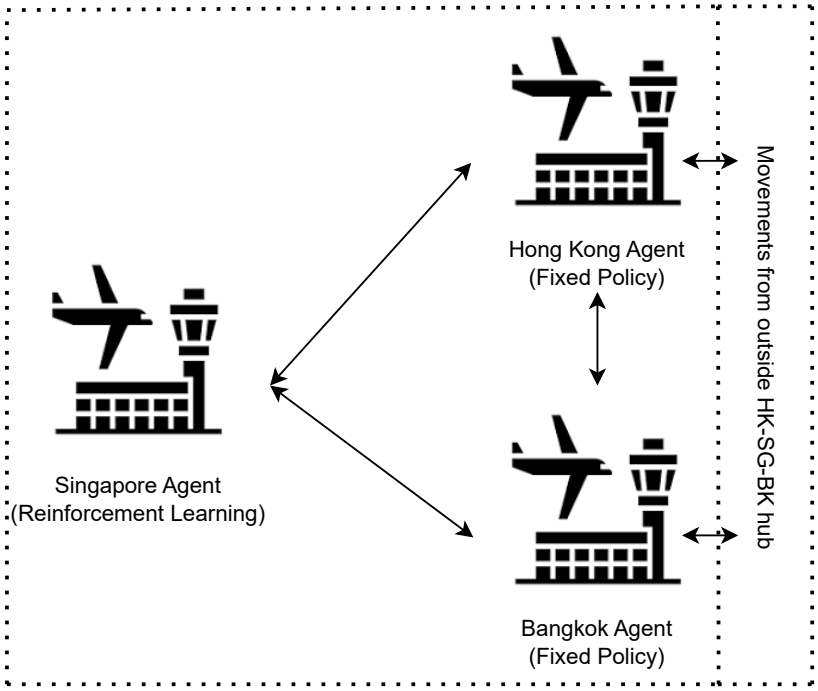


Figure 5.2: Interactions between the airport agents in the Hong Kong-Singapore-Bangkok hub.

Figure 5.2 illustrates the interactions between airports in this system. While the RL agent can assign flights to the nearest available time slot pairs without violating capacity constraints, effective use of information from other airports can further minimize delays (displacements), as shown in Figure 5.3. To achieve this, the RL agent must learn a policy that can capture the other airports' policies via system-wide interaction across the hub.

For flights departing from or arriving outside the hub, external airports will accept movement adjustments as long as capacity constraints are not violated. This simplifies modeling

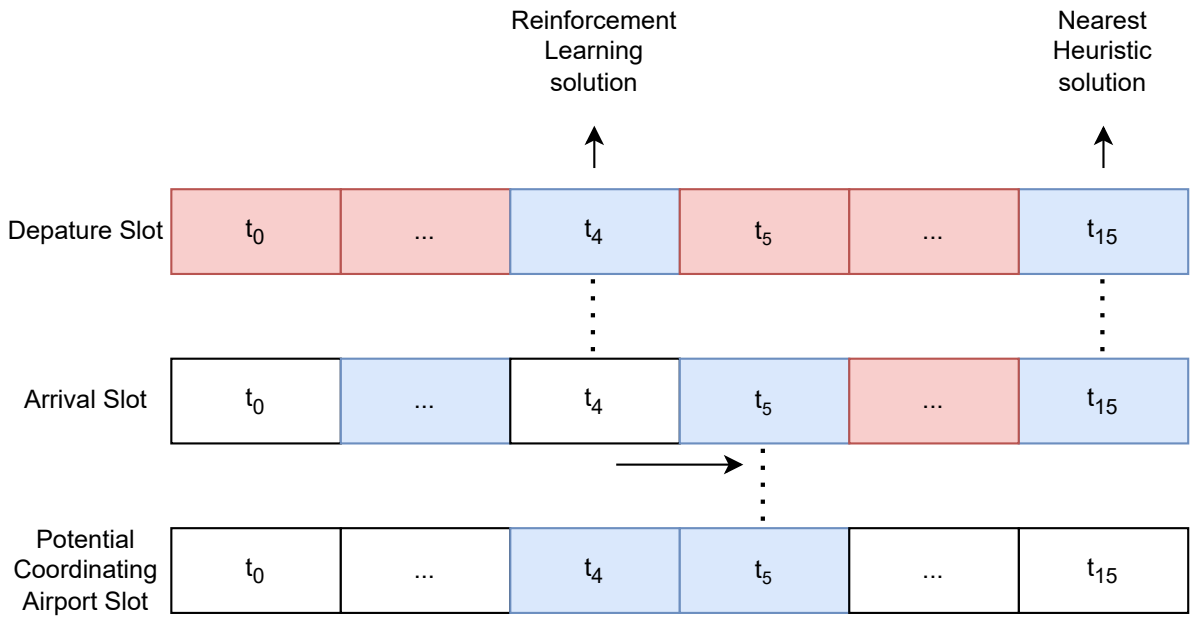


Figure 5.3: A solution illustration: The Reinforcement Learning agent achieves lower displacement by capturing the policy of the arrival airport, which can assign a delay to one of its movements to create a slot for the RL agent. (Red: Over-capacity slot, White: Capacity of the slot equals 0, Blue: Under-capacity slot).

while keeping the focus on hub-level coordination and policy development.

5.2.2 Generating Scenarios

We prepare the 2018 OAG dataset by removing missing entries, duplicates, and unnecessary columns. We standardize flight times by converting local times to UTC and binning them into 5-minute intervals. Departure and arrival airports are categorized into four classes: Hong Kong, Singapore, Bangkok, and Outside (hub). We group flights by date, with each day representing a unique scenario for the learning environment.

For airport capacities, we set Changi Airport's capacity to either 4 movements per 5-minute interval (heavy-reduced capacity) or 5 movements per 5-minute interval (medium-reduced capacity) to simulate a reduction in operational capacity. Other airports maintain fixed capacities based on flight demand from the OAG data within each scenario. We provide detailed capacity configurations in a later section.

Figure 5.4 illustrates an overview of a scenario generated by the learning environment. The flight movements are derived from the 2018 OAG data, where each movement repre-

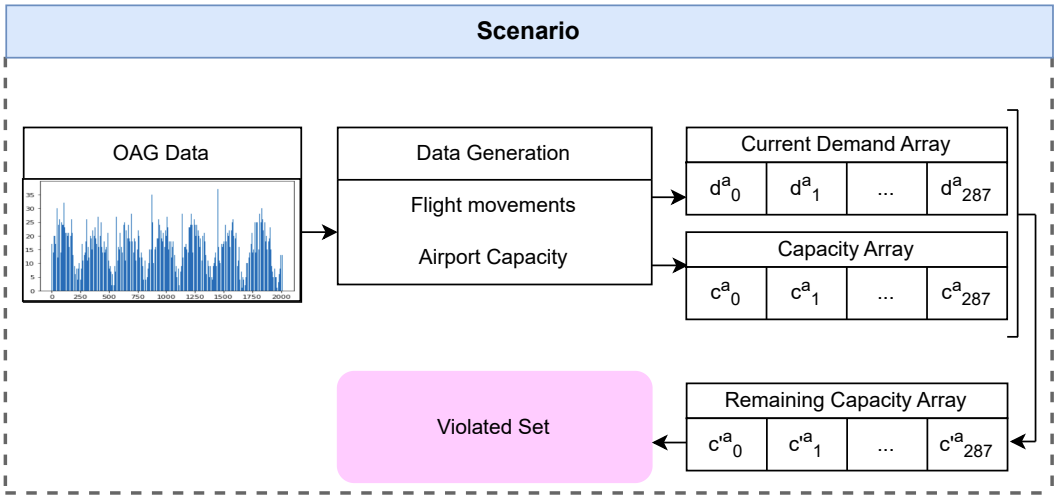


Figure 5.4: An overview of a scenario and its encapsulated data.

sents either a departure or an arrival. A flight movement $m \in M$ is defined as a tuple $(a_{dep}, t_{dep}, a_{arv}, t_{arv})$, where $a_{dep}, a_{arv} \in A$ and $t_{dep}, t_{arv} \in T$. Each departure or arrival movement associated with a time slot $t \in T$ at airport $a \in A$ increases the current demand $d_t^a \in D_T^A$ by one unit. The current demand d_t^a reflects the total number of movements assigned to the time slot t_a. The capacity for a time slot t at airport a is represented by c_t^a.

The Reinforcement Learning agent focuses on reallocating departure movements at Changi Airport to resolve capacity violations. Let $T_{violated} = \{t_S \mid d_t^S > c_t^S\}$ denote the set of all violated time slots for the Singapore agent. If a movement m has $t_{dep} \in T_{violated}$, it is added to the violated set V. Table 5.1 summarizes the notations used in this chapter.

5.2.3 Learning Mechanism

The learning environment initializes each episode by generating a scenario that includes flight movements and airport capacities. The capacity and current demand for each time slot of an airport aa are represented as arrays: $Capacity\_Array^a = [c_0^a, c_1^a, ..., c_{287}^a]$ and $Current\_Demand\_Array^a = [d_0^a, d_1^a, ..., d_{287}^a]$, respectively. Each array has a size of (1x288), corresponding to the 288 5-minute time slots in a day. The remaining capacity array, $Remaining\_Capacity\_Array^a = [c'^a_0, c'^a_1, ..., c'^a_{287}]$, is calculated as $c'_t = c_t - d_t$. Using this, the violated set V is identified.

Table 5.1: Summary of Notations

Notation	Description
$r \in R$	Set of flights: $r = (a_{dep}, t_{dep}, a_{arv}, t_{arv})$ (departure/arrival airports and time slots).
A	Set of airports: $A = \{B, H, O, S\}$, where B, H, O, and S represent level-3 airports at Bangkok, Hong Kong, Outside (hub), and Singapore, respectively.
T	Set of time slots: $T = \{0, 1, \ldots, 287\}$ (5-minute intervals).
C_T^A	Capacity constraints: c_t^a for time slot t at airport a.
D_T^A	Current demands: d_t^a for time slot t at airport a.
$T_{violated}$	Violated time slots: $T_{violated} = \{t \mid d_t^a > c_t^a\}$.
V	Unaccommodated flights: Flights with t_{dep} or t_{arv} in $T_{violated}$.

At the start of each time step, the learning environment randomly selects an unaccommodated movement from the violated set. Addressing movements one by one is a common practice for searching-based approaches (Ribeiro et al., 2019a), (Wang et al., 2019a). The selected movement is then encapsulated as state observations, which serve as inputs for the agent's decision-making process.

After the agent takes an action, one of the following three cases will occur:

- Case 1: The newly assigned time slot for the selected movement satisfies the capacity constraints of both the departure and arrival airports.

- Case 2: The new time slot satisfies the capacity constraints of the departure airport (Changi, Singapore) but not the arrival airport. However, if the arrival airport can accommodate the movement by delaying one of its own movements based on its policy, the selected movement will be assigned to the new time slot.

- Case 3: If neither Case 1 nor Case 2 is satisfied, the movement retains its original time slot.

This mechanism ensures stepwise handling of unaccommodated movements while incorporating airport-specific policies.

The learning environment updates the current demand arrays, remaining capacity arrays, and the violated set based on the outcomes of the above cases. At the end of each time step, the learning environment provides feedback (reward), and the agent adjusts its model accordingly. This process repeats iteratively until the violated set is cleared or the predefined maximum number of steps per episode is reached.

5.3 Reinforcement Learning Model

5.3.1 Action Space

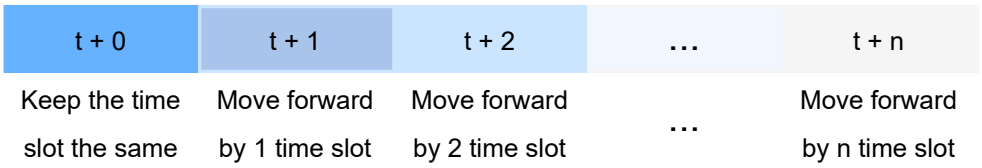


Figure 5.5: Action space of the Reinforcement Learning agent.

Figure 5.5 illustrates the action space of the agent. The agent can delay a movement by up to 12 time slots, i.e., move forward by n time slots ($+n$), where $n <= 12$, or retain the current time slot. While it is possible to increase the maximum displacement and expand the action space, this approach has two significant drawbacks. First, a larger action space can negatively impact training efficiency (Yehudai et al., 2022). Second, excessive delays may compromise schedule acceptability (Zografos et al., 2018). To address these issues, the maximum displacement is capped at 1 hour (12 time slots) by limiting the number of actions. Empirical studies have shown that maximum slot displacements rarely exceed 12 time slots (Zografos et al., 2018; Katsigiannis and Zografos, 2021). Therefore, setting this limit ensures alignment with empirical experiments while avoiding excessive computational complexity. However, determining the optimal number of actions for the RL agent requires further investigation.

5.3.2 State Observation

In Reinforcement Learning, the agent's decision-making inputs are encapsulated as state observations. The design of these observations significantly impacts the agent's performance. An overly complex state observation with excessive inputs can hinder training efficiency, while insufficient information may lead to poor performance or failure to achieve model convergence. As shown in Figure 5.3, the agent requires, at a minimum, data on the demand and capacity of the corresponding departure and arrival time slots for the given movement. For effective coordination, the agent also needs information on potential movements at the arrival airport that could be delayed to accommodate the shifted movements. In summary, the agent requires three key types of information: details about its own time slots, the time slots of the associated arrival airport, and the time slots of the potential coordinating airport.

