Quantifying the Value of Information and Collaboration in the Decentralized ASEAN Airspace Network

Submitted by

Darryl TEO

Thesis Advisor

Dr. Nuno RIBEIRO
Assistant Professor, Engineering Systems and Design
Deputy Director, Aviation Studies Institute

Engineering Systems and Design

A thesis submitted to the Singapore University of Technology and Design in fulfillment of the requirement for the degree of Doctor of Philosophy

2025

PhD Thesis Examination Committee

TEC Chair:	Karthik Natarajan Professor — Pillar of Engineering Systems and Design
Main Advisor:	Nuno Ribeiro Assistant Professor — Pillar of Engineering Systems and Design Deputy Director — Aviation Studies Institute
Co-Advisor:	Peter Jackson Professor and Former Head of Pillar — Pillar of Engineering Systems and Design Distinguished Visiting Professor — Aviation Studies Institute
Internal TEC member 1:	Antonios Varvitsiotis Assistant Professor — Pillar of Engineering Systems and Design
Internal TEC member 2:	Karthyek Murthy Assistant Professor — Pillar of Engineering Systems and Design

Abstract

Engineering Systems and Design

Doctor of Philosophy

Quantifying the Value of Information and Collaboration in the Decentralized ASEAN Airspace Network

by Darryl TEO

Reimagining Air Traffic Flow Management (ATFM) — toward a smarter, more resilient future. While information sharing and collaborative decision-making are well-established in regions such as the United States and Europe, Southeast Asia continues to operate in a highly decentralized manner, posing significant challenges to the effective implementation of ATFM. A major undertaking to support a transition into a collaborative environment is the Flight & Flow Information for a Collaborative Environment (FF-ICE) initiative, conceived to eliminate the limitations of the current Flight Plan system.

This dissertation presents the first comprehensive quantitative study of the FF-ICE program, examining the benefits and drawbacks of an incremental FF-ICE R1 implementation in the ASEAN region. Owing to the interconnectivity, complexity, uncertainty, and decentralized nature of the ASEAN airspace, solving the ATFM problem at the scale of the entire region fails to capture the operational nuances, particularly because incremental participation is anticipated. To address this, we introduce a Rolling Horizon Concept (RHC) that more accurately represents varying levels of information sharing and aligns more closely with current operational capabilities.

We present a suite of mathematical models and algorithmic routines designed to simulate decentralized ATFM systems, incorporating key constraints such as airspace capacity and aircraft separation requirements. These algorithms operate independently within each region, using the available information at each time step to optimize ATFM decisions including speed adjustments, ground delay programs and airborne holding.

A realistic simulator of the ASEAN airspace network, supporting information sharing and collaborative decision-making, was developed using the proposed mathematical tools and real airspace data. The results demonstrate significant savings under various collaboration regimes. Notably, partial collaboration, even within a single country, can yield considerable benefits. The results also reveal a surprising consequence of minor collaborations, say between only two airports, that has been shown to erratically impact other flights far from the time which collaborative measures take place.

Publications

Darryl Teo, Peter Jackson, Daniel Delahaye, and Nuno Ribeiro. "Optimization-Based Simulation of Air Traffic Flow Management Within the Decentralized ASEAN Region" In: *Computers & Operations Research (Major Revision). Preprint available at: ssrn*. Supporting paper for Chapters 4 and 5.

Darryl Teo, Peter Jackson, Nuno Ribeiro, Rakesh Nandi, and Gengling Dai. "Estimating the Benefits of Information Sharing and Collaboration in a Decentralized Flight Network" In: *Working paper*. Supporting paper for Chapters 6 and 7.

Presentations

Darryl Teo, Peter Jackson, and Daniel Delahaye. "Measuring the Value of Ground Delay Programs in the ASEAN Airspace" In: *Presentation at the Singapore Aviation Research Technology Exchange Day (2025)*

Darryl Teo, Peter Jackson, Nuno Ribeiro, Gengling Dai, and Rakesh Nandi. "Quantifying Airspace Network Benefits and Impacts on Non-Participating Flights Under the FF-ICE R1 Initiative" In: *Presentation at the 2025 INFORMS International Meeting (2025)*

Darryl Teo, Peter Jackson, Nuno Ribeiro, Gengling Dai, and Rakesh Nandi. "Simulated Impact of the FF-ICE R1 Initiative Under Multiple Capacity Constraints and Collaboration" In: *Presentation at the 28th ATRS World Conference (2025)*

Darryl Teo, Peter Jackson, and Daniel Delahaye. "Optimization Algorithms Applied to the ASEAN Airspace" In: *Presentation at the NUS SG Analytics and Operations Day 2024 (2024)*

Darryl Teo, Peter Jackson, Nuno Ribeiro, Gengling Dai, and Rakesh Nandi. "Simulated Impact of the FF-ICE R1 Initiative With Incremental Participation" In: *Presentation at the INFORMS 2024 Annual Meeting (2024)*

Acknowledgements

First and foremost, I would like to thank Prof PJ for believing in me, and giving me this amazing opportunity to pursue the PhD. To Prof PJ: From day one until now, I could not have asked for a better mentor, who exhibits the values of care, integrity, industriousness, friendliness and beyond. The way you treat everyone with respect, regardless of social status, has left a deep impression on me, and I strive to follow in your footsteps. Also, no matter how busy you were, you always found time to respond and help me with my tasks or personal matters, and I am extremely grateful for your generosity with time. I have grown so much in the past few years thanks to your mentorship, developing as a better researcher, as well as a better human, and I endeavor to emulate your example! Wishing you and your family good health and happiness.

Next, I would like to thank Prof Nuno for advising my PhD for the past two years, and always offering advice and help whenever I needed it. I thank you for the knowledge you have imparted, and guidance you have generously offered over the many years we have known each other. To Prof Nuno: I am continually impressed by your ability to analyze data and offer insights that few others can. Your ability to create convincing arguments is truly admirable. I am deeply grateful to have learned from you —– your mentorship has greatly boosted my confidence in presenting and articulating my ideas. Furthermore, during our conversations, I feel as if I am talking to another Singaporean; your ability and adapt and thrive in new surroundings is a powerful and enviable trait. The stories of your adventures overseas have never failed to captivate me and opened my eyes to how big the world is. Thank you so much for your guidance and I look forward to future collaborations together.

I would like to thank Prof Daniel for hosting me at Ecole Nationale de l'Aviation Civile. To Prof Daniel: Thank you for taking care of me during my stay in Toulouse, France, and beyond. It was my first time living abroad, and while scary at first, my fears dissipated soon after, in no small part due to your warmth and kindness. Your vast knowledge on aviation and reputation in the field has left me starstruck, and I am proud to have you as my mentor. In addition, your physical ability is extremely impressive — the intensity of your cycles and mountain climbing, and frequent travels, has become a tale of legends in my book. Thank you for dropping by whenever you visit Singapore too, I am always happy to see you again. Wishing you good health, and safe travels! Merci beaucoup!

I also wish to express my heartfelt gratitude to Chen Sin Chee, for being the best graduate studies manager. He has always gone above and beyond to help me whenever I face any difficulty in any administrative or personal matters. Thank you to Prof Jeffery Chan for the interesting discussions during the ethics class, and encouraging me

to enroll into the PhD program. Thank you to Cecilia Radhalechumi, Samantha Septribellina, Lim Lee Chen, and Tan Siew Gan for always helping to schedule meetings and other admin matters; there has been 0 hiccups as far as I recall, amazing! Thank you to Prof Bikramjit Das and Prof Stefano Galelli for providing guidance during my term as your teaching assistant. Thank you to Li Hongbo and Ruan Yanqiu for the teamwork and support as friends and groupmates during the PhD coursework. Thank you to my project mates over the past few years — Gengling Dai, Rakesh Nandi, Wang Weimeng, Ho Huiqi, Chua Wanyun, Faye Chiang; you have been the most amazing people to work with. Thank you to everyone in the ESD office for the welcoming environment, especially during my undergrad days. Thank you to everyone in the ASI office for making it an enjoyable place to work. Thank you to the PhD matters office for the smooth processing of all the overseas attachments and conferences I have attended. I would like to extend my sincere thanks to everyone I had the pleasure of interacting with during my time at SUTD. My experience has left me with a lasting impression of SUTD as a highly efficient research institution that offers world-class education and is home to exceptionally capable and welcoming staff and students.

I would also like to thank SUTD for the generous funding provided under the PGF(CIS) scholarship, and CAAS for funding many projects within the ASI.

Last but not least, I would like to thank God, and my family. I thank the Lord for blessing me with this opportunity, and keeping me from harm; I will seek your ways in all seasons of life. *The heart of man plans his way, but the LORD establishes his steps. -Proverbs 16:9*. To my family: Home is where the heart is, and my heart lies with you. Thank you for being there for me, supporting and encouraging me always; the love and warmth of our family is irreplaceable. My mother's favorite quote: "A watermelon in a box will end up square.". Given that the shape of a watermelon is determined by its surroundings, I hope to pursue the next chapter of my life in an environment that positively shapes my personal and professional life and do you proud.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

As our world becomes increasingly globalized, international travel, trade, and other business activities have continued to fuel growth in the extensive industry that is aviation. For a sense of scale, the aviation sectors' contribution as a percent of GDP, including tourism, ranges from 3.2% of Canada's GDP, to 11.8% of Singapore's GDP, and for a more extreme example, 58.8% of Maldive's GDP (IATA, 2020).

Despite the meltdown in aviation services during the COVID-19 pandemic, the recovery has been strong, with flight revenues at the end of 2023 exceeding pre-pandemic levels by seven percent, as shown in Figure 1.1. This was led by exceptionally strong growth in passenger revenue, especially in the Asia-Pacific region, which accounted for more than half of the global increase.

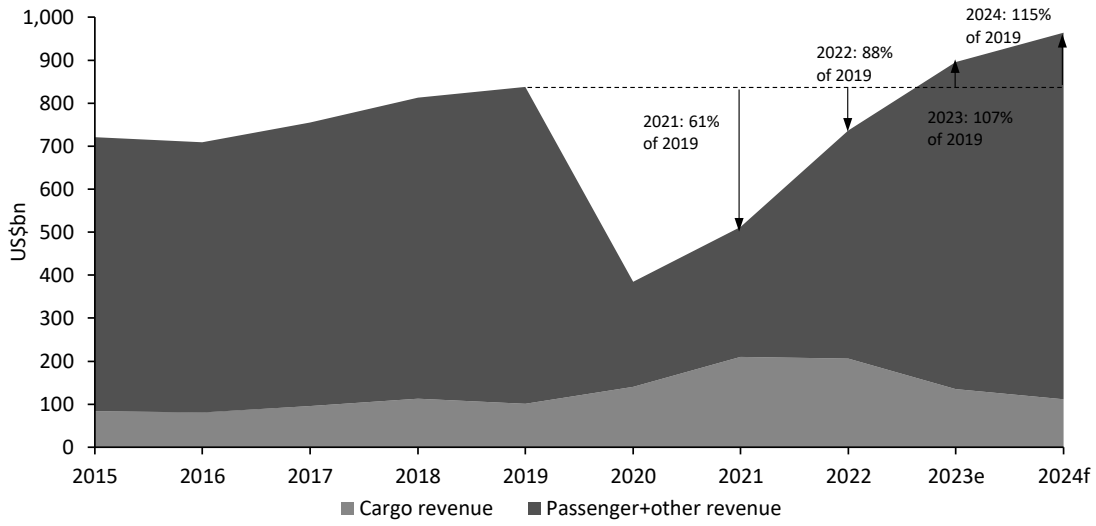

FIGURE 1.1: IATA Chart of the Week: Airline Revenue to Surpass Pre-Pandemic Levels in 2023 (IATA, 2023b).

With air travel demand projected to continue growing at a Compounded Annual Growth Rate (CAGR) of 4.3% from 2015 to 2035 (ICAO, 2019), the question whether capacity can keep up with demand becomes an increasingly pressing concern. A EUROCONTROL study (EUROCONTROL, 2018) forecasts a 63% rise in average delays

and a 167% increase in the number of congested airports by 2040, underscoring the urgent need for measures to address the growing demand. Airspace congestion leads to three major issues: delays, safety risks, and negative environmental impacts. These problems, in turn, lead to increased costs for all stakeholders including passengers, airlines, and airports. A study done by the Federal Aviation Administration (FAA) estimated delays to cost a total of 33.0 billion USD in 2019, with passengers bearing the brunt of the cost due to time lost from schedule buffers, delayed flights, flight cancellations, and missed connections (FAA, 2019). With regard to safety risks stemming from a reduction of separation requirements, multiple studies including (Blom and Bakker, 2015) and (Ye, M. Hu, and Shortle, 2014) have brought attention to the risk-capacity trade off in airspace management. Fortunately, the results of the studies show that, for various airspace operations, the risk of a loss of separation is very low even with an increase in the current airspace traffic. Such results reflect the rarity of midair collisions due to a loss of separation, with zero such accidents recorded in 2023 (ICAO, 2023a), and less than ten in the past two decades.

In general, it is surmised that there are two primary ways to increase capacity. The first approach involves increasing airspace capacity by expanding physical infrastructure, such as constructing new airports or runways, and by increasing the number of available airways. A notable example of the impact of opening airways can be traced back to the 1990s, where during the Cold War, no flights were allowed over the Soviet airspace, where even accidental entries into the airspace had been shot down (ASN, 2024). Due to this, aircraft had to take alternative routes such as shown in Figure 1.2, stopping at Anchorage before continuing on their journey. After the Cold War, the airways were open once again, allowing flights to take a shorter route, as shown in Figure 1.3, increasing airspace capacity significantly. While physically increasing capacity is effective, the political complexities of opening new airways, the significant investments required in opening new airports, and longer implementation time horizons, are factors that have contributed to the increasing emphasis on research in airspace operations. Through better utilization of resources and optimization of airspace operations, airspace capacity has undergone recent improvements, with lower costs, and more immediate impacts. Some examples include planning for more optimal runway schedules, slot allocations and ground delay programs, to name a few. We note that both these methods to increase capacity are not mutually exclusive, but rather, are being developed simultaneously and have a symbiotic relationship. For example, research in aircraft routing involves more efficient utilization of the current airspace, and at the same time, may provide a quantification of the extent to which airspace users benefit from the opening of new airways. The outcome of some aviation research projects,

has culminated in the implementation of various airspace initiatives including Functional Airspace Blocks (FABs) in Europe and Next Generation Air Transportation System (NextGen) in the USA. We reserve a review of the main research areas in Section 2.2.

FIGURE 1.2: Flight Routes Under a Restricted Airspace, During the Cold War (SCMP, 2018)

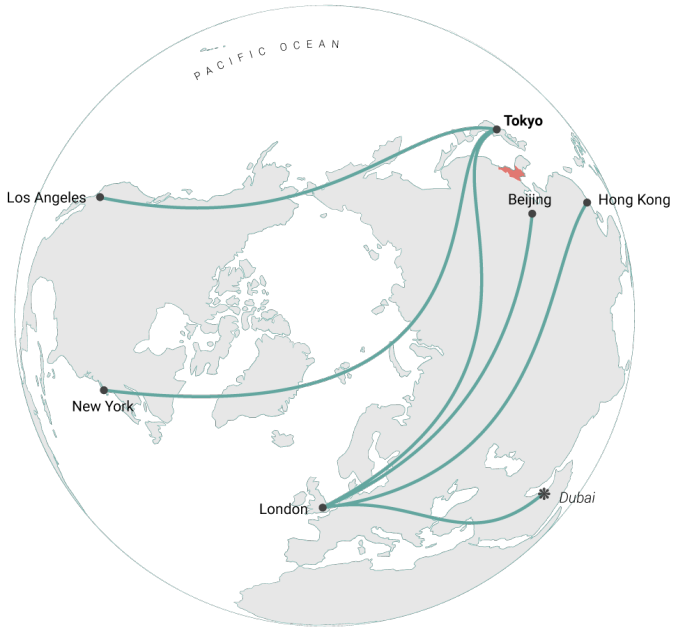

FIGURE 1.3: Flight Routes in an Open Airspace, After the Cold War (SCMP, 2018)

1.1 Air Traffic Management and Collaboration Initiatives

This section describes systems that work together to improve Air Traffic Management (ATM) operations, with a focus on systems that are built upon the basis of collaboration between stakeholders. Other closely related initiatives that are not elaborated on here are NextGen, Single European Sky ATM Research (SESAR).

1.1.1 Global Air Traffic Management Operational Concept

The Global Air Traffic Management Operational Concept (GATMOC) was introduced by the International Civil Aviation Organization (ICAO) in 2005 with the aim of achieving an interoperable global air traffic management system, for all users during all phases of flight, that meets the agreed safety levels, provides for optimum economic operations, is environmentally sustainable and meets national security requirements (ICAO, 2005). The GATMOC listed seven guiding principles, as given in Figure 1.4.

1.1.2 Collaborative Decision-Making

In 2023, a study done by IATA (IATA, 2023a) indicated that 60.1% of flights are international. The global nature of air traffic requires international collaboration in planning, implementation and operation. Due to a lack of accurate and precise flight information, Air Traffic Flow Management (ATFM), especially for international flights, experience a

Guiding Principles

Safety. The attainment of a safe system is the highest priority in air traffic management, and a comprehensive process for safety management is implemented that enables the ATM community to achieve efficient and effective outcomes.

Humans. Humans will play an essential and, where necessary, central role in the global ATM system. Humans are responsible for managing the system, monitoring its performance and intervening, when necessary, to ensure the desired system outcome. Due consideration to human factors must be given in all aspects of the system.

Technology. The ATM operational concept addresses the functions needed for ATM without reference to any specific technology and is open to new technology. Surveillance, navigation and communications systems, and advanced information management technology are used to functionally combine the ground-based and airborne system elements into a fully integrated, interoperable and robust ATM system. This allows flexibility across regions, homogeneous areas or major traffic flows to meet the requirements of the concept.

Information. The ATM community will depend extensively on the provision of timely, relevant, accurate, accredited and quality-assured information to collaborate and make informed decisions. Sharing information on a system-wide basis will allow the ATM community to conduct its business and operations in a safe and efficient manner.

Collaboration. The ATM system is characterized by strategic and tactical collaboration in which the appropriate members of the ATM community participate in the definition of the types and levels of service. Equally important, the ATM community collaborates to maximize system efficiency by sharing information, leading to dynamic and flexible decision making.

Continuity. The realization of the concept requires contingency measures to provide maximum continuity of service in the face of major outages, natural disasters, civil unrest, security threats or other unusual circumstances.

FIGURE 1.4: Guiding Principles for GATMOC (ICAO, 2005)

lack of predictability and stability. In Europe, such airspaces have been shown to suffer from a disproportionately large number of incorrect and missing flight plans, necessitating a capacity buffer to account for unforeseen changes in airspace demand and capacity (ICAO, 2018). As such, programs and initiatives that promote collaborative decision-making are projected to increase the effectiveness and efficiency of the global airspace to meet the increasing demand for air travel.

As of present, information sharing is supported by systems including the System Wide Information Management (SWIM) and Airport Operations Centre System. The primary sources of shared information, ideally as updated and precise as possible, are flight schedules, flight cancellations, operational constraints and route availability. Figure 1.5 illustrates this, along with the inclusion of other vital information that support ATFM operations.