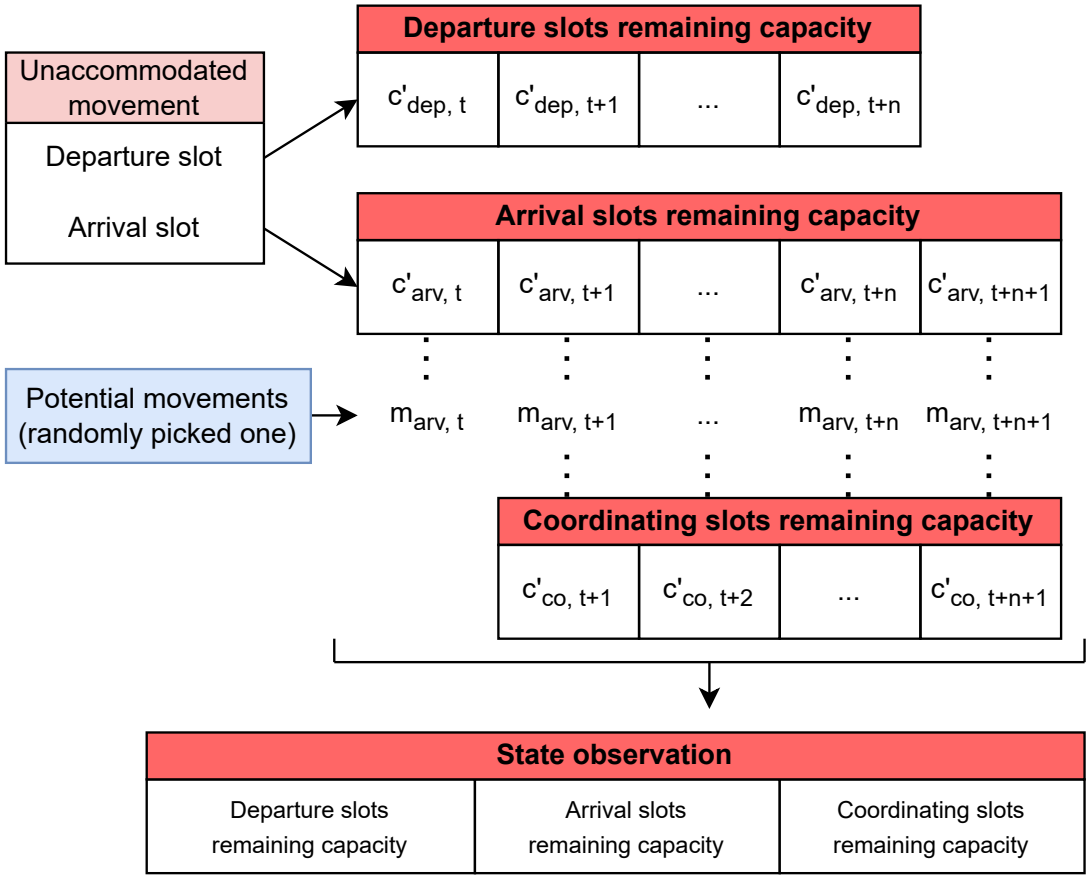


Figure 5.6: State observation of the Reinforcement Learning agent.

Figure 5.6 illustrates the design of the state observation. Since the capacity is fixed per

scenario, the remaining capacity, c'_t, encapsulates both capacity and demand information. While it might seem intuitive to provide the agent with the remaining capacity for all 288 time slots, this approach would result in a large state observation of size $(288, 3)$, considering the three identified pieces of information. The abundance of irrelevant data would lead to inefficient training.

To improve efficiency, the state observation is designed to include only the remaining capacity of time slots directly related to the agent's possible actions. Since the arrival airport's policy involves delaying one of its movements to the next time slot, the observation also includes $c'_{arv,t+n+1}$, covering $n + 1$ time slots for the arrival airport. Each arrival time slot, t, is associated with various movements, $m_{arv,t}$, which can be shifted to create space for the unaccommodated movement. However, considering all potential coordinating movements would result in a complex and inefficient state observation.

We first test with the design of randomly selecting one potential coordinating movement per time slot and incorporating the remaining capacity of the corresponding time slot of the coordinating airport, $c'_{co,t}$. If no potential coordinating movement exists in a time slot, the observation assigns $c'_{co,t} = 0$. This streamlined approach balances complexity and efficiency while retaining the critical information needed for effective decision-making.

5.3.3 Reward Design

The reward design is crucial for guiding the Reinforcement Learning agent toward learning an effective policy (Dayal et al., 2022). The objectives are to ensure that the agent assigns new time slots without causing capacity violations while minimizing displacement. These objectives are translated into the following reward components:

- $R_{local} = z * (-|t - t_m|)$, where t represents the newly assigned time slot and t_m denotes the original time slot of the movement. This reward component penalizes the agent for displacing the movement further from its original time slot, encouraging minimal displacement. The constant z scales the reward to a smaller value to enhance training efficiency. The agent receives this reward component at each time step.

- $R_{solving} = +s$, where s is a constant value, is awarded to the agent when the considered movement is successfully cleared from the violated set. This reward encourages the agent to resolve capacity violations. To balance $R_{solving}$ with R_{local}, we set $R_{solving} = z * (n + 1)$. Striking a balance between these components is crucial to prevent the agent from adopting undesired behavior, such as prolonging the episode by creating additional unaccommodated movements to maximize $R_{solving}$.

- $R_{time\_step} = -p$, where p is a constant, is assigned to the agent at every time step. This reward component encourages the agent to solve the problem more quickly and discourages undesired behavior, such as unnecessarily prolonging the episode.

5.4 Experiments

5.4.1 Training Configurations

The Official Airline Guide (OAG) provides one of the most comprehensive and widely recognized sources of global aviation data (OAG, 2025). According to OAG, the OAG Schedules Analyzer contains airline-filed schedules that represent planned operations or intended flight demand, whereas the OAG Status dataset captures executed operations, including real-time flight statuses such as delays and cancellations. Figure 5.7 illustrates the relationship between these two datasets and their respective roles across operational phases. As the Schedules Analyzer reflects airline-filed schedules that convey airlines' operational intentions, it provides a credible and realistic data source for representing flight demand in this study.

We utilize data from the OAG Schedules Analyzer to represent daily flight movements. Figure 5.8 illustrates the distribution of movements across 288 time slots within a single day at Singapore Changi Airport.

For Singapore Changi Airport, capacity reductions are simulated relative to the nominal capacity of 6 movements per five-minute interval.. The airport's capacity is fixed at $c^S_t = 4$ (heavy-reduced) or $c^S_t = 5$ (medium-reduced capacity), $\forall t \in T$. For the other airports in the multi-airport network, where official declared capacity data are not publicly available, the

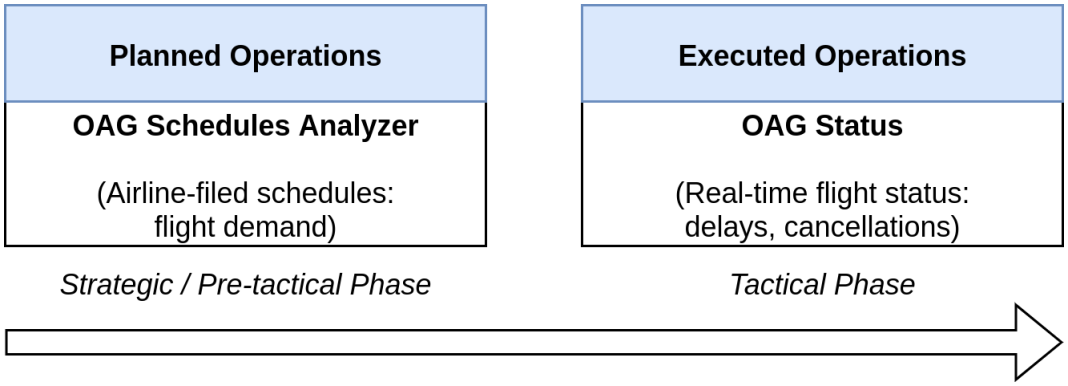


Figure 5.7: Relationship between OAG Schedules Analyzer and Status datasets across the planning and operational phases.

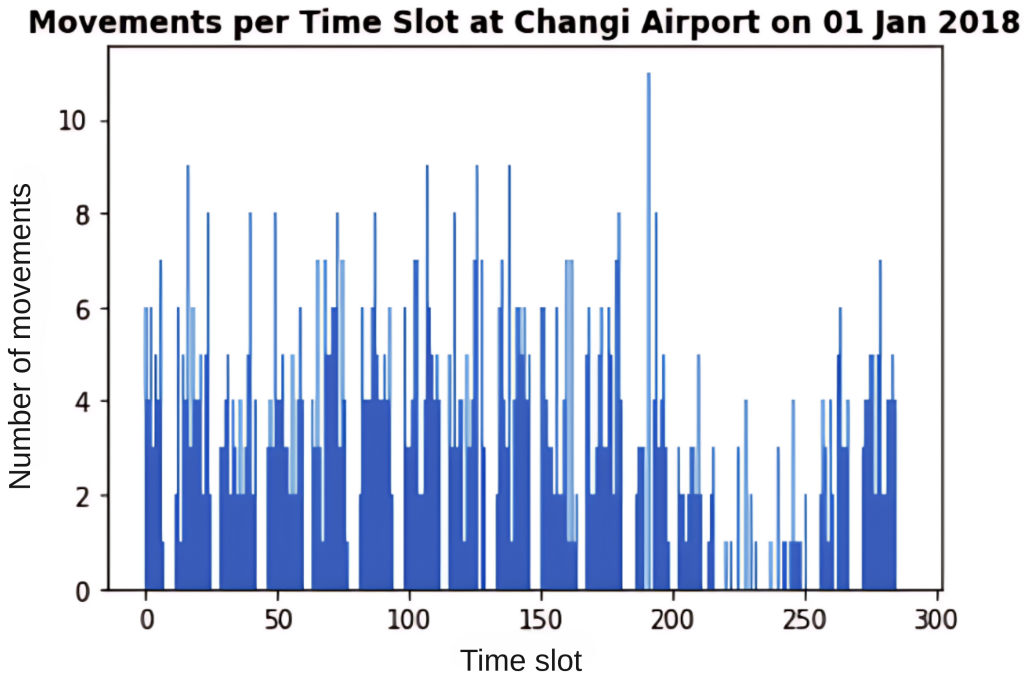


Figure 5.8: The number of movements per time slot at Changi airport on 01 Jan 2018.

capacity per time slot is approximated based on the observed number of scheduled movements in the OAG dataset. The capacity at each time slot is defined as the observed number of scheduled movements plus a small tolerance term, $c'^{a}_{t} \in 0, 1, \forall a \in B, H, O, \forall t \in T$, which represents minor operational flexibility allowed during slot coordination. Training is conducted using medium-reduced capacity scenarios, while testing is performed on both medium- and heavy-reduced capacity scenarios to evaluate the agent's performance on unseen conditions. This setup allows for a comprehensive analysis of the Reinforcement Learning agent's ability to generalize across different scenarios.

5.4.2 Learning Algorithms and Hyper-Parameters

Besides the differences in the inputs, which arise due to modifications in the state observations, we adopt the Deep Q-Network (DQN), as outlined in Algorithm 1, as the learning algorithm.

Table 5.2 summarizes the hyper-parameters used for training the DQN. The number of actions (n) is 12, corresponding to a maximum displacement of 60 minutes per movement. The scaling factor z in R_{local} is set to 0.1, and the penalty p in R_{time_step} is set to 1.3 to balance the reward obtained from solving a movement. This design prevents the agent from creating additional unaccommodated movements to earn higher rewards. Lastly, the maximum number of steps per episode is defined as twice the number of unaccommodated movements in that episode.

5.5 Results and Discussion

5.5.1 Training Results

Figure 5.9 and Figure 5.10 show that the DQN agent can achieve convergence, as indicated by the increasing trend in average episode reward and the decreasing trend in average episode length. These trends highlight the agent's ability to allocate slots more efficiently, reflected in higher rewards, and solve problems faster, evidenced by shorter episode lengths.

Table 5.2: DQN Training Hyper-Parameters

Parameter	Value	Description
Discount Rate (γ)	0.99	Discount factor for future rewards.
Learning Rate	0.0001	Step size for updating network parameters.
Target Update Cycle	10,000	Number of steps between target network updates.
Training Steps	100,000	Total number of training steps.
Number of Actions (n)	12	Maximum displacement of 60 minutes per time step.
z in R_{local}	0.1	Scaling factor for the local reward.
p in $R_{\text{time\_step}}$	1.3	Penalty per time step to offset solving rewards.
Max. Steps per Episode	2x Unaccom.	Upper limit for time steps in an episode.

The current average episode length of 23.47 and the average number of 15.03 unaccommodated movements per episode suggest that there is still room for performance improvement, especially given the current average episode length of 23.47. For an episode with 15 unaccommodated movements, the expected optimal performance after convergence is approximately 10.5, with an episode length of 15. Extending the number of training steps could enhance the agent's performance further.