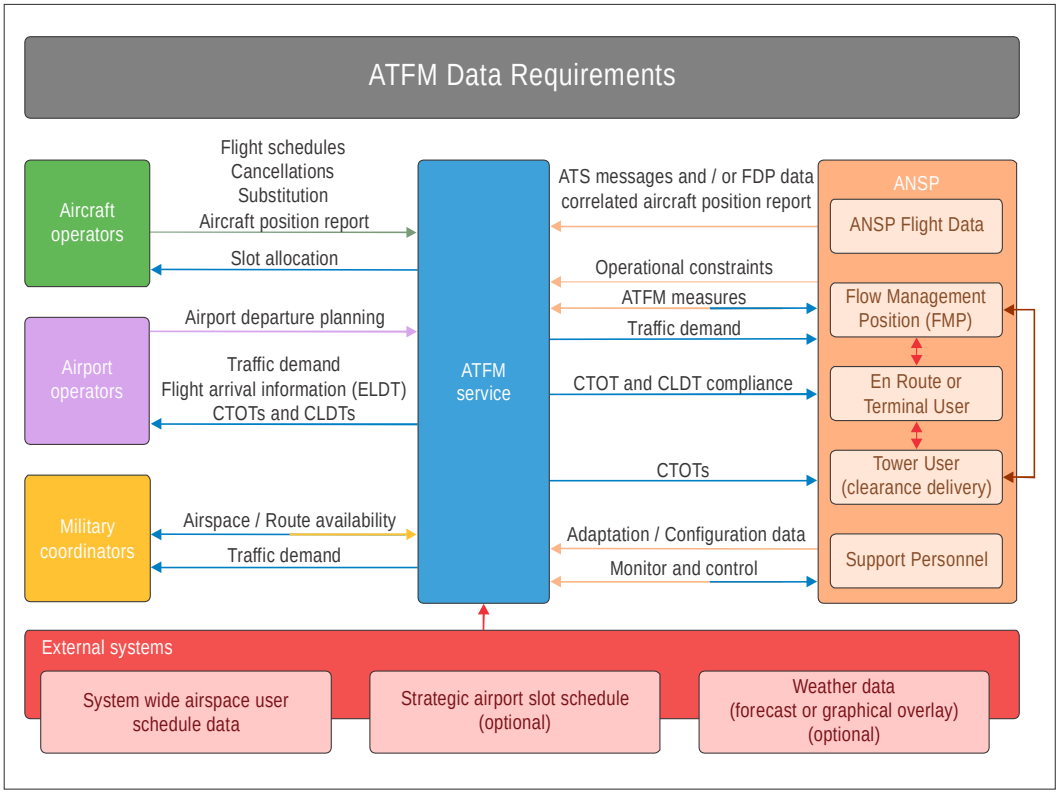

FIGURE 1.5: Data Requirements of an ATFM Service (ICAO, 2018)

As the world becomes increasingly globalized, leading to increased participation in collaborative operations, and technology supporting such collaboration continue to improve, Collaborative Decision-Making (CDM) is anticipated to bring about benefits as follows:

- Increased level of predictability for ATFM, ATC units and aerodromes while operating close to maximum capacity;

- Increased flexibility of operations;
- Increased time horizon of ATFM leading to better decisions of the appropriate ATFM procedure to execute; and
- Improved predictability and punctuality of flights; leading to reduced fuel use, delays, and flight cancellations.

Some of these benefits have already been realized and evaluated. One example is at the John F. Kennedy International Airport, where CDM allowed flexibility in having planes wait at their gates instead of the tarmac due to increased predictability of runway schedules. An evaluation of the airport, in 2017, has shown CDM benefits translating to approximately USD 123 million of savings, a reduction of 44,000 metric tons of CO2 emissions, 152,000 person days of passenger time saved, and a reduction of average taxi-in times by over three minutes as compared to 2009 (ICAO, 2018). CDM benefits were also evaluated at Singapore Changi Airport in 2017, with a reduction of taxi-out time by an average of 90 seconds despite increases in traffic volume (CAAS, 2017).

While the benefits of CDM have been demonstrated across multiple airports, due to the significant investment required and inertia of utilizing current methods, not all stakeholders are enthusiastic in implementation of CDM methods. More work has to be done to assure stakeholders and the wider community of the potential benefits of CDM, and forms a primary motivator for our research direction.

1.1.3 System Wide Information Management

System Wide Information Management (SWIM) is the underlying system infrastructure, rolled out as part of the GATMOC, that supports both the CDM and Flight and Flow Information for a Collaborative Environment (FF-ICE) initiatives. Current ATM systems lack the interoperability envisioned by the GATMOC, and SWIM seeks to bring about the change from point-to-point data exchanges, to system-wide interoperability.

There is a need to shift away from the point-to-point approach, as it comes with several drawbacks. These drawbacks include challenges in onboarding new users or new data formats due to multiple interfaces, asymmetric costs in accessing information on a timely basis, and data duplication or inconsistencies arising from the differences in handling ATM data by different parties (ICAO, 2015). The illustration in Figure 1.6 depicts the way in which SWIM facilitates the exchange of information between various stakeholders throughout the aviation ecosystem through a single interface, achieving the goal of a globally interoperable environment for its participants.

The SWIM initiative provides many benefits, including improved decision-making during the flight planning (strategic) and operational (tactical) phases, cost-effective

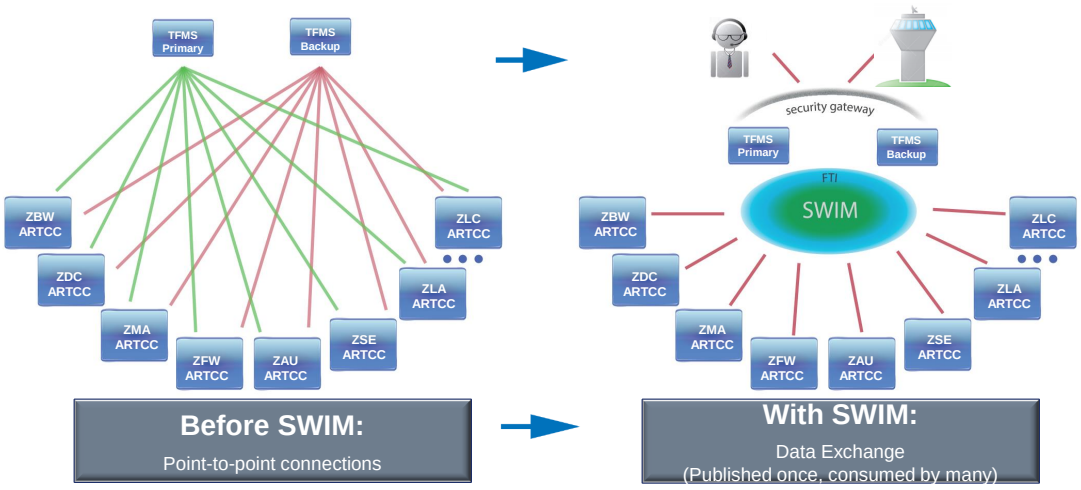

FIGURE 1.6: Data Exchange Under SWIM (ATO, 2016)

communications with global standards for the exchange of information and improved system performance (ICAO, 2015). Furthermore, ICAO has prepared for mixed-mode operations, where some regions may continue to use legacy systems, and others the upgraded systems. To that end, ICAO has proposed specialized gateways for messaging and a staged transition to maintain an acceptable degree of system interoperability (ICAO, 2015).

1.1.4 Trajectory-Based Operations

Trajectory-based operations (TBO) calls for strategical planning and managing flights throughout the its trajectory, information exchanged between air and ground systems, and the capability of aircraft to accurately trace a trajectory in 4D, subject to rescheduling or rerouting requests. A 4D trajectory defines the flight path of an aircraft from departure to arrival, in four dimensions — latitude, longitude, altitude, and time. Major players including Europe, the United States, and countries within Asia (Kok, 2023; SESAR, 2024) are continuing to develop their capabilities to operate the TBO concept. An example of how the TBO concept will benefit airspace users is given in Figure 1.7.

1.1.5 Flight and Flow Information for a Collaborative Environment

In support of the GATMOC, the FF-ICE describes the information environment to support an integrated, harmonized, and globally interoperable ATM system, and supports both the aforementioned SWIM and CDM initiatives. The FF-ICE is aimed at addressing several of the current limitations faced by ATM systems, which are elaborated in

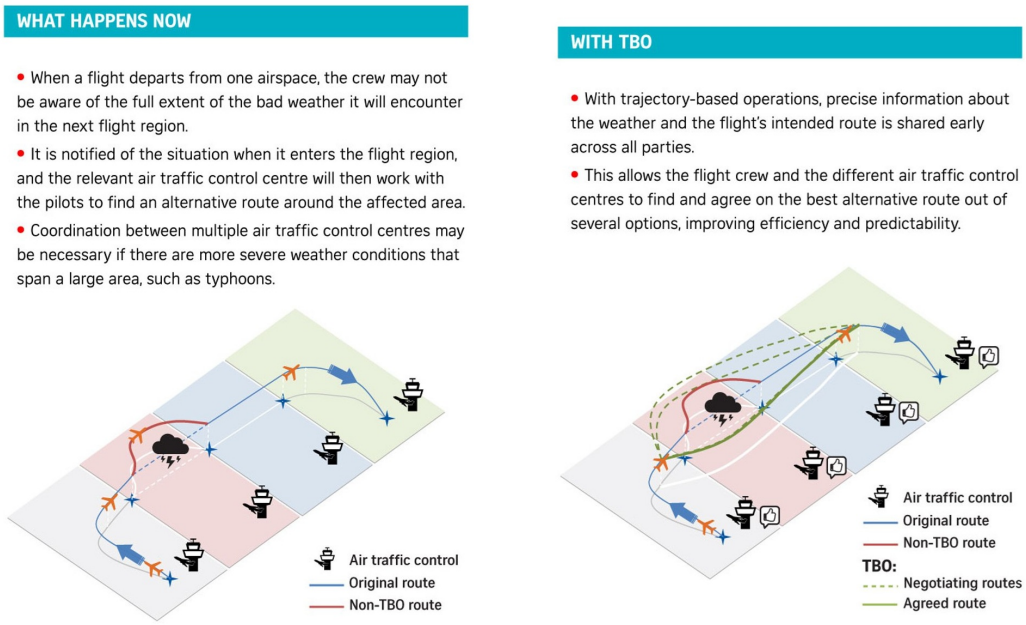

FIGURE 1.7: TBO Achieves Greater Airspace Efficiency (Kok, 2023)

detail in (ICAO, 2012). Most importantly, the key objective may be summarized as providing secure, consistent, and timely flight information to all stakeholders through an effective information sharing network.