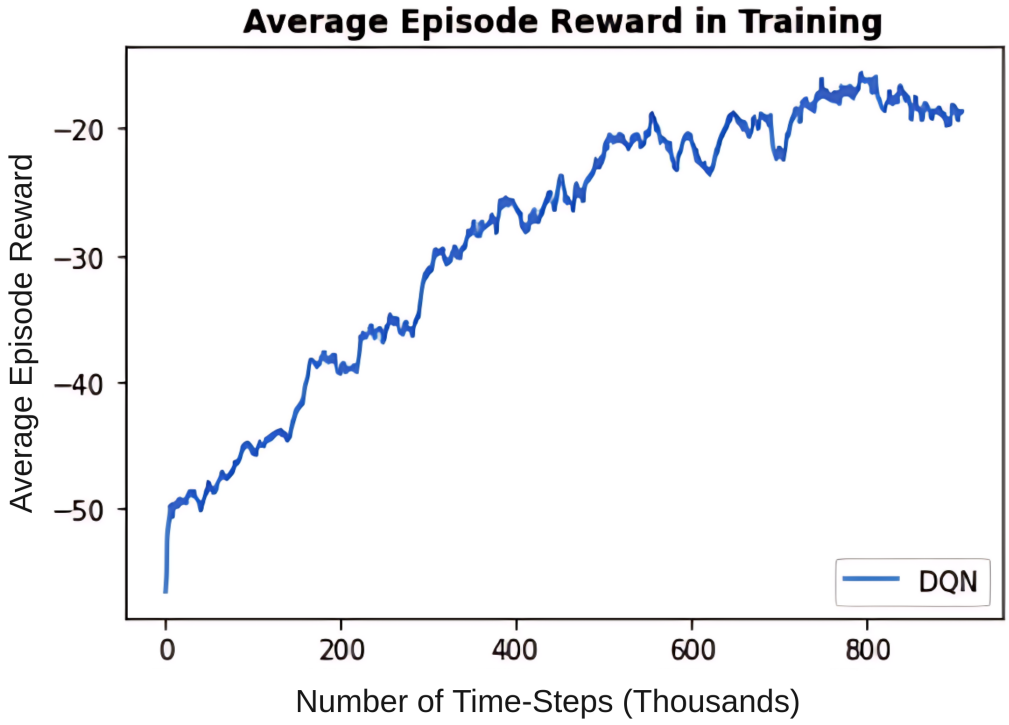


Figure 5.9: Episode reward convergence of the Deep Q-Network (DQN) model.

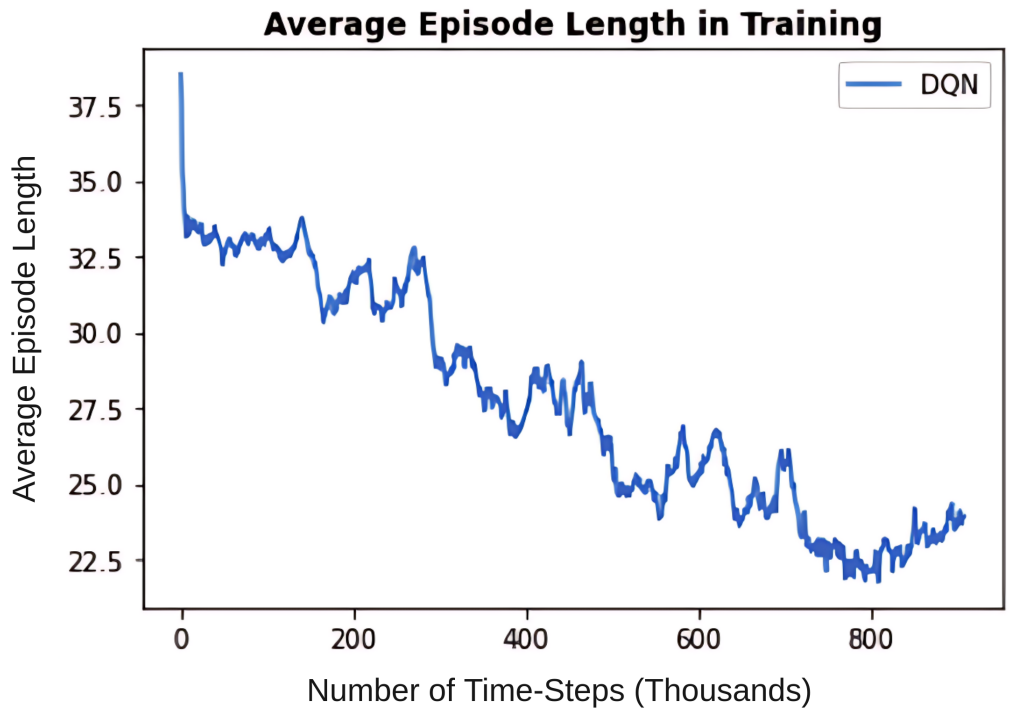


Figure 5.10: Episode length convergence of the Deep Q-Network (DQN) model.

5.5.2 Testing Results

We employ the same metrics as Chapter 4 for performance comparison:

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

We validate the performance of the DQN agent against the Nearest Heuristic. Algorithm 3 presents the pseudo-code of the Nearest Heuristic. Results from Figure 5.11 indicate that, under a medium-reduced capacity scenario, the RL agent effectively learns to assign movements to the nearest available slot. Under a medium-reduced capacity scenario, the availability of multiple nearest slot pairs limits the opportunities for the agent to achieve

Case 2 (as discussed in Section Learning Mechanism). Consequently, the optimal solution in this scenario closely aligns with the performance of the Nearest Heuristic. There are 15 unaccommodated movements, and in most cases, the Reinforcement Learning (RL) agent assigns delays comparable to those of the Nearest Heuristic. The total delay of the RL agent is 45 time slots, slightly higher than the 41 time slots of the Nearest Heuristic. However, the RL agent achieves a lower maximum displacement across all movements at 10 time slots, compared to 11 time slots for the Nearest Heuristic. Both methods successfully address all unaccommodated movements.

Other heuristic approaches were not included in the baseline comparison because they are primarily designed for centralized optimization frameworks, where a single decision-maker jointly allocates slots across all airports to achieve a globally optimal solution. While such centralized methods are theoretically efficient, they are not operationally feasible in regions without a central scheduling authority, such as ASEAN, where airports maintain autonomous decision-making. This study instead focuses on developing and benchmarking decentralized frameworks, where each airport maintains its own decision-making autonomy. In practice, achieving a fully optimal global solution is less critical than ensuring feasible decentralized solutions.

Algorithm 3 Nearest Heuristic for Slot Allocation

Input: Violated set V, $Remaining\_Capacity\_Array^a$

Output: Final slot allocations and unaccommodated requests

while $V \neq \emptyset$ **do**
  Select $m \in V$ with time slot t, departure airport dep_m, and arrival airport arv_m.
  if $\exists\ t'$ such that $Remaining\_Capacity\_Array^{dep_m}[t'] > 0$
    and $Remaining\_Capacity\_Array^{arv_m}[t' + flight\_time] > 0$ **then**
    Update allocated time slot: $t \leftarrow t'$
    Update $Remaining\_Capacity\_Array^a$ for both dep_m and arv_m.
    Remove m from the violated set: $V \leftarrow V \setminus \{m\}$
  else
    Mark m as unaccommodated.
  end if
end while
return Final slot allocations and list of unaccommodated requests.

Figure 5.11 illustrates that, in a heavy-reduced capacity scenario, the RL agent outperforms the Nearest Heuristic by leveraging coordination to minimize delay. For movements with large nearest-slot displacements (e.g., 20-time-slot displacement), the RL agent significantly reduces delay (e.g., to 3 time slots and 9 time slots). The RL agent achieves a total delay of 84 time slots compared to the Nearest Heuristic's 107 time slots, with a lower maximum displacement (11 time slots vs. 20 time slots). For the number of unaccommodated movements in the Nearest Heuristic, only delays within 12 time slots are considered to ensure a fair comparison with the RL agent. Under this condition, the RL agent achieves a lower number of unaccommodated movements, with 3 compared to 5 for the Nearest Heuristic. Additionally, the RL agent has a lower average displacement per movement at 3.82 time slots, compared to 4.86 time slots for the Nearest Heuristic.

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

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

Figure 5.11: Solutions for medium- and heavy-reduced capacity scenarios (Light yellow: Equal delay, Dark yellow: Lower delay).

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

M_3 is measured in *requests*, whereas all other metrics are measured in *time slots*.

5.6 Summary

We present a Reinforcement Learning (RL) formulation for a multi-airport system with diverse decision-making policies. The results indicate that the Deep Q-Network (DQN) agent can leverage interactions between airports to reduce delays. Under scenarios with higher capacity reductions, the RL agent surpasses the Nearest Heuristic by capturing inter-airport dynamics, achieving Case 2 and resulting in lower delays. However, there remains scope for improvement by increasing the number of training steps. Future work will focus on expanding the problem to include more airports with additional policies and incorporating more informative features into the state observations to enhance performance.

When considering multiple airports utilizing distinct RL-based policies, the convergence of the DQN learning process is not guaranteed due to two key reasons. First, non-stationarity arises as multiple learning agents continually update their policies, which introduces instability. In Multi-Agent Reinforcement Learning (MARL), the state transitions and reward functions perceived by one agent depend on the actions of other agents, whose policies are not fixed during the learning process. This dynamic behavior leads to a non-stationary environment. Second, conflicting objectives among different airport agents with RL-based policies can emerge, as they may pursue varying goals. These agents might learn to counter the policies of others to maximize their own rewards, potentially creating inefficiencies in the system.

To address these challenges, techniques such as Centralized Training with Decentralized Execution (CTDE), self-play, stabilizing experience replay, opponent modeling, meta-learning, and communication protocols can be explored in future research.

Chapter 6

Incorporating Fairness into Reinforcement Learning for Airport Slot Scheduling

Reinforcement Learning (RL) has been explored in Air Traffic Management (ATM) contexts. In the pre-tactical phase, RL has been applied to adjust flight schedules by introducing delays to ease congestion.

However, the integration of RL with fairness considerations in airport slot scheduling remains largely unaddressed. To tackle this gap, in this chapter, we build on the literature of existing fairness integration methods, including the mini-max criterion, proportionality principles, and constraint-based strategies, to formulate the airport slot scheduling problem with fairness considerations as a Markov Decision Process (MDP). While constraint-based fairness can be incorporated by limiting available actions, achieving fairness through RL depends critically on reward function design, requiring a balance between efficiency and fairness objectives.

Previous chapters demonstrate that introducing positive rewards for accommodating unassigned requests enhances training stability. Building on this insight, we extend the reward design by introducing positive, dense reward signals at each time step, aligning both efficiency and fairness goals. We compare the dense-reward approaches with the sparse-

reward approaches, where rewards are issued only at terminal states, to evaluate their effects on training convergence. The content of this chapter is based on our published work, "Reinforcement Learning for Airport Slot Allocation: Incorporating Fairness and Trade-off Analysis", presented at the 25th Integrated Communications, Navigation and Surveillance Conference (ICNS), 2025.

6.1 Overview

Although the literature explores fairness through multiple perspectives, three major approaches dominate the integration of fairness into the airport slot scheduling problem: (1) the mini-max criterion, (2) the principle of proportionality, and (3) constraint-based methods. The fairness approaches identified in the literature and summarized in Chapter 2 are outlined below:

Mini-max Criterion

The mini-max criterion focuses on minimizing the displacement experienced by the most adversely affected flight. Displacement is defined as the absolute difference between the requested and assigned slot times, expressed as:

$$\text{Minimize} \quad \max_{m \in M} |t_m - t'_m| \,, \tag{6.1}$$

where t_m and t'_m denote the requested and allocated time slots for movement m, respectively.

By minimizing the largest displacement, this approach promotes inter-flight fairness by ensuring that the most delayed flight faces the smallest possible deviation (Jacquillat and Odoni, 2015; Zografos et al., 2018; Ribeiro et al., 2018, 2019b).

Principle of Proportionality

(Zografos and Jiang, 2016, 2019) developed fairness models based on the principle of propor-

tionality within the IATA framework. Rather than evaluating individual flights, these models consider airlines as units and allocate total displacement proportionally to each airline's volume of slot requests. The models aim to minimize the maximum deviation of an airline's assigned displacement relative to the network-wide average, capturing a broader notion of fairness.

This proportional fairness is quantified using the following index:

$$\mathrm{pa} = \frac{\frac{\mathrm{da}}{D}}{\frac{|\mathrm{Ma}|}{\sum a \in A|\mathrm{M_a}|}}, \forall a \in A \tag{6.2}$$

where the numerator represents an airline's share of total delays and the denominator reflects its share of total slot requests. A fairness index $\mathrm{p_a}$ equal to 1 indicates complete fairness, while deviations highlight disparities between airlines.

The corresponding fairness objective can be expressed as:

$$Min\{Max\{|\mathrm{p_a} - \frac{\sum_{a \in A} \mathrm{p_a}}{|A|}, \forall a \in A|\}\} \tag{6.3}$$

Although complete proportional fairness would imply that all airlines experience delays exactly aligned with their demand, operational and capacity constraints make this ideal practically unattainable. To better align the model with real-world conditions, (Zografos and Jiang, 2016) later incorporated weighted factors accounting for aircraft size and flight length.