As the aviation community migrates towards a TBO airspace model, FF-ICE will lay down the necessary foundations to achieve efficient TBO operations. The filing of flight plans are soon expected to include a 4D trajectory, which encompasses both the route and the times at each point along the route. Furthermore, when the ATM system receives updated information such as meteorological conditions, winds, specific aircraft availability, and resource demand, new constraints are imposed on the aircraft, necessitating updates for the accuracy of flight plans. Under the FF-ICE, these flight plans may be refined even after departure and shared to all relevant stakeholders through the SWIM system. The revised flight plans provide more precise forecasts of departure and arrival times, along with any necessary route modifications in response to airspace capacity constraints arising from adverse weather conditions or military operations. Timely access to such information would allow Air Navigation Service Providers (ANSPs) and airspace users to take immediate action, ensuring the limited system resources are efficiently distributed. Figure 1.8 illustrates the high level interactions between FF-ICE stakeholders.

FF-ICE envisions various levels of information sharing, termed as FF-ICE releases. As ICAO and the Air Traffic Management Requirements And Performance Panel (ATM-RPP) have not officially released the ICAO provisions for FF-ICE Release 1 (R1) and

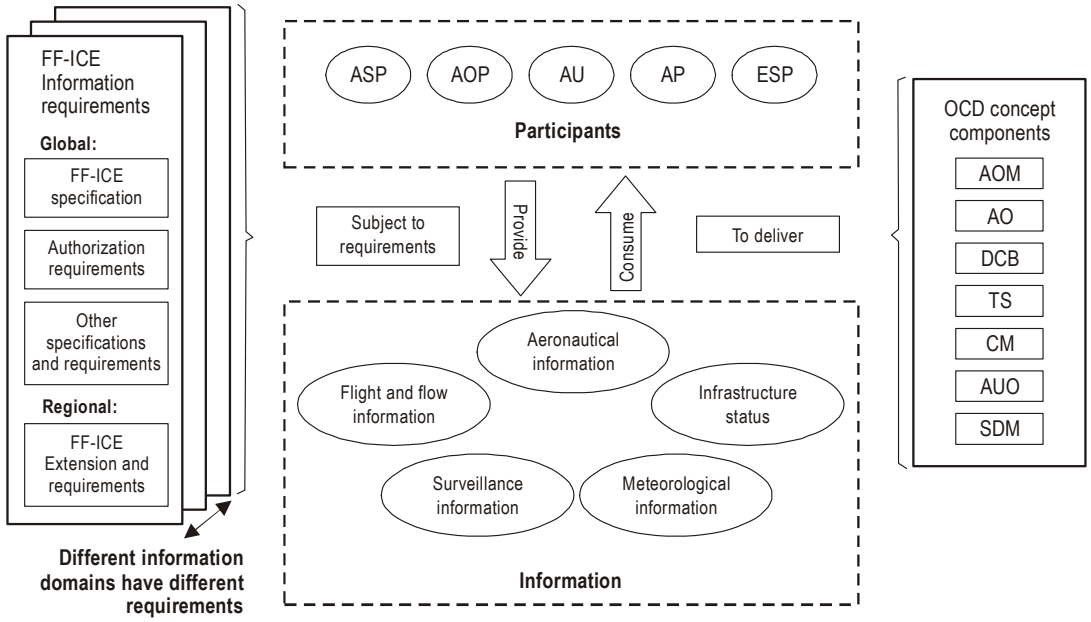

FIGURE 1.8: Information Provision and Consumption by FF-ICE Participants (ICAO, 2012)

FF-ICE Release 2 (R2) at the time of writing, we define these based on our discussions with the Civil Aviation Authority of Singapore (CAAS) and several FF-ICE evaluation reports (D. Liang, Ngo, et al., 2021). We denote FF-ICE Release 0 (R0) as the minimum level of information sharing, where pre-departure flight plans are shared with the relevant ANSPs, containing only route and scheduled departure time information. The FF-ICE R1 builds on this, where filing of pre-departure flight plans are optimized using Preliminary Flight Plans and trial requests, both of which will be filed in the SWIM system. The actual departure time of flights will also be shared among participants in FF-ICE R1, by the airport of origin. The FF-ICE R2 is still written as a draft and not vetted by ICAO, but the general intent will be sharing of post-departure information with relevant ANSPs, during the execution phase of flight. This is proposed to include systemic data sharing for all aircraft between air and ground systems and full harmonization of trajectory information across all stakeholders.

Owing to the level of detail and timeliness of information received, FF-ICE is expected to bring about the following benefits. The FF-ICE will provide airspace users with increased flexibility to negotiate for trajectories, increased speed of reaction to changes in the airspace, increased predictability and progressive migration towards TBO, another key driver of the GATMOC (D. Liang, Ngo, et al., 2021). Central to this dissertation, we also aim to show that for regions operating at FF-ICE R1, the actual departure time may be utilized to improve the efficiency of collaborative ground delay programs, leading to a reduction of airborne delays.

1.2 Motivation

Many regions of the world operate in a decentralized fashion. However, a common hypothesis is that even in a decentralized system, outcomes can be improved through information sharing. A possible method to evaluate the hypothesis, is through simulating ATFM operations, modeling key decision makers as independent optimizers, but responding to shared information. We seek to demonstrate the value of information sharing and collaboration by building a realistic simulation tool that models the Southeast Asia region, specifically in the Association of South East Asia Nations (ASEAN) region, which consists of Brunei Darussalam, Myanmar, Cambodia, Indonesia, Laos, Malaysia, Philippines, Singapore, Thailand, and Vietnam. We also include Flight Information Regions (FIRs) for Sanya and Hong Kong, given their proximity to ASEAN. We collectively refer to this region as *ASEAN Plus*. Critical to this dissertation, an analyst using the tool would be able to modify the extent of information sharing by selecting regions to be placed under either the FF-ICE R0 or FF-ICE R1 information regime. This analysis will be conducted for both full and partial participation in the FF-ICE program, reflecting the expected incremental adoption in which only specific FIRs or airport pairs engage in information sharing and collaborative decision-making. The analysis would generate results detailing the value of information sharing and collaboration, such as reductions in airborne delays or fuel consumption, or trade-offs between flights that do and do not participate in the FF-ICE program. The analyst would also be able to modify the permitted flow capacity at waypoints and airports, as a representative analysis should be conducted under multiple airspace capacity levels, given that an airspace operating at near full capacity would likely display greater benefit from information sharing as compared to a sparsely populated airspace. Additionally, these results may be aggregated or disaggregated at various degrees of granularity, such as at the regional level or individual flight level.

In line with the goals outlined in the GATMOC, implementation of the FF-ICE seeks to bring about a greater level of collaboration between stakeholders across different regions, and realize the vision of an integrated, harmonized, globally interoperable ATM system. This dissertation seeks to deliver research insights, through use of an airspace network simulator, on the value of information sharing and collaboration under FF-ICE R0, and FF-ICE R1 and mixed mode operations. Both complete participation and mixed mode operations, only involving some regions or particular airport pairs operating FF-ICE R1, are used to demonstrate and quantify the benefits of collaborative ground delay programs and the enhanced information sharing capabilities of FF-ICE R1.

1.3 Overview of Modeling Formulations and Objectives

To ensure clarity for both academic and practitioner audiences, we briefly summarize the key formulations and objectives of each modeling endeavor undertaken in this dissertation.

- **Airport Flow Regulator (AFR).** The AFR formulation captures runway capacity envelopes and assigns departure and arrival slots. We use a three-channel system that generically represents any airport runway configuration. The objective function minimizes the makespan while ensuring conflict-free operations.
- **Waypoint Flow Regulator (WFR).** The WFR formulation balances airspace demand and capacity by adjusting flight trajectories, subjected to aircraft separation requirements. The objective function minimizes the deviation from the scheduled time while ensuring conflict-free trajectories.
- **Discrete Event Simulation (DES).** The DES generates conflict-free flight trajectories under either a First-Come-First-Served (FCFS) or First-Scheduled-First-Served (FSFS) scheduling rule, depending on the collaboration status of the aircraft. The primary collaboration mechanic is the use of Ground Delay Programs (GDPs). It is integrated into a rolling horizon framework, and models regions as independent agents that make decisions under various information regimes. The objective is not an optimization, but instead, the DES runs a system-level simulation that allows evaluation of the impact of various collaboration regimes. Performance outcomes are evaluated using metrics such as airborne time, ground delay, and fuel consumption.

1.4 Contributions and Key Findings

In Section 2.2.1, we review multiple research papers that are directed towards optimizing complete flight trajectories, with the implicit assumption that there exists a central manager who receives full information of all flights and any changes made, along with a central controller who is able to assign a change in decision variables across all flights. While these research papers exhibit the potential value of a centralized aviation network, there is a gap in the literature in the value of information sharing and collaboration in a decentralized aviation network. Furthermore, while related research articles on the FF-ICE program such as (D. Liang, Cropf, et al., 2019; D. Liang, Ngo, et al., 2021; D. Liang, Sandhu, et al., 2021; N X.D Lu, 2022; Egami et al., 2019; Ngo et al., 2019) provide the operational and technical details of the FF-ICE implementation, they do not offer a quantitative framework for modeling and evaluation. A related work by

Mondoloni (Mondoloni, 2013) evaluates the impact of trajectory prediction accuracy on airborne and ground delay with reference to the FF-ICE. Our work which evaluates sharing of departure information that is to be updated in rolling horizon fashion, explores a different avenue from the work by Mondoloni which is based on a general prediction accuracy criteria. As such, the first and primary contribution of this dissertation is a leading work that demonstrates, in a decentralized ASEAN Plus network, the value of information sharing and collaboration for combinations of single and mixed mode of operations under the FF-ICE R0 and FF-ICE R1 information regimes.

Another contribution of this dissertation is the formulation and implementation of mathematical models and algorithmic routines, that allow us to simulate decentralized ATFM systems. These models include gradient descent ascent, simulated annealing, exact methods, and discrete event simulation, of which results demonstrate to have generated conflict-free trajectories across the entire ASEAN Plus region. Also unique to this dissertation, our simulator has the property of separation of concerns, emulating individual decision makers in a decentralized network, reflecting the reality that decisions for flights often involve multiple stakeholders that may not be working in collaboration with one another.

To support the above contributions, we assert that constructing a detailed ASEAN Plus network comprising of all nodes, arcs, flight capacities, and flight schedules, is essential. The simulator developed represents a realistic ASEAN Plus airspace network, and contain capabilities to simulate flights within this region, under various information sharing schemes, and generate reasonable results that guide the development and adoption of FF-ICE R1. Serving as the testbed for this research, the simulator played a central role in the dissertation and was also delivered as part of the grant supporting this project. Furthermore, the data sources described in Chapter 3, Section 3.6, enhance the realism and credibility of the simulation, thereby strengthening the overall contribution of this work.

We also summarize the key quantitative findings corresponding to the modeling frameworks described in Section 1.3.

For the AFR, we benchmarked the performance of the Queue Pressure (QP) algorithm against the First-Come-First-Served (FCFS) algorithm. The results demonstrated negligible differences, where the makespan under the QP algorithm differed from FCFS by only 0.03% in the regular capacity case and by -0.01% in the reduced capacity case. These results indicate that no substantial capacity and flow regulation benefits can be achieved from resequencing aircraft under the QP algorithm, particularly as ATFM models determine separation times independently of aircraft weight class.

For the WFR, we compared the Gradient Descent Ascent (GDA), Simulated Annealing (SA), and Exact Methods (EM) approaches. Under the reduced capacity case, GDA

demonstrated the best computational performance, resolving all conflicts within a decision window in under 9 seconds. By comparison, SA, EM with a 10-second time limit per region (EM10), and EM with a 600-second time limit per region (EM600), required 75, 102, and 5413 seconds, respectively. While all conflicts were resolved much more rapidly, the deviation from scheduled times under GDA was 26% worse than the best solution obtained. The SA algorithm generated solutions that were 10% better than GDA, 8% better than EM10, but 13% worse than EM600.

The results demonstrate that either QP or FCFS may be used for the AFR without significant performance loss. For the WFR, the preferred algorithm depends on the operational priority. We provide some instances where each algorithm would be chosen over the others.

- GDA is the preferred choice for rapid generation of feasible solutions.
- SA is the preferred choice for generating robust solutions with better quality and scalability to larger problem instances.
- EM is the preferred choice for achieving the most optimal schedules during strategic planning, which is done weeks ahead of actual operations.

Taken together, the AFR and WFR algorithms provide an optimization framework for supporting ATFM units in scheduling conflict-free trajectories within a decentralized operating environment such as the ASEAN region.

In the next modeling framework, the DES, the results demonstrated the value of information sharing and collaboration within the ASEAN Plus region. Key findings include a system level reduction of airborne time of 0.75% and 2.19% for the base and reduced capacity case respectively. However, since most flights do not experience a collaborative GDP instruction, we narrowed down to consider only flights that experience an airborne saving of more than 5 minutes. Under this condition, the savings increased to 12.3% and 17.4% respectively. Another key result is that partial collaboration is already beneficial for participating regions, without requiring complete participation from all regions. For example, with the baseline of all regions within ASEAN collaborating, having only 6 regions collaborating, out of a total of 14, yields 46% of the total possible savings. The results also revealed a conundrum for when too few flights participate in collaborative GDPs, where fewer than 15 collaborative flights led to greater delays as compared to airborne savings at the network level. Finally, the results showed that with a reduction of airspace capacity of 20%, we see an increase of airborne savings under collaboration of over 200%, signifying the importance of collaborative measures as the growth of demand is predicted to outpace the increase in airspace capacity.

1.5 Dissertation Structure

The organization for the remainder of this dissertation is as follows. Chapter 2 starts introduces Air Traffic Flow Management (ATFM) and the concepts that will be used throughout the dissertation, and also discusses the key areas of ATFM research. The three key themes we explore and discuss as part of the literature review are the *Flow Management Problem for Complete Trajectories*, the *Flow Management Problem within the Terminal Maneuvering Area*, and the *Runway Sequencing Problem*. In Chapter 3, we provide a high level modeling and interpretation of information sharing under FF-ICE R0, FF-ICE R1, and mixed mode operations. Chapter 3 also gives an overview of two simulation models, the first is the Airport Flow Regulator (AFR) and the Waypoint Flow Regulator (WFR). These algorithms work together to simulate control decisions at the airport and waypoints separately, and combine their results at the end of each simulation step. The second is the Discrete Event Simulation (DES), which provides an integrated control algorithm for flow regulation at both airports and waypoints. Application of the rolling horizon concept, as well as the data sources, are also included in this chapter. In Chapter 4, we describe in detail the AFR, discussing our model for airport capacity, our mathematical model for the AFR, and a comparison of results between a novel queue pressure algorithm and a classic First-Come-First-Served (FCFS) algorithm applied to the AFR. Following, Chapter 5 formalizes our mathematical model for the WFR, and introduces two machine learning models, namely Gradient Ascent Descent (GDA) and Simulated Annealing (SA), that we have applied to the WFR. Details of the application of GDA and SA to the WFR follow, and a comparison of results between the two machine learning models against exact methods concludes the chapter. Chapter 6 introduces a separate algorithm, a Discrete Event Simulation (DES) which provides a rule-based alternative to the AFR and WFR, providing an end-to-end decision making algorithm based on simple priority rules. In Chapter 7, we briefly describe the integration between the inputs and outputs of the AFR and WFR and how the simulation tool works, together with a comparison of the optimization-based and rule-based approaches. Conclusions and recommendations based on the results generated through the lens of the value gained through information sharing are also included in this chapter. In Chapter 8, we conclude the dissertation by evaluating the insights and conclusions derived from our experiments and results. We also summarize the key contributions of this work and outline potential directions for future research.

Chapter 2

Air Traffic Flow Management

2.1 ATFM Background

The early beginnings of Air Traffic Flow Management (ATFM) can be traced back to the 1930s, with maps, drawings and mental computations to ensure safety of all flights. While this process was sufficient with few flights in the airspace, as the demand for air freight and passenger transport surged, in part catalyzed by World War II, it became apparent that more sophisticated services needed to be put in place to manage air traffic. In 1958, more people crossed the Atlantic by air than by sea, bringing about the Jet Age. This was a period in history which fostered a burgeoning growth in air traffic, with passenger travel growing by more than 1,000 percent from 1958 to 1977 (NATCA, 2019). During this period, the Federal Aviation Administration (FAA) was created with the responsibility to handle air navigation and traffic control in the United States. By 1975, all Air Route Traffic Control Centers were equipped to receive real time, in flight data on computers, and along with the newly established Central Flow Control Facility, began to use more modern methods to regulate nationwide air traffic flow (NATCA, 2019).

In 1981, outdated systems and long working hours for Air Traffic Controllers (ATCs) led to stress related ailments such as hypertension occurring at abnormally high rates. These conditions culminated in a strike by Professional Air Traffic Controllers Organization in the United States. The strike concluded with 11,000 ATCs being fired by then president, Ronald Reagan. This once again beckoned the necessity of more efficient ATFM measures, as few ATCs had to bear the immense workload of the managing the flight ecosystem. Such events motivated researchers to study and propose new ATFM models for Air Navigation Service Providers (ANSPs), airlines and airports to adopt. Some early papers include (Dear, 1978; Odoni, 1987; Butler, 1987). The research and advancement in technology did bear fruits, as demonstrated in 1995, when the new Denver International Airport, with ATFM measures implemented, recorded a fivefold decrease in delays compared to the old Denver Stapleton Airport (NATCA, 2019).

In recent years, the aviation industry has continued to grow at an astounding pace,

of about 6% annually from 2009 to 2019 (ICAO, 2023b). This growth was temporarily upended, when the COVID-19 pandemic caused air travel to abruptly crater in 2020. At the height of the pandemic, airline capacity fell to one third of its 2019 levels. Despite that, recovery has been swift, with 2023 passenger travels almost equivalent to that of the peak in 2019 as shown in Figure 2.1, and revenues exceeding 2019 levels, with data given in Figure 1.1.