Constraint-Based Approach

Constraint-based methods introduce fairness by embedding strict conditions directly into the scheduling models, rather than optimizing fairness as a standalone objective. For instance, (Jacquillat and Odoni, 2015) proposed two fairness constraints:

$$|u_i| \leq \delta, \quad \forall i \in \mathcal{F}, \sum_{i \in \mathcal{F}} |u_i| \leq \Delta, \tag{6.4}$$

where u_i denotes the displacement for flight i, δ sets the maximum permissible displacement for any single flight, and Δ limits the total displacement across the entire system.

The maximum displacement constraint establishes a baseline level of service quality, ensuring no flight experiences an excessive delay. Although this method effectively controls extreme cases, it does not explicitly balance the relative treatment across different flights or airlines. Whether explicitly modeling these disparities can yield better fairness, at the potential expense of efficiency, remains an open question. Enforcing strict displacement thresholds safeguards the worst-off flights but may also diminish overall system performance, illustrating the fundamental trade-off between fairness and efficiency.

Building on these approaches, we examine various RL formulations for incorporating inter-flight fairness, as detailed in the following sections.

6.2 Learning Environment

6.2.1 Generating Scenarios

To implement Reinforcement Learning, the agent requires a learning environment to interact with. We adopt the environment introduced in Chapter 4 and Chapter 5 with modifications in the rewards given to the agent to incorporate fairness. At the beginning of each episode, the environment randomly selects a scenario comprising flight movements and airport capacities. These scenarios are created based on the 2018 OAG dataset, which records daily flight operations, where each day represents an independent scenario. Since the OAG data reflects scheduled flights where capacity meets demand, we simulate a capacity reduction at Changi Airport. This modification allows the RL agent, acting as the slot coordinator for Changi, to learn slot allocation strategies while balancing system efficiency and fairness. The capacities

of the other airports remain constant across all scenarios. More detailed information about airport capacities will be provided later.

We employ the scenario generated by the learning environment, identical to that used in Chapter 5, as illustrated in Figure 5.4. The flight movements are extracted from the 2018 OAG data, where each movement corresponds to either a departure or an arrival. A flight movement $m \in M$ is represented as a tuple $(a_{dep}, t_{dep}, a_{arv}, t_{arv})$, where $a_{dep}, a_{arv} \in A$ are the departure and arrival airports, and $t_{dep}, t_{arv} \in T$ are the respective departure and arrival times. Each departure or arrival during a time slot $t \in T$ at an airport $a \in A$ increases the current demand $d_t^a \in D_T^A$ by one unit. Here, d_t^a denotes the number of movements assigned to time slot t at airport a, and c_t^a denotes the capacity constraint at that time slot.

In this framework, Reinforcement Learning is applied to reallocate departure movements at Changi Airport to prevent capacity violations. We focus only on departure flights, as the airport has full control over delaying departures. Let

$$T_{violated} = \{t_S \mid d_t^S > c_t^S\}$$

denote the set of time slots at Changi Airport where demand exceeds the available capacity. A flight movement m with a departure time $t_{dep} \in T_{violated}$ is added to the set of unaccommodated movements V.

6.2.2 Learning Mechanism

At the beginning of each episode, the learning environment generates a scenario that includes both the flight movements and the airport capacities. For each airport a, the capacity and current demand across all time slots are represented as arrays:

$$\texttt{Capacity\_Array}^a = [c_0^a, c_1^a, \ldots, c_{287}^a]$$

and

$$\texttt{Current\_Demand\_Array}^a = [d_0^a, d_1^a, \ldots, d_{287}^a],$$

respectively. As each day consists of 288 5-minute intervals, both arrays have dimensions of 1×288.

The remaining capacity array,

$$\texttt{Remaining\_Capacity\_Array}^a = [c'^a_0, c'^a_1, \ldots, c'^a_{287}],$$

is derived from the capacity and current demand values according to:

$$c'_t = c_t - d_t. \tag{6.5}$$

Using the remaining capacity array, the violated set V is identified. At the beginning of each time step, an unaccommodated movement is randomly drawn from the violated set. Handling each movement one by one is a practical approach, consistent with techniques employed in heuristic methods (Ribeiro et al., 2019a; Wang et al., 2019a). The selected movement, which encapsulates all necessary information, is then structured into state observations that serve as inputs for the decision-making process.

6.3 Reinforcement Learning Model

6.3.1 Action Space

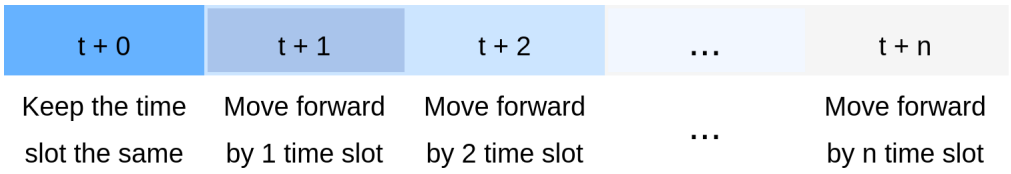


Figure 6.1: Action space of the Reinforcement Learning agent.

Figure 6.1 depicts the action space available to the agent. The agent can either maintain the current time slot or delay a movement by up to n time slots. Although it is possible to allow larger displacements, thereby increasing the number of available actions, doing so presents two key disadvantages. First, a larger action space can significantly degrade train-

ing efficiency (Yehudai et al., 2022). Second, introducing excessive delays could undermine schedule feasibility and acceptability (Zografos et al., 2018). Therefore, we cap the maximum displacement at 1 hour ($n = 12$), limiting the number of available actions. In the constraint-based fairness formulation, we further restrict n to 6 to ensure no flight experiences a delay greater than 30 minutes, thus maintaining a minimum level of service quality.

6.3.2 State Observation

To make informed decisions, the agent must be aware of the current demand and capacity for each time slot. Since the capacities are static, the essential information can be condensed into the remaining capacity c'_t.

While it might seem logical to provide the agent with the remaining capacities for all 288 time slots, this approach would inundate the agent with largely irrelevant data, thereby reducing training efficiency. For example, when handling a request linked to time slot 100, the remaining capacity at time slot 200 is unlikely to have any influence. Providing a complete state observation over all time slots would burden the agent and slow down the learning process, as proven in Chapter 4. To address this, we employ a Local State Observation approach, represented by an array of size $2 \times (n + 1)$. This array includes the remaining capacities for the current time slot and the subsequent n time slots, covering both the departure and arrival sides. Figure 6.2 visualizes the Local State Observation structure used by the Reinforcement Learning agent.

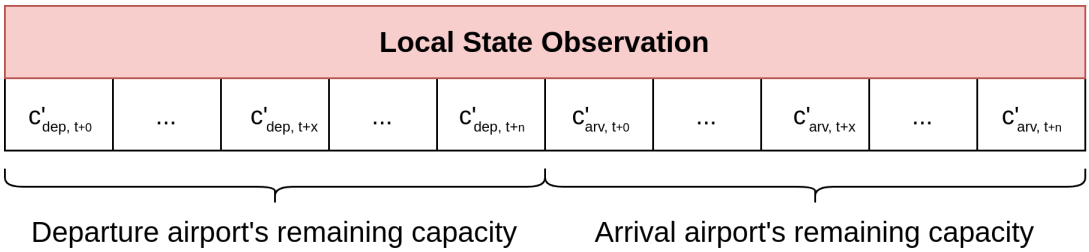


Figure 6.2: Local State Observation of the Reinforcement Learning agent.

6.3.3 Reward Design

The structure of the reward function plays a crucial role in determining the training efficiency of the Reinforcement Learning agent (Dayal et al., 2022). To guide the agent towards the objective of minimizing total delays (i.e., improving system efficiency), we incorporate the following reward component:

- $R_{global} = -(\sum_{m \in M} |t - t_m|)$, where t is the newly assigned time slot for the request, and t_m is its original scheduled time. This displacement penalty mirrors the standard cost function commonly used in airport slot scheduling problems.

Additionally, we integrate supplementary reward components that have been demonstrated to enhance training efficiency in Chapter 4:

- $R_{solving} = +s$: Here, s is a positive constant awarded when the agent successfully removes the current request from the violated set. This incentive encourages behaviors that actively resolve conflicts. However, it is crucial to balance $R_{solving}$ with R_{local} components; otherwise, the agent might exploit the reward by creating more unaccommodated requests or unnecessarily extending the episode length to accumulate rewards.

- $R_{time\_step} = -p$: In this case, p is a constant penalty applied at each time step, incentivizing the agent to solve requests promptly rather than delaying decisions.

While fairness in the constraint-based approach is addressed by limiting the number of available actions to maintain an acceptable service level, we integrate fairness into the reward design when applying the mini-max criterion and the principle of proportionality. Fairness considerations are incorporated both locally (at each time step) and globally (at terminal states). The local reward provides frequent guidance towards achieving the broader global objective.

Without local reward signals, agents are prone to encountering the sparse reward problem, where feedback is provided only after a long sequence of actions, typically at the end of an episode. Sparse rewards introduce several challenges. First, they create a credit assignment

problem, making it difficult for the agent to determine which specific actions contributed to the eventual outcome, thereby complicating the policy update process. Second, sparse feedback hinders effective exploration, as the agent lacks regular guidance to identify promising strategies within the action space. As a result, the agent may spend significant time exploring suboptimal behaviors. Third, sparse rewards reduce training efficiency, leading to slower learning progress and requiring a substantially larger number of episodes to achieve effective policy convergence, thereby increasing computational costs and training time.

To overcome these challenges, we propose the following reward components to incorporate fairness through the mini-max criterion and the principle of proportionality.

Mini-max Criterion Reward Components

- $R_{min\_max\_local} = z \cdot \big(-|t - t_m|\big)$: Here, t is the newly assigned time slot for the request and t_m is the original time slot. This reward penalizes displacement locally, following the mini-max objective presented in Equation 6.1. The constant z serves as a scaling factor to normalize the reward for better training stability. This reward is applied at every time step.

- $R_{min\_max\_global} = -(\max_{m \in M} |t - t_m|)$: This terminal reward reflects the mini-max criterion globally, directly aligning with Equation 6.1.

Principle of Proportionality Reward Components

In this study, we focus solely on inter-flight fairness. To integrate fairness based on the principle of proportionality, we first compute the average delay:

$$p_f = \frac{\sum_{m \in M} |t - t_m|}{M} \tag{6.6}$$

- $R_{proportional\_local} = -|t - p_f|$: Here, M represents the total number of unaccommodated requests solved at the current time step. This reward ensures fairness locally by dis-

couraging assignments that deviate significantly from the average, consistent with the idea behind Equation 6.3.

- $R_{proportional\_global} = -\sum_{i \in M} |t_i - p_f|$: In this case, t_i denotes the newly assigned time slot for each request, while p_f is the average delay calculated based on the total number of resolved unaccommodated requests at the terminal state. This global reward mirrors the fairness principle outlined in Equation 6.3. We sum deviations from the average rather than using the maximum, providing a stronger reward signal that scales with the number of resolved requests and encouraging better learning dynamics.

It is important to note that while local reward components help guide learning, they only partially reflect the final global objectives. Overemphasizing local rewards might divert the agent's focus from the true global goals. In the experiment section, we analyze the effects of different reward weightings to mitigate this issue.

6.4 Experiments

6.4.1 Training Scenarios

We use the OAG dataset to obtain flight movement information, which reflects the allocated movements that comply with existing capacity constraints. In the absence of disruptions that might reduce available capacity, we infer capacity values based on the observed movements. For airports other than Changi, we assume that the capacity at each time slot corresponds to the number of scheduled movements plus an additional margin of either 0, 1, or 2 movements, expressed as ${c'^{a}}_t \in \{0, 1, 2\}, \quad \forall a \in \{B, H, O\}, \quad \forall t \in T$. This assumption results in relatively dense scenarios with tight capacity constraints.

For Changi Airport (Singapore), we fix the capacity at each time slot to be ${c^{S}}_t = 6$.

6.4.2 Models

We denote the reward components as follows:

$R_1 = R_{\text{min\_max\_local}}$; $R_2 = R_{\text{solving}}$; $R_3 = R_{\text{global}}$; $R_4 = R_{\text{time\_step}}$; $R_5 = R_{\text{min\_max\_global}}$; $R_6 = R_{\text{proportional\_local}}$; $R_7 = R_{\text{proportional\_global}}$.

By assigning different weights to these reward components, we develop several types of models:

Baseline Models

Table 6.1 summarizes the reward settings for the baseline models. These models are designed to focus solely on minimizing total schedule delay. Since R_1 also encourages minimizing displacement, it is used as a per-time-step reward to guide the agent toward the global objective. In Baseline 1, rewards are provided only at the terminal state through R_3, representing a sparse reward setting. To investigate the impact of sparse rewards, we compare Baseline 1 against Baseline 2 and Baseline 3, which include per-time-step reward signals via R_1. Fairness is not explicitly integrated in any baseline model, as indicated by the zero weights assigned to R_5, R_6, and R_7.

Model	R_1	R_2	R_3	R_4	R_5	R_6	R_7
Baseline 1	0	1	0.1	1	0	0	0
Baseline 2	1	1	0	1	0	0	0
Baseline 3	1	1	1	1	0	0	0

Table 6.1: Reward Settings for Baseline Models.