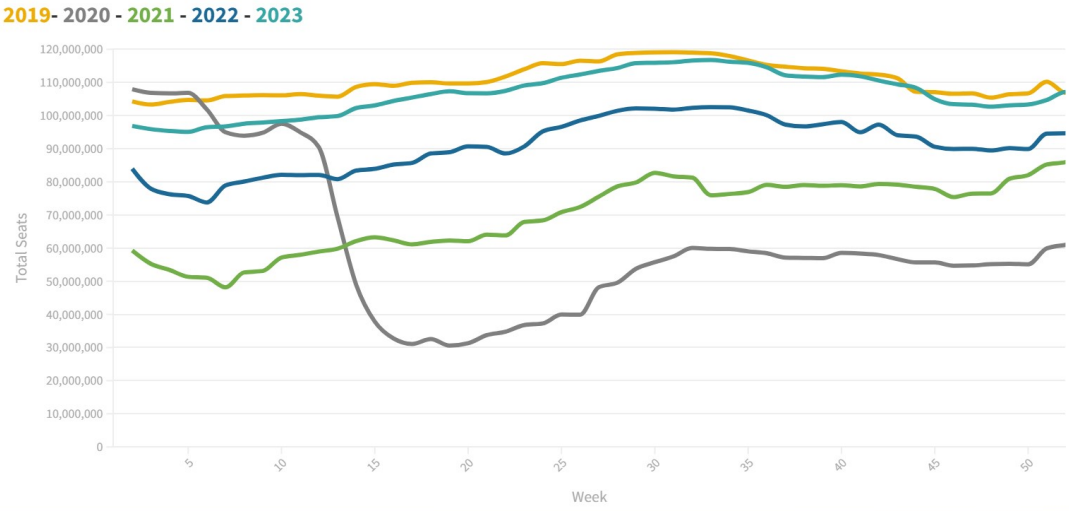

FIGURE 2.1: Total Airline Capacity 2019-2023 (OAG, 2024a)

With air travel demand projected to continue growing at a Compounded Annual Growth Rate of 4.3% from 2015 to 2035 (ICAO, 2019), there exists an ever present need to develop and implement more efficient ATFM models, services and systems. We continue our discourse by describing a few common ATFM concepts, followed by ATFM objectives, phases, and procedures. Next, we will explore how Collaborative Decision-Making (CDM) plays a pivotal role in improving the efficiency, reliability and predictability of modern day ATFM.

2.1.1 Common ATFM Concepts

This subsection introduces some of the common ATFM concepts that will be used throughout this dissertation.

- Flight Information Region (FIR): An FIR is a specified region of airspace that is delegated to be controlled by an Area Control Center (ACC). In the U.S. such centers are named Air Route Traffic Control Center (ARTCC). Smaller countries may be contained within a single FIR, while larger countries may have their airspace split into multiple FIRs. The division of airspace is performed via international agreement, through the International Civil Aviation Organization (ICAO).

- 4D trajectory: The 4D trajectory of an aircraft integrates the dimension of time into the spatial 3D path. This implies the 4D trajectory consists of the routing information (space) and time at each point of the flight (time). This term is commonly used in Single European Sky ATM Research (SESAR) and Next Generation Air Transportation System (NextGen). For the remainder of this dissertation, a path refers to a time independent sequence of nodes or vertices, while a trajectory refers to a time dependent sequence of nodes or vertices.
- Waypoint: A waypoint is a geographical node that can be used to define a flight path, most often used for flight routes using area navigation (RNAV). Waypoints are defined in terms of longitude and latitude.
- Standard Instrument Departure Route (SID) and Standard Arrival Route (STAR): A SID is a route from the take-off phase to the en-route phase of a flight. A STAR is a route from the en-route phase to the initial approach for landing. Both the SID and STAR a pilot intends to use are often included in the submitted flight plan.
- En-route: The en-route phase of a flight begins after the flight completes the SID, and ends when it enters a STAR.
- Terminal Maneuvering Area (TMA): A TMA is an area of controlled airspace near major airport where there is normally a high density of air traffic. It is often delineated as a circular boundary centered on the airport.

2.1.2 Objectives

The International Civil Aviation Organization (ICAO) defines ATFM as a service established with the objective of contributing to a safe, orderly and expeditious flow of air traffic by ensuring that Air Traffic Control (ATC) capacity is utilized to the maximum extent possible, and that the traffic volume is compatible with the capacities declared by the appropriate Air Traffic Service (ATS) authority (ICAO, 2016). These goals may be achieved through effective balancing of airspace demand and capacity, and minimizing fuel use, delays and potential conflicts. Such procedures are discussed in Section 2.1.4. Other methods include improved services through means such as information sharing and more accurate and timely flight updates, as discussed in Section 1.1.2.

2.1.3 Phases

ATFM is carried out in three phases, following the standards set out by ICAO's Air Traffic Management (ATM) procedures manual (ICAO, 2016):

- Strategic planning. This is when the action is performed more than one day before on which it will take effect. It is often carried out far ahead of time, typically two to six months ahead.
- Pre-tactical planning. This is when the action is performed one day before which it will take effect.
- Tactical operations. This is when the action is performed on the day on which it will take effect.

The strategic planning phase is carried out after consultations with ATC and aircraft operators, with the aim of jointly reaching potential solutions that would resolve demand and capacity imbalances. Demand forecasts for the upcoming season are assessed, and appropriate measures are taken in cases where imbalances between demand and available ATC capacity are anticipated. Some of the common ATFM measures are rescheduling flights, re-routing traffic flows, proposing to the appropriate ATC authority for a capacity adjustment (e.g. rescheduling the duty roster for traffic controllers at each facility), and identifying the need for tactical ATFM measures. The planning should, as far as reasonably practical, minimize the flight time and distance penalties, while allowing some room for the choice of routes, especially for long-range flights. As these plans are made way in advance of the day of flight, they may be adjusted according to the forecasted demand, if and when it changes.

The pre-tactical planning phase consists of fine-tuning the strategic plan given access to more accurate demand and ATC capacity information as flight details for the next day should have been published and made available to all stakeholders. The aim of this phase is to optimize capacity through an effective management of available resources. A common method to optimize capacity is the re-routing of traffic flows. However, during this phase. less flexibility in re-routing is available as compared to re-routing during the strategic planning phase. Also, while tactical possibilities were identified earlier, tactical measures will be decided upon in this phase.

The tactical operations phase is when the agreed tactical measures are executed to provide a smooth flow of traffic where demand would have otherwise exceeded capacity. ATC and ATFM units are responsible for monitoring the current air traffic circumstances to ensure that ATFM measures are applied with the desired effect, and to initiate corrective measures such as re-routing traffic or ground delay programs where excessive delays are reported, as to maximize the available capacity. ATC and ATFM units are also responsible for coordinating plans with the aircraft flight crew and issuing them advisories where new restrictions or delays may be imposed. Additional measures to mitigate capacity balances should be proactively implemented when

unforeseen changes transpire resulting from staffing problems, flight delays, extreme weather conditions or a revision of ATC capacity.

Figure 2.2 gives an overview of the phases of ATFM. As the crux of this thesis explores the value of information sharing and collaboration between receiving real-time flight information and collaborative ground delays under the FF-ICE R1 initiative, our focus will be on the tactical operations phase.

FIGURE 2.2: ATM Planning and ATFM Phases (ICAO, 2018)

2.1.4 Procedures

ATFM procedures are techniques used to manage air traffic demand according to system capacity. While effective, these procedures should be implemented as early as possible in the ATFM planning process, so as to reduce disruption to normal flight operations. We first outline four common ATFM measures: runway scheduling, ground delay program (GDP), re-routing, and minutes-in-trail (MINIT).

Runway scheduling may take place at all phases. For example, long-term planning and decisions, such as runway configuration and capacity allocation, are typically carried out well in advance, during the strategic phase. Runway scheduling during the pre-tactical and tactical phase involves assigning runway times to each aircraft, in a manner that satisfies the wake turbulence separation requirements, while maximizing

the usage of runway capacity, and reducing delays. This task is significant and receives heavy attention due to airport runways being recognized as one of the major bottlenecks in ATFM.

GDPs may take place either during the pre-tactical or tactical phase, where aircraft have their departures delayed to mitigate an imbalance of demand and capacity within the airspace or at an aerodrome. The main aim of GDPs are to minimize airborne delays through the substitution of airborne delays with ground delays with the goal of reducing operational costs. GDPs observe greater benefits when they are done in a collaborative manner (ICAO, 2018), as it is likely that the reserved time slots have to be revised at both the origin and destination aerodrome.

Re-routing of flights can take place either during the pre-tactical or tactical phase, where mandatory or advisory re-routing notices are promulgated to aircraft, with the aim of reducing the number of flights scheduled to arrive at a capacity constrained airspace or aerodrome.

MINIT operations take place exclusively during the tactical phase, where the number of minutes between successive aircraft is changed at a boundary point. Due to the potential cascading effect of MINIT operations increasing ATC workload, regular use may suggest that other ATFM measures should have been considered.

We also describe another two procedures for management of air traffic, airborne holding and speed control. Although these may sometimes be categorized separately as ATC measures, due to their common use case for handling minor demand and capacity imbalances. For the purpose of this discussion we subsume them under the more general classification of ATFM procedures.

Airborne holding, commonly executed with aircraft circling racetrack-shaped holding pattern, is usually carried out when aircraft have arrived to their destination, but are unable to land due to reasons such as inclement weather or the runway being unavailable due to congestion or other reasons. Multiple aircraft may use a holding pattern simultaneously, with multiple levels existing for a holding pattern that are separated vertically by 300m or more. A single layer of a holding pattern is depicted on the left in Figure 2.3. Other forms of holding, include vectoring and the point merge system, with such operations are usually carried out to achieve the required separation at a boundary, or the specified runway sequencing. Here, aircraft are instructed to deviate slightly from their original route to increase or decrease the time to reach the next point in its flight trajectory. This is depicted on the right of Figure 2.3.

ATC may also impose speed control by requesting an aircraft to adjust its speed in a specified manner, often with the aim to maintain a required amount of separation, to absorb delay, or as an alternative to vectoring or holding patterns.