Mini-max Models

Table 6.2 presents the reward settings for the mini-max models. These models emphasize the mini-max reward components R_1 and R_5, while excluding the proportionality-related rewards (R_6 and R_7). We analyze the trade-off between system efficiency and fairness: Mini-max 1 focuses purely on fairness, setting $R_3 = 0$, while Mini-max 2 considers both efficiency and fairness by setting $R_3 = 1$.

Model	R_1	R_2	R_3	R_4	R_5	R_6	R_7
Mini-max 1	1	1	0	1	10	0	0
Mini-max 2	1	1	1	1	10	0	0

Table 6.2: Reward Settings for Mini-max Models.

Principle-of-Proportionality Models

Table 6.3 provides the reward settings for the proportionality models. Like the mini-max models, these emphasize fairness, but through proportionality components. Since R_1 also helps reduce total delay, it is retained in most configurations. To study its contribution, Proportional 1 excludes R_1 entirely. In Proportional 2, a dense reward setting is applied to investigate the impact of providing excessive feedback. Proportional 4–8 explore the trade-off between efficiency and fairness by varying the reward weight combinations.

Model	R_1	R_2	R_3	R_4	R_5	R_6	R_7
Proportional 1	0	1	0.1	1	0	1	0
Proportional 2	1	1	1	1	0	10	0
Proportional 3	1	1	0.1	1	0	1	0
Proportional 4	1	1	0	1	0	1	0
Proportional 5	0.5	1	0.5	1	0	1	1
Proportional 6	0.25	1	0.25	1	0	1	1
Proportional 7	0.75	1	0.75	1	0	1	1
Proportional 8	1	1	1	1	0	1	1

Table 6.3: Reward Settings for Proportional Models.

Constraint-Based Models

Table 6.4 outlines the reward settings for the constraint-based models. In these models, fairness is enforced by limiting the number of available actions to $n = 6$, ensuring that no flight is delayed by more than 30 minutes. Since fairness is incorporated through action restrictions, the fairness-related rewards R_5, R_6, and R_7 are set to zero. Constrained-based 2 retains R_1 as a per-time-step signal to improve efficiency, allowing comparison against

Constrained-based 1, which omits it.

Model	R_1	R_2	R_3	R_4	R_5	R_6	R_7
Constrained-based 1	0	1	0.1	1	0	0	0
Constrained-based 2	1	1	1	1	0	0	0

Table 6.4: Reward Settings for Constraint-Based Models.

Justification of Reward Weight Settings

The reward settings presented in the above tables are designed based on the empirical findings from Chapter 4, which demonstrated that the R_2 (reward solving) component is essential for effective agent training. Therefore, R_2 is retained in all models. The R_1 component has also been shown to significantly optimize system efficiency and is therefore retained in most models.

For the baseline models, R_2 is included to maintain training stability, while R_1 is omitted in Baseline 1 to assess its significance. Since R_1 is excluded, R_3 (local delay reward) is scaled to a smaller value to ensure effective learning. As these models do not incorporate fairness, no weight is assigned to fairness-related reward components.

For the mini-max models, R_5 (fairness reward) is scaled to be comparable in magnitude to R_1, as R_5 is applied only once at the terminal state, whereas R_1 accumulates across multiple steps.

For the proportional models, R_1 and R_3 optimize system efficiency, while R_6 and R_7 capture proportional fairness. Their weights are systematically tuned to analyze the efficiency–fairness trade-off.

For the constraint-based models, the rationale is similar to that of the baseline model, as fairness is captured indirectly through action constraints rather than explicit reward formulation.

Since R_5, R_6, and R_7 represent different fairness approaches, assigning non-zero weights to all of them simultaneously would lead to a misleading interpretation of the individual effects of each fairness method.

6.4.3 Learning Algorithms and Hyper-Parameters

During training, we utilize the Deep Q-Network (DQN) algorithm. DQN is particularly well-suited for environments with discrete action spaces, and we further enhance its performance by implementing experience replay and a target network, which improve data efficiency and contribute to more stable training. For a detailed explanation of DQN, readers are referred to (Mnih et al., 2015).

The hyper-parameters used in training are as follows: a replay buffer size of 50,000, a batch size of 32, a discount factor of 0.99, a learning rate of 0.0001, a target network update frequency of 10,000 steps, and a total training duration of 1,500,000 steps.

In our reward design, the constant z in $R_{min\_max\_local}$ is set to 0.1, and the time step penalty is defined as $R_{time\_step} = -0.1 \cdot n$. These values are carefully selected to maintain a balance between achieving a reasonable average reward upon resolving each request and discouraging the agent from unnecessarily prolonging episodes.

6.5 Results and Discussion

6.5.1 Training Results

After completing the training phase, we evaluate training efficiency by examining the convergence patterns of both the reward and the episode length. To assess convergence, we summarize two key metrics as follows:

- Average Episode Reward = $\frac{\sum \text{Episode Rewards}}{\text{Number of Episodes}}$
- Average Episode Length = $\frac{\sum \text{Episode Lengths}}{\text{Number of Episodes}}$

Baseline Models

Figure 6.3 shows the convergence graphs for the baseline models, illustrating the trends of average episode reward and average episode length. All three models display clear convergence behavior, characterized by steadily increasing rewards and decreasing episode lengths.

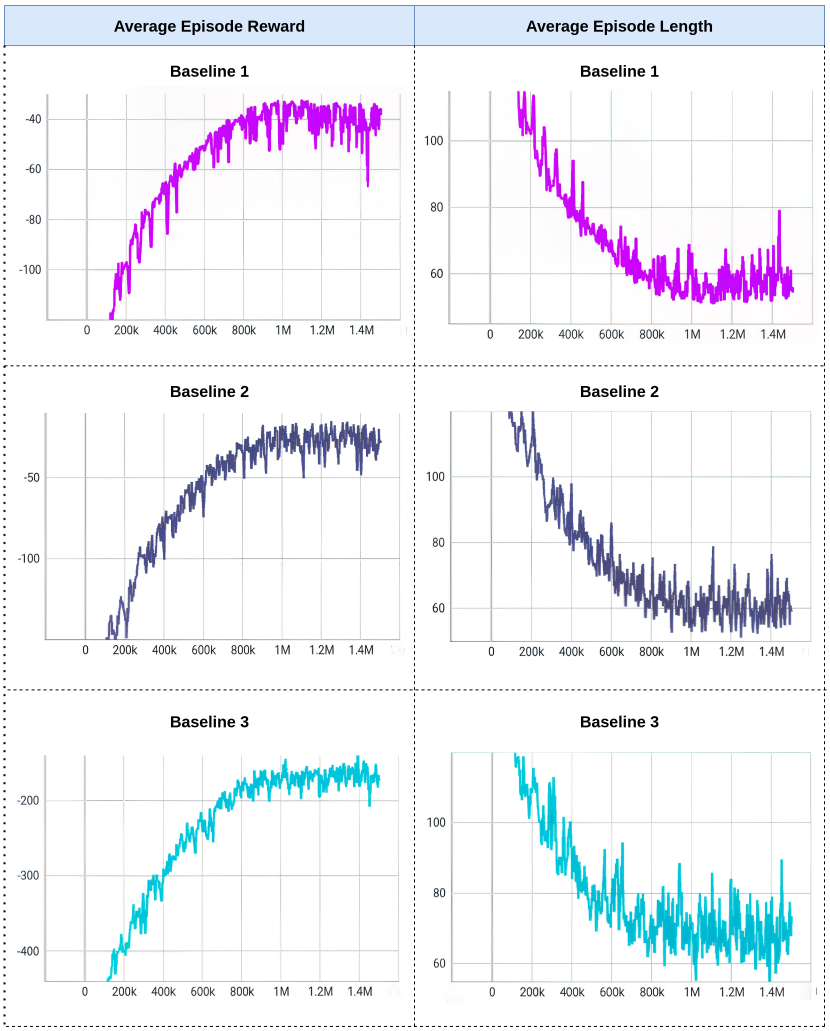


Figure 6.3: Convergence graphs of the baseline models.

Mini-max Models

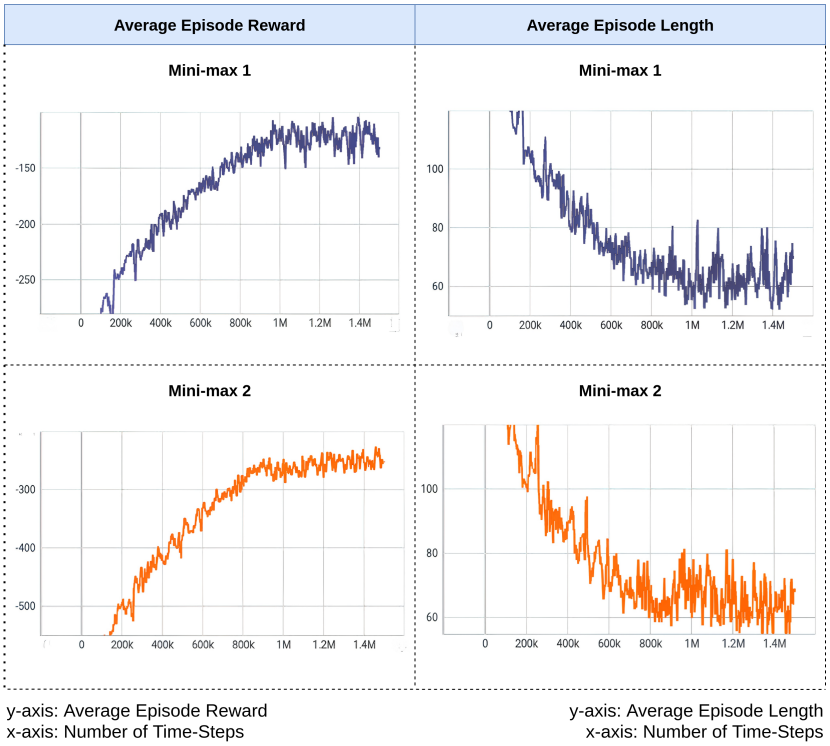


Figure 6.4: Convergence graphs of the mini-max models.

Figure 6.4 presents the convergence results for the mini-max models. Both models successfully achieve convergence, as reflected by the upward reward trends and corresponding reductions in episode lengths.

Principle-of-Proportionality Models

Figure 6.5 illustrates the convergence behavior for proportional models 1–3. Notably, model Proportional 2 fails to converge, as evidenced by unstable reward fluctuations and an increasing trend in episode lengths. This outcome suggests that excessively dense reward signals can impair training efficiency. Figure 6.6 shows the convergence plots for proportional models 4–8, all of which exhibit stable convergence patterns.

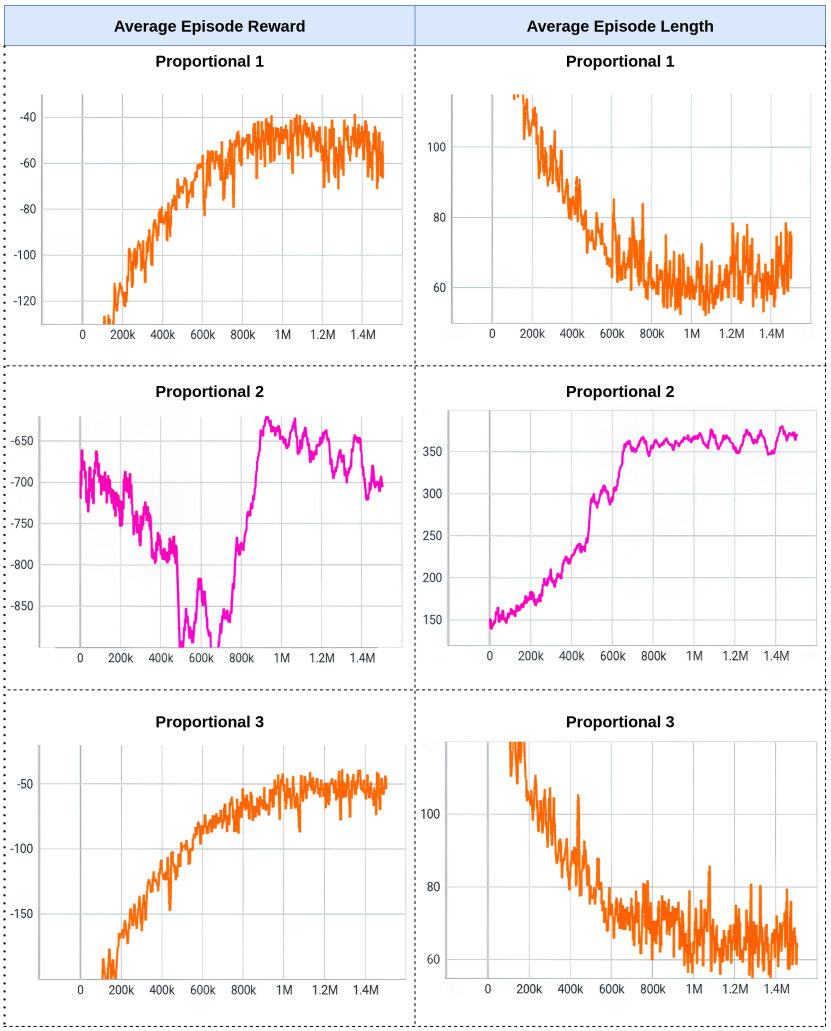


Figure 6.5: Convergence graphs of the proportional models (1, 2, 3).