Note that ATFM procedures vary by country, with each country deciding upon its

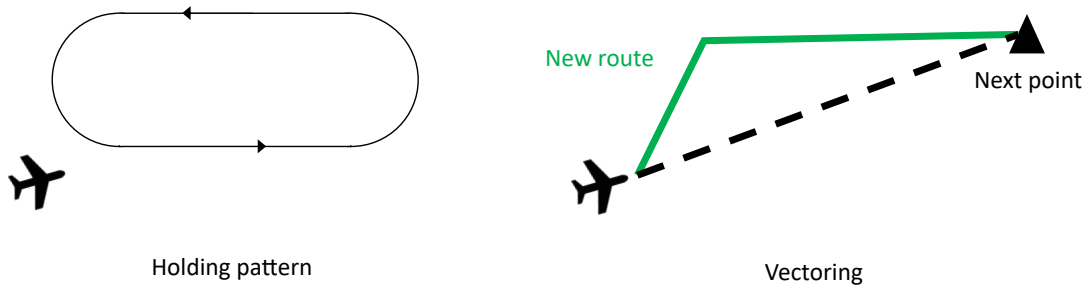

FIGURE 2.3: Two Types of Airborne Holding

allowed and preferred operations. For example, in France and Germany, speed regulation, ground holding, the point merge system and vectoring are permitted, while holding patterns are prohibited. For the models we have developed in this paper, the ATFM procedures focused on are runway scheduling for the Airport Flow Regulator (AFR) in Chapter 4, and airborne holding and speed control for the Waypoint Flow Regulator (WFR) in Chapter 5. Additionally, the Discrete Event Simulation (DES), which has full control of the entire flight trajectory, may utilize all the ATFM procedures used by the WFR and AFR.

2.2 Key Areas of ATFM Research

ATFM research may be subdivided into a set of smaller, more specific problems. We have identified three key areas of ATFM research that are of major importance to the ATFM and wider aviation community. The first, and of the largest scope, is ATFM for complete trajectories. The next, is ATFM within the Terminal Maneuvering Area (TMA), and last, the runway scheduling problem. The extent of these subfields of research are contained within the solid, dashed and double-lined boxes in Figure 2.4 respectively.

As a preamble to the literature review in the following subsections, the objectives and constraints for each research study may differ due to varying stakeholder aims. We have chosen to classify the problems according to its scope, based on the area of air and/or ground of which the study is focused upon. The following subsections introduces the problem and its modeling framework. It will be followed by classifying the studies by objectives, constraints, variables and algorithms chosen.

While it is possible to formulate and solve for an optimal solution for these three problems simultaneously, as done by (Tan, 2021; Balakrishnan and Chandran, 2014) in Flow Management Problem For Complete Trajectories, in this dissertation we will model and solve these problems in a decentralized fashion, under a rolling horizon, integrating the solution for one problem into the next. The aim of such a formulation

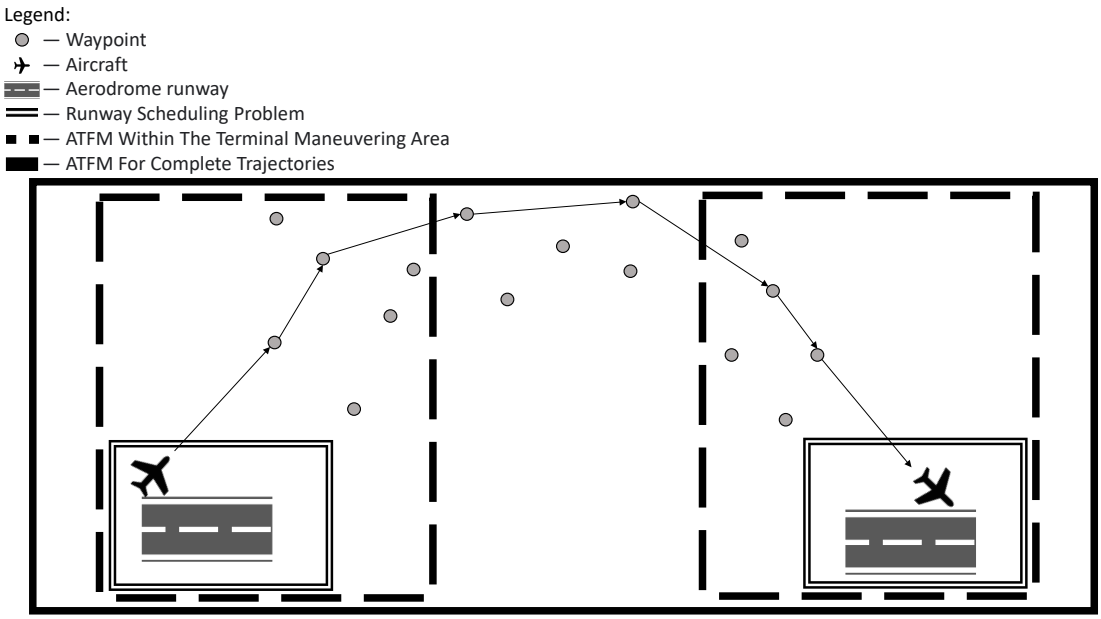

FIGURE 2.4: Three Key Areas of ATFM Research

is displaying the value of information sharing and collaboration, for a decentralized ASEAN Plus region, under various levels of regional participation. The lens through which the problem is viewed, is an operational standpoint of ATCs, where decisions are made in real time, with the scope of control limited to individual FIRs under FF-ICE R0, or multiple collaborating FIRs under FF-ICE R1. This is contrasted by the more common studies that solve the problem for an entire day or so, considering all flights for the entire time period, as a single system with a central decision maker. The problem formulation, along with other pros and cons of a decentralized model, are further discussed in Section 3.

The rolling horizon nature of the simulation allows us to model uncertainty and limited information. At each simulation step, the information available to each FIR is updated based on its level of information sharing. We list some examples of information that would be useful for FIRs to exchange with each other:

- The time at which a flight is expected to enter its subsequent FIR, in particular if it differs from its scheduled flight plan;
- The updated flight departure time; and
- Requests for GDP to be carried out, especially during periods of high congestion.

The subroutine for flight planning within each FIR must be re-run with this new information at each time step, while maintaining the minimum required separation, and minimizing arrival delays, for all planned trajectories. A key parameter in defining a

rolling horizon simulation is the planning horizon, or look-ahead time. Tactical ATFM models typically look ahead at most a few hours because forecasts beyond that time are too unreliable for tactical operations. For example, (Henry, Delahaye, and Valenzuela, 2022) and (Rapaya, Notry, and Delahaye, 2021) have look ahead times of 21 minutes, and 120 minutes respectively. Rolling horizon models typically split a flight into multiple phases, where the control over a flight is reduced as its TTO gets later, relative to the current situation time. For example, in (Huo, Delahaye, and Sbihi, 2021) and (Henry, Delahaye, and Valenzuela, 2022), flights are categorized as either 'planned', 'active', 'ongoing', or 'completed'. Similarly, we require a process for classifying flights to progressively identify TTOs that should be updated or frozen as the simulation progresses.

Uncertainty is a commonly considered criterion in flight optimization, as it causes fluctuations in both airspace capacity and demand. Some studies that incorporate uncertainty include (Guruge, 2020) and (Bosson and Sun, 2016). The study by (Guruge, 2020) considers the maximum and minimum times of a flight based on a generated probability distribution function and imposes an either-or constraint. The study by (Bosson and Sun, 2016), has a different way of addressing uncertainty. The study first generates independent and identically distributed scenarios for each flight, then subsequently solves the problems using mixed integer linear programming (MILP), and finally, integrates and selects the solutions that minimize the expected value of the objective function.

We discuss, in greater detail, how the rolling horizon captures and handles uncertainty in the subsequent paragraphs.

A common concept employed by a number of studies in our literature review is the Receding Horizon Control (RHC), also used under the more general term, sliding window or rolling horizon, in other literature. This concept was first applied to ATFM in (X.-B. Hu and Chen, 2005), who adapted it from control engineering. The study by (X.-B. Hu and Chen, 2005) found that flights experience less or equal delays utilizing the RHC concept as compared to a single optimization routine carried out at the start of the day, when both experiments were evaluated using the same optimizer. More recently, (Huo, Delahaye, and Sbihi, 2023; Ma, 2019; Henry, Delahaye, and Valenzuela, 2022; Rapaya, Notry, and Delahaye, 2021) have applied the RHC concept to address uncertainty in optimization problems and to reduce problem dimensionality.

RHC looks N steps ahead, in terms of a given cost function, and implements the decision at the current time step based on the observed cost function. The implemented decision at the current time step is then propagated to the next time step. This is known to be more robust against uncertainties as the online updated information is used to improve the decision at each time step (X.-B. Hu and Chen, 2005). This concept also

benefits from a reduction in computational complexity, as most formulations render the problem computationally intractable. Hence, by limiting the variables to only that within the current time step, the RHC leads to an exponential reduction in computational time. With reference to Figure 2.5, we illustrate a toy implementation of RHC, with a look-ahead of three steps and a time-shift of one step. In the first iteration, we optimize the variables pertaining to the first eight flights. In the second iteration, we optimize the variables pertaining to the fourth to the tenth flight inclusive. The algorithm terminates when the start time of the sliding window to be modified exceeds the time of last flight.

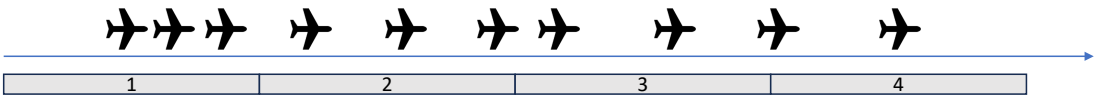

FIGURE 2.5: Sliding Window Concept

Chapter 3 later introduces the application of RHC within the FF-ICE simulator to model and compare information sharing regimes under the FF-ICE R0 and R1 frameworks. An example of applying the sliding window concept to different information regimes is the handling of flight delay data: under FF-ICE R1, a flight delay is reflected in an earlier sliding window, whereas under FF-ICE R0, this information is only incorporated once the flight has entered the relevant FIR. This modeling approach enables the evaluation of various ground delay and flight performance metrics across the different information sharing regimes under investigation.

2.2.1 Flow Management Problem for Complete Trajectories

ATFM for complete trajectories considers the entire flight trajectory, from departure, to en-route, to arrival. These problems are often solved on large-scale instances and cater to the strategic and pre-tactical planning phase, as such, although shorter code execution times are still favored and have been more common in recent years, studies with code execution times of over an hour are still deemed feasible in the ATFM model in studies such as (Balakrishnan and Chandran, 2014). Note that the study by (Huo, Delahaye, and Sbihi, 2023) was included despite not encompassing the departure stage, as it aligns well with this category given that the primary components of a complete trajectory are represented, albeit divided between the en-route and runway phases. Later in the Section 3.2, we see that a parallel may be drawn from this study to our own model of the simulator tool. Both employ a similar concept of partitioning the entire trajectory into smaller components, distinguished by their respective FIRs, which are then optimized independently.