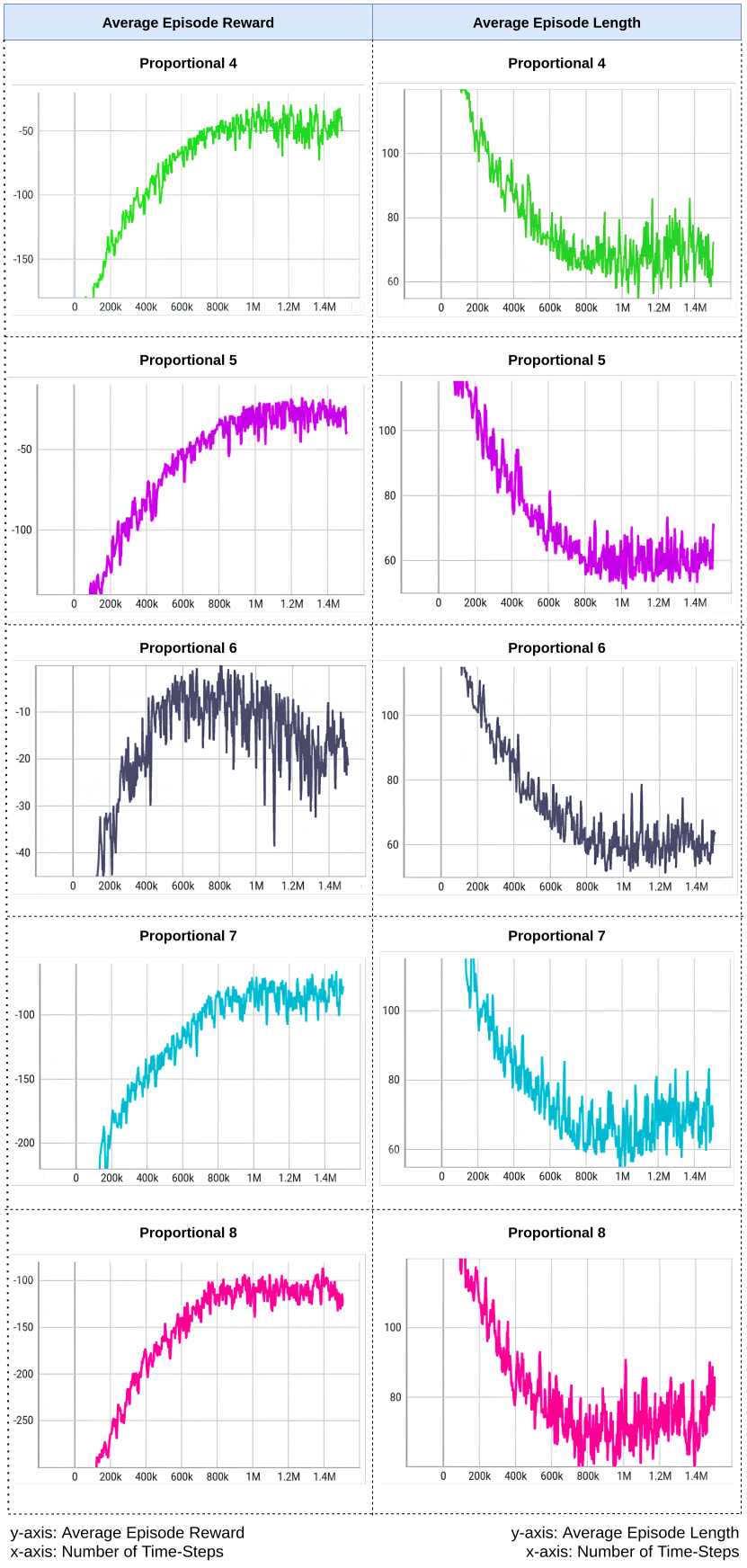


Figure 6.6: Convergence graphs of the proportional models (4, 5, 6, 7, 8).

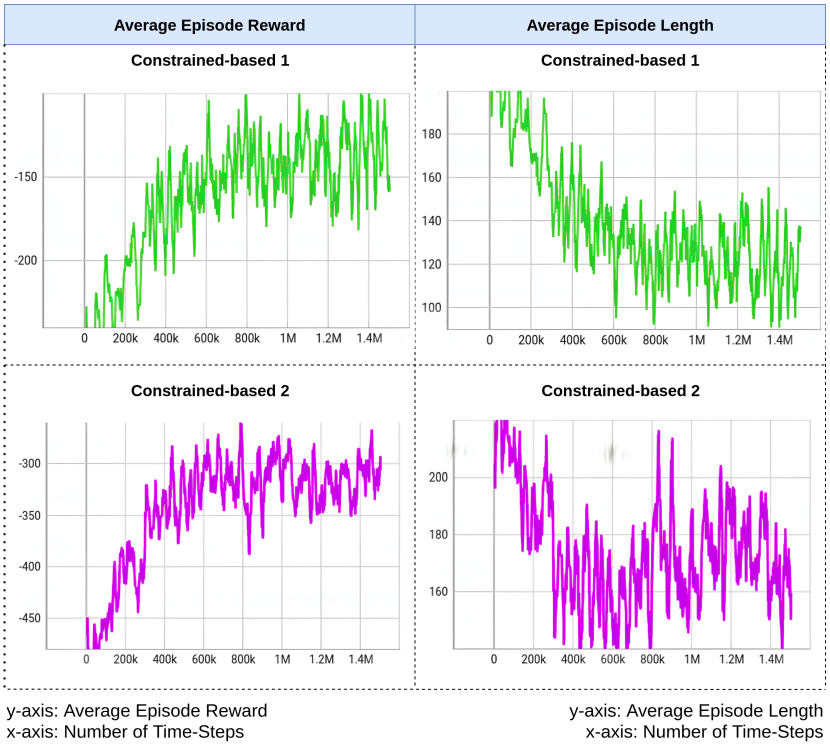


Figure 6.7: Convergence graphs of the constrained-based models.

Constraint-Based Models

Figure 6.7 displays the convergence graphs for the constraint-based models. Although both models ultimately converge, the episode lengths remain relatively unstable. This variability can be attributed to the limitation of a maximum displacement of only 30 minutes, which restricts the agent's flexibility in resolving conflicts given the constrained capacity.

In summary, with the exception of model Proportional 2, which was influenced by excessively dense reward signals, all models demonstrate successful convergence. While sparse rewards can hinder training efficiency, overly dense reward structures can likewise compromise the learning process.

6.5.2 Testing Results

We evaluate model performance using 30 testing scenarios, which are consistently applied across all models. Figure 6.8 presents the average action distribution during testing. Models such as Baseline 1, Proportional 1, Proportional 6, and Constrained-based 1 show a clear

tendency toward prioritizing fairness.

Table 6.5 further details the performance results. System efficiency is assessed through average total delay (ATD), the average number of requests solved (ARS), and the average delay per request (ADpR). Fairness is evaluated using Jain's Fairness Index (JFI), which ranges from 0 (least fair) to 1 (most fair), measuring the evenness of resource distribution, and the standard deviation (StDv) of delays.

The results indicate that models incorporating the per-time-step reward component R_1 significantly outperform those without it (e.g., Baseline 1 and Proportional 1) in terms of system efficiency. This underscores the negative impact of sparse reward signals and highlights the importance of providing agents with timely feedback at each decision step.

The performance difference between Mini-max 1 and Mini-max 2 is relatively minor, suggesting that minimizing the maximum displacement not only enhances fairness but also indirectly supports system efficiency.

Both Proportional 1 and Proportional 6 achieve high fairness scores, with Jain's Fairness Indices of 0.90. This outcome aligns with their design emphasis on fairness through higher reward weights. However, this focus comes at the cost of efficiency: Proportional 1 and Proportional 6 report total delays of 297.53 and 231.47, respectively, demonstrating the typical trade-off between fairness and efficiency.

Similarly, the Constrained-based models maintain strong fairness. However, constraining the action space to a maximum delay of 30 minutes sometimes leads to unsolvable cases, resulting in a slightly lower average number of accommodated requests in Constrained-based 2 compared to other models.

6.6 Summary

We incorporate fairness into the airport slot scheduling (allocation) problem using Reinforcement Learning by employing three approaches: the mini-max criterion, the principle of proportionality, and the constraint-based method. Our results show that excluding the per-time-step reward signals leads to inferior performance in achieving the primary goal of

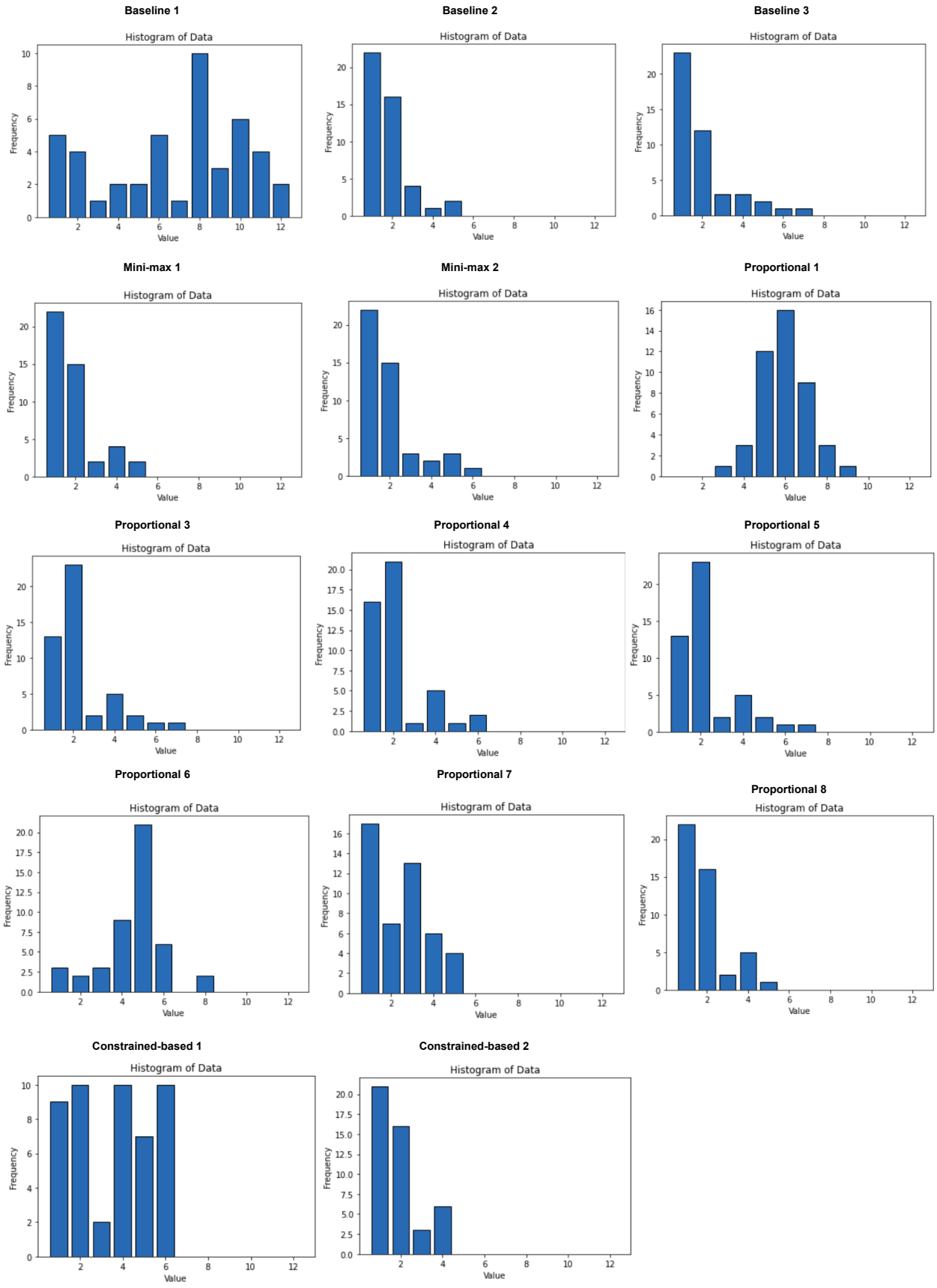


Figure 6.8: Action distribution of the models on the testing scenarios.

Model	ATD	ARS	ADpR	JFI	StDv
Baseline 1	342.93	50.67	6.81	0.76	3.79
Baseline 2	110.73	50.63	2.19	0.59	1.84
Baseline 3	124.53	50.60	2.47	0.61	2.13
Mini-max 1	114.07	50.67	2.26	0.60	1.83
Mini-max 2	116.63	50.67	2.30	0.66	1.33
Proportional 1	297.53	50.63	5.89	0.90	1.89
Proportional 3	124.07	50.57	2.46	0.73	1.46
Proportional 4	118.07	50.43	2.35	0.73	1.51
Proportional 5	140.37	50.57	2.78	0.74	1.60
Proportional 6	231.47	50.47	4.59	0.90	1.56
Proportional 7	131.03	50.60	2.59	0.66	1.61
Proportional 8	107.40	50.50	2.13	0.62	1.53
Constrained-based 1	179.40	50.50	3.57	0.76	1.85
Constrained-based 2	94.47	48.60	1.94	0.75	1.23

Table 6.5: Evaluation Results of the Models.[1]
[2]ATD: Average Total Delay; ARS: Average Requests Solved; ADpR: Average Delay per Request; JFI: Jain's Fairness Index; StDv: Standard Deviation.

maximizing system efficiency. While fairness-focused reward components, particularly those based on the principle of proportionality, are effective in maintaining fairness, they introduce a noticeable trade-off: enhancing fairness often comes at the cost of significantly reduced system efficiency.

Chapter 7

Summary

This chapter presents a comprehensive overview of the research conducted in this thesis, summarizing the key findings, identified limitations, and pathways toward real-world applications. The study developed a Reinforcement Learning (RL)-based framework for airport slot scheduling, progressively extending it from a single-airport formulation to a multi-airport coordination system and, ultimately, to a fairness-integrated framework. Each component represents a critical step toward enhancing operational efficiency, scalability, and fairness in slot allocation. Building upon the limitations identified in each stage, this chapter outlines a roadmap for translating the research outcomes into practical decision-support tools.

7.1 Research Outcomes

Figure 7.1 illustrates the three core components of this thesis, outlining the progressive development of the research. It summarizes the key research outcomes achieved in each component, identifies their corresponding limitations, and highlights the necessary improvements to bridge the gap toward real-world applications.

Component 1 formulates the single-airport slot scheduling problem as a Reinforcement Learning (RL) problem and develops a dedicated learning environment essential for agent training and evaluation. Several key factors influencing RL performance were examined, including the structure of state observations and the design of reward functions. The results

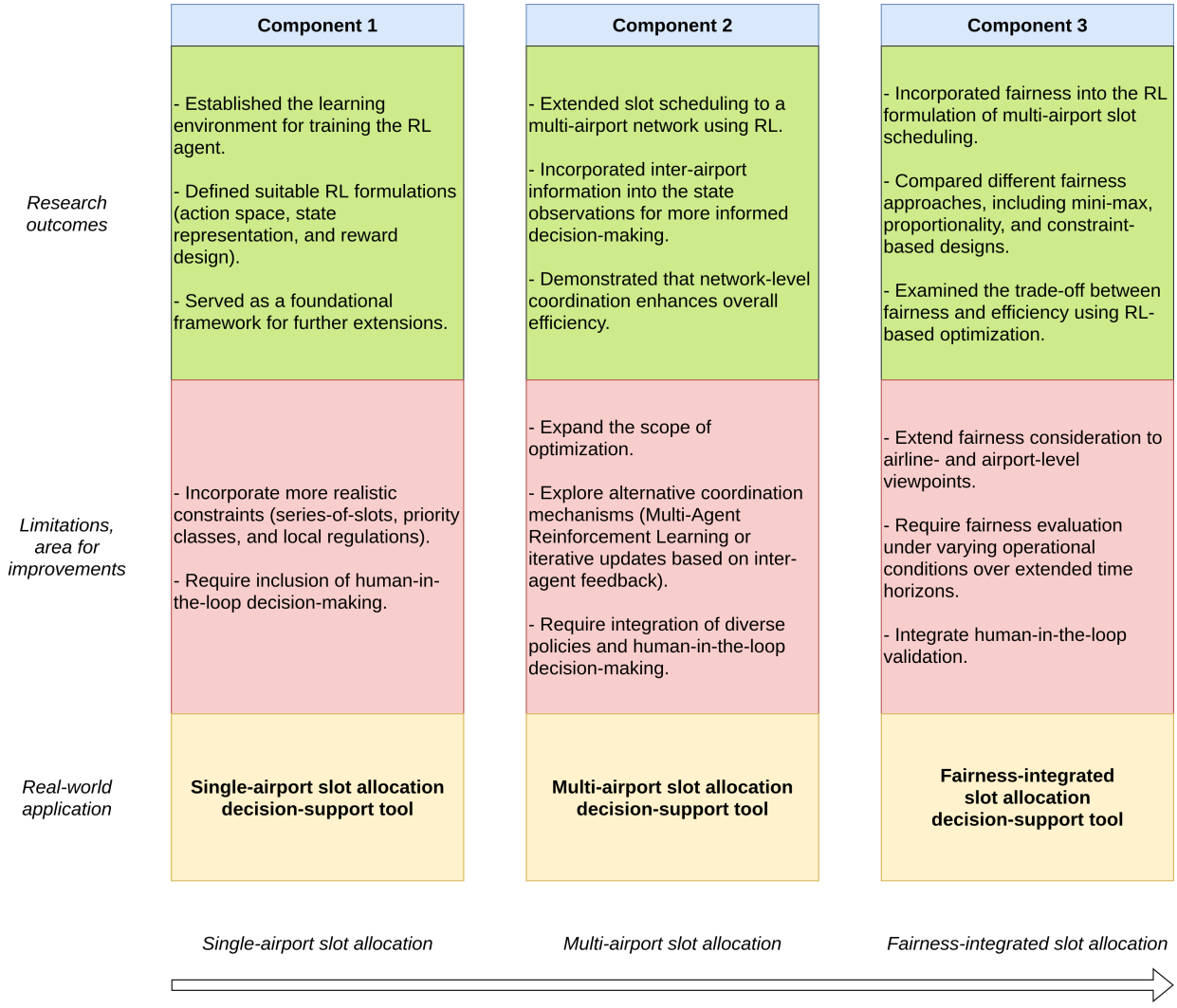


Figure 7.1: Summary of research outcomes, limitations, and real-world applications

indicate that both the compactness of the state representation and the frequency of reward feedback strongly influence the stability and efficiency of the learning process. During training, the DQN model with local state observations and positive reward feedback achieved stable convergence, while PPO exhibited slightly lower performance. In testing, DQN outperformed PPO under both medium- and high-density traffic scenarios and demonstrated robust generalization to unseen conditions. The key insight from this component is that integrating compact local observations with per-time-step solving reward signals significantly enhances training stability.

Building upon the single-airport formulation, Component 2 extended the framework to a multi-airport system, where each airport operates as an independent agent without centralized control. This decentralized framework reflects real-world conditions, particularly in regions such as ASEAN, where a central scheduling authority is absent. By incorporating inter-airport information, the results show that RL agents can leverage shared data to enhance overall system efficiency. Under medium-reduced capacity scenarios, the RL model performs comparably to the nearest-heuristic approach, whereas under heavy-reduced capacity scenarios, it significantly outperforms the heuristic, achieving lower delays and fewer unaccommodated flight requests.

Component 3 integrated fairness objectives into the RL framework by incorporating three distinct fairness concepts, the mini-max criterion, the principle of proportionality, and the constraint-based approach, within the learning process. The study systematically examined the trade-off between fairness and efficiency across these formulations. Findings show that (i) per-time-step rewards enhance convergence stability and overall efficiency, (ii) fairness-focused models achieve higher fairness levels but with a reduction in efficiency, and (iii) the proportionality and constraint-based approaches exhibit stronger fairness-oriented behavior.

7.2 Limitations

Despite demonstrating the feasibility of applying RL to single-airport slot scheduling, Component 1 remains limited in capturing the full complexity of real-world airport operations.

The current framework simplifies several critical aspects, such as the series-of-slots constraint, priority classes, and adherence to local regulatory requirements. These factors are fundamental to actual slot allocation decisions and should be incorporated to enhance the realism and applicability of the model. Moreover, the framework functions in a fully autonomous manner, without human oversight, an aspect that is particularly crucial in safety-critical industries such as aviation. Incorporating human-in-the-loop decision-making will be essential to ensure that the system complements expert judgment, maintains operational transparency, and supports practical deployment in real-world airport environments.

Building on the single-airport framework, Component 2 extended the approach to a multi-airport system, demonstrating the potential of RL for decentralized decision-making across airports but revealing several limitations for real-world application. The current framework focuses on a relatively small and simplified network, which limits the scope of optimization. Future extensions should expand this scope to include a larger number of airports, enabling more comprehensive optimization and the discovery of additional feasible solutions. Moreover, the coordination mechanism employed is relatively basic, relying on limited information exchange between agents. More advanced approaches, such as Multi-Agent Reinforcement Learning (MARL) or iterative policy updates based on inter-agent feedback, could better capture interdependencies and enhance adaptability under dynamic operational conditions. Finally, similar to Component 1, the current system functions in a fully automated manner without incorporating diverse decision-making policies or human oversight. In practice, different stakeholders, such as airlines and airports, often pursue distinct and sometimes conflicting objectives, which should be explicitly considered to ensure applicability.

Component 3 further extended the framework by introducing fairness as an additional objective. However, practical implementation challenges remain. The current formulation primarily evaluates inter-flight fairness, while inter-airline and inter-airport fairness have yet to be explored. Extending the analysis to these additional perspectives would provide a more comprehensive understanding of how fairness can be effectively incorporated into airport slot allocation. Moreover, fairness evaluation has so far been conducted under static simulation

conditions. Future research should investigate fairness performance under varying operational conditions and extended time horizons to capture dynamic and evolving system behaviors. Finally, the framework lacks human-in-the-loop validation, which is essential for ensuring that fairness-driven outcomes align with stakeholder perspectives and support consensus in real-world decision-making environments.

7.3 Real-World Applications

Based on the defined limitations, this section outlines the key elements required to develop the outcomes of this thesis into real-world applications. The chapter identifies three potential applications derived from the three core components of the research, each of which can function independently while also building progressively upon the previous one.

Single-airport slot allocation decision-support tool

Although much of the literature emphasizes network-level optimization, slot coordinators still need decision-support tools at the individual airport level, as the current IATA practice requires that slot allocation be initially conducted at each airport individually. Therefore, Component 1 can be extended toward a real-world application as a decision-support tool for slot coordinators at the single-airport level, provided that the following key elements are addressed:

- Realistic operational constraints: Incorporate series-of-slots constraints, priority classes, and local regulations to ensure compliance with actual scheduling practices.
- Human-in-the-loop decision-making: Integrate expert feedback mechanisms for decision validation. The learning environment should allow slot coordinators to adapt and refine the RL-generated solutions according to their operational needs.
- System integration: Establish connections with real operational databases, including flight schedules, slot requests, and airport capacity data, to enable seamless use in practice.

- User interface: Develop a user-friendly dashboard for slot visualization and real-time adjustments. The interface should display recommended slot allocations generated by the RL model and allow slot coordinators to either accept or modify them as needed.

Multi-airport slot allocation decision-support tool

Although Component 2 has not yet incorporated fairness considerations, it can be extended to serve as a baseline framework for multi-airport slot allocation. Rather than relying on ad hoc adjustments to non-coherent slot requests, an application developed from this component could provide optimized, data-driven solutions. Such an application would also pave the way toward a comprehensive slot allocation decision-support tool, serving both industry and research as a foundation for developing future fairness-oriented extensions. Key elements that need to be addressed for a realistic application include:

- Advanced coordination: Implement Multi-Agent Reinforcement Learning (MARL) or iterative inter-agent communication to enhance the coordination mechanism. A selective coordination strategy should also be developed to identify which agents are most relevant to each decision, thereby optimizing solution quality while reducing computational cost.

- Scalability: Once an appropriate coordination mechanism is established, it should be evaluated for scalability within a multi-airport system comprising heterogeneous airport agents with diverse objectives, constraints, and operational characteristics.

- Data integration: Incorporate real-time data streams, such as capacity disruptions and slot requests, to enable dynamic schedule updates and enhance the system's robustness under changing operational conditions.

- Human oversight: Provide human-in-the-loop decision protocols to enhance stakeholder acceptance, ensure operational transparency, and validate algorithmic recommendations before implementation.

- User interface: A user-friendly interface should be developed to allow stakeholders to input their decisions, objectives, and constraints effectively.

Fairness-integrated slot allocation decision-support tool

This application can be extended from the previous framework to incorporate fairness considerations, resulting in a comprehensive slot allocation decision-support tool. Key elements to be considered include:

- Analysis of different fairness perspectives: Incorporate inter-airline and inter-airport fairness alongside inter-flight fairness to provide a comprehensive view of equitable slot allocation.
- Stakeholder alignment: Align fairness definitions with the perspectives of airlines, airports, and regulators. Develop clear guidelines on how fairness should be addressed under different operational circumstances.
- Dynamic evaluation: Assess fairness performance under varying operational conditions and over extended time horizons to capture dynamic and evolving system behaviors.
- Human-in-the-loop validation: Conduct stakeholder-involved testing to evaluate perceived fairness, operational acceptability, and alignment with real-world decision-making processes.
- User interface: Develop a user-friendly interface that allows stakeholders to input their decisions, objectives, and constraints effectively, supporting interactive and transparent decision-making.

Bibliography

AirHelp (2022). Cost of disrupted flights to the economy. Technical report, AirHelp.

Androutsopoulos, K. N., Manousakis, E. G., and Madas, M. A. (2020). Modeling and solving a bi-objective airport slot scheduling problem. European Journal of Operational Research, 284(1):135–151.

APAC), A. (2023). Capacity and airport slots. Website.

Aviation, U. (2025). Global air transport outlook to 2032 and trends to 2042. Web Page.

Ball, M., Barnhart, C., Nemhauser, G., and Odoni, A. (2007). Air transportation: Irregular operations and control. Handbooks in operations research and management science, 14:1–67.

Benlic, U. (2018). Heuristic search for allocation of slots at network level. Transportation Research Part C: Emerging Technologies, 86:488–509.

Bertsimas, D., Farias, V. F., and Trichakis, N. (2011a). The price of fairness. Operations research, 59(1):17–31.

Bertsimas, D., Lulli, G., and Odoni, A. (2011b). An integer optimization approach to large-scale air traffic flow management. Operations research, 59(1):211–227.

CAAS (2021). Aip supplement 049/2021: Singapore changi airport – closure of runway 02c/20c and taxiways due to changi east development works. https://aim-sg.caas.gov.sg/aim-content/uploads/aip/2025-07-24/final/2021-04-08/html/eSUP/eSUP-2021-049-en-GB.html.

Castelli, L., Pellegrini, P., and Pesenti, R. (2012). Airport slot allocation in europe: economic efficiency and fairness. International journal of revenue management, 6(1-2):28–44.

Changi (2022). Changi airport statistics.

Chen, Y., Xu, Y., Hu, M., and Yang, L. (2021). Demand and capacity balancing technology based on multi-agent reinforcement learning. In 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), pages 1–9. IEEE.

Corolli, L., Lulli, G., and Ntaimo, L. (2014). The time slot allocation problem under uncertain capacity. Transportation Research Part C: Emerging Technologies, 46:16–29.

Corrigan, S., Mårtensson, L., Kay, A., Okwir, S., Ulfvengren, P., and McDonald, N. (2015). Preparing for airport collaborative decision making (a-cdm) implementation: an evaluation and recommendations. Cognition, Technology & Work, 17:207–218.

Cui, D., Ke, W., Peng, Z., and Zuo, J. (2015). Multiple dags workflow scheduling algorithm based on reinforcement learning in cloud computing. In International Symposium on Computational Intelligence and Intelligent Systems, pages 305–311. Springer.

Dayal, A., Cenkeramaddi, L. R., and Jha, A. (2022). Reward criteria impact on the performance of reinforcement learning agent for autonomous navigation. Applied Soft Computing, 126:109241.

Donohue, G. L. and III, R. D. S. (2008). Terminal chaos: Why US air travel is broken and how to fix it. American Institute of Aeronautics and Astronautics.

Duong, T., Todi, K. K., Chaudhary, U., and Truong, H.-L. (2019). Decentralizing air traffic flow management with blockchain-based reinforcement learning. In 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), volume 1, pages 1795–1800. Ieee.

EASA (2022). European aviation environmental report 2022. Technical report, EASA.

Eurocontrol (2014). Coda digest: Delays to air transport in europe (annual 2013). report prepared by eurocontrol's central office for delay analysis (coda). Technical report, Eurocontrol.

EUROCONTROL (2023). Airport collaborative decision making (a-cdm). https://www.eurocontrol.int/concept/airport-collaborative-decision-making.

EUROCONTROL (2023). Coda digest: All-causes delays to air transport in europe quarter 1 2023. Technical report, EUROCONTROL.

EUROCONTROL (2024). Comparison of air traffic management related operational and economic performance: U.s. — europe. Report (PDF).

FAA (2025). Slot administration office. Website.

Fairbrother, J., Zografos, K. G., and Glazebrook, K. D. (2020). A slot-scheduling mechanism at congested airports that incorporates efficiency, fairness, and airline preferences. Transportation Science, 54(1):115–138.

Fernández, E. C., Cordero, J. M., Vouros, G., Pelekis, N., Kravaris, T., Georgiou, H., Fuchs, G., Andrienko, N., Andrienko, G., Casado, E., et al. (2017). Dart: A machine-learning approach to trajectory prediction and demand-capacity balancing. SESAR Innovation Days, Belgrade, pages 28–30.

Garey, M. R. (1979). A guide to the theory of np-completeness. Computers and intractability.

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

IATA (2014). Fact sheet: Single european sky (ses).

IATA (2015). Worldwide Slot Guidelines, 7th ed. http://qatarcoordination.org/pdf/wsg-7.pdf.

IATA (2021). Outlook for the global airline industry april 2021 update.

IATA (2022a). List of level 2 and level 3 airports.

IATA (2022b). Worldwide airport slots, fact sheet. https://www.iata.org/en/iata-repository/pressroom/fact-sheets/fact-sheet---airport-slots/.

IATA (2023). Wasg annex 12.7: Contact list for level 2 and level 3 airports. Excel file, IATA content assets.

IATA (2024). Worldwide airport slot guidelines (wasg).

IATA (2024). Worldwide airport slot guidelines (wasg) — edition 3. https://www.iata.org/en/programs/ops-infra/slots/slot-guidelines/wasg-edition-3-english-version.pdf.

ICAO (2007). Icao long term traffic forecasts.

ICAO (2022). Asia/pacific seamless air navigation services (ans) plan version 3.0. https://www.icao.int/sites/default/files/APAC/Documents/Asia-Pacific-Seamless-ANS-Plan-Version-3.0.pdf.

Jacquillat, A. and Odoni, A. R. (2015). An integrated scheduling and operations approach to airport congestion mitigation. Operations Research, 63(6):1390–1410.

Jacquillat, A. and Vaze, V. (2018). Interairline equity in airport scheduling interventions. Transportation Science, 52(4):941–964.

Jiang, J. and Lu, Z. (2019). Learning fairness in multi-agent systems. Advances in Neural Information Processing Systems, 32.

Jiang, Y. and Zografos, K. G. (2021). A decision making framework for incorporating fairness in allocating slots at capacity-constrained airports. Transportation Research Part C: Emerging Technologies, 126:103039.

Jones, I., Holder, S., et al. (2004). Study to assess the effects of different slot allocation schemes: a report for the european commission,'. NERA Economic Consulting.

Kargar, K., Zografos, K., and Mouzas, S. (2024). A mechanism for bilateral airport slot trading in the secondary market. Available at SSRN 4878947.

Katsigiannis, F. and Zografos, K. (2024). A multi-objective genetic algorithm for airport slot allocation decision-making. SSRN Electronic Journal. Available at SSRN: https://ssrn.com/abstract=4742975, DOI:10.2139/ssrn.4742975.

Katsigiannis, F. A. and Zografos, K. G. (2021). Optimising airport slot allocation considering flight-scheduling flexibility and total airport capacity constraints. Transportation Research Part B: Methodological, 146:50–87.

Keskina, M. and Zografos, K. G. (2022). A modelling framework for solving the network-wide airport slot allocation problem.

Khadilkar, H. (2019). A scalable reinforcement learning algorithm for scheduling railway lines. IEEE Transactions on Intelligent Transportation Systems, 20(2):727–736.

Koesters, D. (2007). Study on the usage of declared capacity at major german airports. Study in cooperation with Eurocontrol's Performance Review Unit (PRU), Aachen, Germany.

Kravaris, T., Spatharis, C., Bastas, A., Vouros, G. A., Blekas, K., Andrienko, G., Andrienko, N., and Garcia, J. M. C. (2019). Resolving congestions in the air traffic management domain via multiagent reinforcement learning methods. arXiv preprint arXiv:1912.06860.

Kravaris, T., Spatharis, C., Blekas, K., Vouros, G. A., and Garcia, J. M. C. (2018). Multi-agent reinforcement learning methods for resolving demand-capacity imbalances. In 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), pages 1–10. IEEE.

Kravaris, T., Vouros, G. A., Spatharis, C., Blekas, K., Chalkiadakis, G., and Garcia, J. M. C. (2017). Learning policies for resolving demand-capacity imbalances during pre-tactical air traffic management. In German Conference on Multiagent System Technologies, pages 238–255. Springer.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Applied Soft Computing, 91:106208.

MacDonald, M. (2006). Study on the impact of the introduction of secondary trading at community airports. Report commissioned by the European Commission.

Madas, M. A. and Zografos, K. G. (2006). Airport slot allocation: From instruments to strategies. Journal of Air Transport Management, 12(2):53–62.

Manley, B. and Sherry, L. (2010). Analysis of performance and equity in ground delay programs. Transportation Research Part C: Emerging Technologies, 18(6):910–920.

Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. Computers & Operations Research, 134:105400.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. nature, 518(7540):529–533.

OAG (2025). Oag — flight database & aviation analytics. https://www.oag.com/.

Odoni, A. and Morisset, T. (2010). Performance comparisons between us and european airports. In Proceedings of the 12th world conference on transport research (WCTR), July 11–15.

Palombarini, J. A. and Martínez, E. C. (2019). Closed-loop rescheduling using deep reinforcement learning. IFAC-PapersOnLine, 52(1):231–236.

Pellegrini, P., Bolić, T., Castelli, L., and Pesenti, R. (2017). Sosta: An effective model for the simultaneous optimisation of airport slot allocation. Transportation Research Part E: Logistics and Transportation Review, 99:34–53.

Pellegrini, P., Castelli, L., and Pesenti, R. (2012a). Metaheuristic algorithms for the simultaneous slot allocation problem. IET Intelligent Transport Systems, 6(4):453–462.

Pellegrini, P., Castelli, L., and Pesenti, R. (2012b). Secondary trading of airport slots as a combinatorial exchange. Transportation Research Part E: Logistics and Transportation Review, 48(5):1009–1022.

Pyrgiotis, N. and Odoni, A. (2016). On the impact of scheduling limits: A case study at newark liberty international airport. Transportation Science, 50(1):150–165.

Ranieri, A., Alsina, N., Castelli, L., Bolic, T., and Herranz, R. (2013). Airport slot allocation: performance of the current system and options for reform. 3rd SESAR Innovation Days (SIDs).

Rawls, J. (1971). A theory of justice. 2. The Law of Peoples, 67.

Ribeiro, N. A., Jacquillat, A., and Antunes, A. P. (2019a). A large-scale neighborhood search approach to airport slot allocation. Transportation Science, 53(6):1772–1797.

Ribeiro, N. A., Jacquillat, A., Antunes, A. P., and Odoni, A. (2019b). Improving slot allocation at level 3 airports. Transportation Research Part A: Policy and Practice, 127:32–54.

Ribeiro, N. A., Jacquillat, A., Antunes, A. P., Odoni, A. R., and Pita, J. P. (2018). An optimization approach for airport slot allocation under iata guidelines. Transportation Research Part B: Methodological, 112:132–156.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Sheikh, H. U. and Bölöni, L. (2020). Multi-agent reinforcement learning for problems with combined individual and team reward. In 2020 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE.

Shone, R., Glazebrook, K., and Zografos, K. G. (2021). Applications of stochastic modeling in air traffic management: Methods, challenges and opportunities for solving air traffic problems under uncertainty. European Journal of Operational Research, 292(1):1–26.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. nature, 550(7676):354–359.

Spatharis, C., Bastas, A., Kravaris, T., Blekas, K., Vouros, G. A., and Cordero, J. M. (2021). Hierarchical multiagent reinforcement learning schemes for air traffic management. Neural Computing and Applications, pages 1–13.

Spatharis, C., Blekas, K., Bastas, A., Kravaris, T., and Vouros, G. A. (2019). Collaborative multiagent reinforcement learning schemes for air traffic management. In 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), pages 1–8. IEEE.

Sutton, R. S. and Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Tang, Y. and Xu, Y. (2021). Multi-agent deep reinforcement learning for solving large-scale air traffic flow management problem: A time-step sequential decision approach. In 2021 IEEE/AIAA 40th digital avionics systems conference (DASC), pages 1–10. IEEE.

Wang, C., Yang, L., Hu, M., Wang, Y., and Zhao, Z. (2024). On-demand airport slot management: tree-structured capacity profile and coadapted fire-break setting and slot allocation. Transportmetrica A: Transport Science, pages 1–35.

Wang, S., Drake, J. H., Fairbrother, J., and Woodward, J. R. (2019a). A constructive heuristic approach for single airport slot allocation problems. In 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1171–1178. IEEE.

Wang, Y., Liu, H., Zheng, W., Xia, Y., Li, Y., Chen, P., Guo, K., and Xie, H. (2019b). Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning. IEEE access, 7:39974–39982.

Wang, Y.-F. (2020). Adaptive job shop scheduling strategy based on weighted q-learning algorithm. Journal of Intelligent Manufacturing, 31(2):417–432.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., and Kyek, A. (2018). Optimization of global production scheduling with deep reinforcement learning. Procedia Cirp, 72:1264–1269.

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

Yutong, C., Minghua, H., Yan, X., and Lei, Y. (2023). Locally generalised multi-agent reinforcement learning for demand and capacity balancing with customised neural networks. Chinese Journal of Aeronautics, 36(4):338–353.

Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., and Chi, X. (2020a). Learning to dispatch for job shop scheduling via deep reinforcement learning. Advances in Neural Information Processing Systems, 33:1621–1632.

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

Zografos, K. and Jiang, Y. (2016). Modelling and solving the airport slot scheduling problem with efficiency, fairness, and accessibility considerations. In TRISTAN Symposium, pages 13–17.

Zografos, K. G., Androutsopoulos, K. N., and Madas, M. A. (2018). Minding the gap: Optimizing airport schedule displacement and acceptability. Transportation Research Part A: Policy and Practice, 114:203–221.

Zografos, K. G. and Jiang, Y. (2019). A bi-objective efficiency-fairness model for scheduling slots at congested airports. Transportation Research Part C: Emerging Technologies, 102:336–350.

Zografos, K. G., Madas, M. A., and Androutsopoulos, K. N. (2017). Increasing airport capacity utilisation through optimum slot scheduling: review of current developments and identification of future needs. Journal of Scheduling, 20(1):3–24.

Zografos, K. G., Salouras, Y., and Madas, M. A. (2012). Dealing with the efficient allocation of scarce resources at congested airports. Transportation Research Part C: Emerging Technologies, 21(1):244–256.