Two common approaches for modeling ATFM decisions are trajectory-based and node-based models. A trajectory-based model generates a set of possible trajectories through use of a subroutine, and the trajectories are selected through the use of binary decision variables, often via integer programming, to optimize an objective function, subject to flow constraints. Papers adopting a trajectory-based approach include (Balakrishnan and Chandran, 2014; Bolić et al., 2017; Huo, Delahaye, and Sbihi, 2023; Lavandier et al., 2021; Tan, 2021). Node-based models are not restricted to selecting from a set of trajectories and so have a larger state space, contributing to their modeling flexibility, at the cost of increased computational complexity. Papers adopting a node-based approach include (Alam et al., 2017; Ozgur and Cavcar, 2014).

The objective functions chosen may be divided into two broad categories. The first are those that only minimize conflicts, done by (Alam et al., 2017; Huo, Delahaye, and Sbihi, 2023; Lavandier et al., 2021). The second category are those that minimize a penalty or maximize benefits, such as congestion, route charges, ground delay, or airline flight connections, done by (Balakrishnan and Chandran, 2014; Bolić et al., 2017; Ozgur and Cavcar, 2014; Tan, 2021). The papers in the second category would instead model the conflict detection and resolution as constraints.

The objective function, and constraints, often dictate the choice of algorithm. For heuristic methods, such as simulated annealing, these are often modeled such that their constraints are part of the objective; while the objective and constraints are modeled separately in the case of exact methods, particularly in linear and integer programming. For example, in (Huo, Delahaye, and Sbihi, 2023), the algorithm chosen was simulated annealing, and the conflicts were modeled as the objective, of which the goal is to minimize. In contrast, the algorithm used in (Bolić et al., 2017) was integer programming, leading to the objective being modeled separately as minimizing the sum of deviation of departure and arrival times, operational costs, airborne costs and route charges, subject to airspace capacity constraints. Hence, we synchronously classify the algorithms in the first category as exact methods, and heuristics for studies in the second category. The benefits and drawbacks of exact and heuristic methods are further discussed in Section 5.2. We highlight the general rule-of-thumb, that heuristic methods are often chosen for problems that are too time-consuming to obtain an exact answer, or in other words, are computationally intractable. The algorithms used for each paper are given in Table 2.1.

The constraints adopted by (Balakrishnan and Chandran, 2014; Bolić et al., 2017; Tan, 2021) are capacity-based, reflecting ATFM research priorities by imposing limits on the number of aircraft allowed to traverse an airspace sector or node within a specified time interval. In (Bolić et al., 2017), an additional constraint on the available

runway configuration at each moment in time, was imposed. The other studies focused on separation based constraints, a central concept in TBO research, on both the spatial and temporal dimensions. This means that a constraint is imposed such that a minimum distance between all flight pairs is required. (Alam et al., 2017) considered the separation based interaction of flights in functional airspace blocks and nodes, (Huo, Delahaye, and Sbihi, 2023) considered separations at nodes, links and runways, (Lavandier et al., 2021) considered speed covariance and proximity at reference points, and (Ozgur and Cavcar, 2014) considered both capacity based and separation based constraints. Occasionally, there have been overlaps between ATFM and TBO research, for example the research (Alam et al., 2017) is geared towards ATFM, but has conflict detection modeled as aircraft violating a protection volume around a point, instead of pure capacity constraints.

Regarding the choice of variables, all studies considered departure time. Here, departure time is equivalent to ground delay, as imposing a ground delay on a flight is equivalent to increasing its departure time. All studies also considered some form of choice in re-routing, with the exception of (Lavandier et al., 2021). Airborne speed was also considered in the papers (Balakrishnan and Chandran, 2014; Huo, Delahaye, and Sbihi, 2023; Tan, 2021).

Table 2.1 provides an overview of the literature on ATFM for complete trajectories.

2.2.2 Flow Management Problem Within the Terminal Maneuvering Area

ATFM for the TMA only concerns itself with managing flights within the TMA. This usually involves either departing flights taking off from the runway up until exiting a SID, or arriving flights from entering a STAR up until landing on the runway. Among all controlled airspace, the TMA is the most congested airspace, with most flight conflicts being detected and resolved within the TMA. As such, optimizing the TMA independently of the complete trajectory has also been considered a critical field of research, given that the requirements are different from the complete trajectories optimization, and a stronger emphasis on conflict detection and resolution exists for ATFM targeted at the TMA level.

This problem may be considered separately in terms of arrivals and departures, for two reasons. First, many airports still operate the Arrival Manager (AMAN) and Departure Manager (DMAN) as separate systems. Next, arrival and departure flights do not interact at SIDs and STARs as they are vertically separated, and although mixed mode runways are not uncommon, ATCs typically use runways as dedicated services, serving only departures or arrivals at any one runway. The studies considering arrivals only are (Henry, Delahaye, and Valenzuela, 2022; Huo, Delahaye, and Sbihi, 2021; Bae

TABLE 2.1: Literature Review for Flow Management Problem for Complete Trajectories

Reference	Objective Function	Variables	Algorithm
Alam et al., 2017	Minimize the sum of interaction across trajectories	Departure time and trajectories	Simulated annealing and local search
Balakrishnan and Chandran, 2014	Maximize the sum of revenue + cancellation penalties − operating costs − delays	Ground delays, airborne speed, rerouting and cancellations	Integer programming and dynamic programming
Bolić et al., 2017	Minimize the sum of deviation of departure and arrival times, operational costs, airborne costs and route charges	Departure time and route	Integer programming
Huo, Delahaye, and Sbihi, 2023	Minimize the sum of conflict and congestion	En-route phase: Alternative route and speed. TMA: Entry speed and choice of runway	Simulated annealing
Lavandier et al., 2021	Minimize the sum of air traffic complexity	Departure time	Selective simulated annealing
Ozgur and Cavcar, 2014	Minimize the sum of weighted ground delays	Departure time and route	Integer programming
Tan, 2021	Maximize the sum of net benefit and connection bonuses	Ground delays, airborne speed, rerouting and cancellations	Integer programming and dynamic programming

et al., 2018), and the studies considering both arrivals and departures are (Frankovich, 2012; Ma, 2019; Ng and Ribeiro, 2023; Zhou et al., 2017).

Similar to ATFM for complete trajectories, exact methods may separate the objectives and constraints, while heuristic methods must include constraints as part of the objective. The study by (Ng and Ribeiro, 2023) is unique, being the only study using integrate the results of both classes of algorithms. They utilized a sequence mutation heuristic to determine sequencing, followed by a linear program to solve the problem for the given sequence, for which the objective is to minimize flight delays. All other studies, with the exception of (Zhou et al., 2017), are modeled with the objective function to minimize flight delays. However, the studies exhibit considerable diversity in algorithmic approaches, as summarized in Table 2.2. For instance, (Zhou et al., 2017) focuses on generating SIDs or STARs that avoid all obstacles, which notably differs from the conventional ATFM objectives for TMA. This represents a valuable research direction, as SIDs and STARs are currently developed manually; the study addresses this labor-intensive process by enabling automated generation of optimized routes.

The constraints chosen across all studies reviewed are consistent, where every study adopted a node-based formulation and hence all the constraints were separation based, considering conflicts at nodes, links and runways. The only variation was that runway conflicts were not considered in (Zhou et al., 2017), and link conflicts were not considered in (Bae et al., 2018; Frankovich, 2012).

All studies have aircraft speed as a variable, with the exception of (Zhou et al., 2017), as its research aim is different, as mentioned. It is reasonable to generalize choice of runway, and the use of airborne holding facilities as a choice of flight path, and this assumption aligns with all studies considering flight path as a variable. Notable variation between studies are, for example that, airborne holding for arrival flights are modeled as a variable only in (Henry, Delahaye, and Valenzuela, 2022) and (Ng and Ribeiro, 2023), with it termed "length of merge point" in the former, and "use of holding stacks and vectoring" in the latter. A point merge system is similar in concept to vectoring, as described in Section 2.1.4, except that it has predefined paths to merge into, while vectoring relies more heavily on the skill of the ATC. Also, other differences exist, where only (Frankovich, 2012; Henry, Delahaye, and Valenzuela, 2022; Huo, Delahaye, and Sbihi, 2021; Ma, 2019) consider multiple runways, and have therefore modeled choice of runway as a variable. In our formulation of the AFR, in Chapter 4, we also allow the possibility of multiple runways, where there may be more than one runway available for an arrival or departure flight to choose from.

TABLE 2.2: Literature Review for Flow Management Problem for Terminal Maneuvering Areas

Reference	Objective Function	Variables	Algorithm
Bae et al., 2018	Minimize the sum of flight time	Flight path and airborne speed	Combination of Dijkstra and First-Come-First-Served
Frankovich, 2012	Minimize the sum of weighted time to complete flight	Runway configuration, choice of runway and flight trajectory	Integer programming
Henry, Delahaye, and Valenzuela, 2022	Minimize the sum of airborne and ground delays, and node and link conflicts	Airborne speed, TMA entry time, choice of runway, and length of merge point	Q-learning, with a genetic algorithm to select Q-learning parameters
Huo, Delahaye, and Sbihi, 2021	Minimize the sum of node, link and runway conflicts	Airborne speed, TMA entry time, and choice of runway	Simulated annealing
Ng and Ribeiro, 2023	Minimize the sum of conflicts and airborne delays	Runway sequence, airborne speed and use of holding stacks or vectoring	Linear programming and sequence mutation
Ma, 2019	Minimize the sum of conflicts, airside capacity overload and time deviation	TMA entry time, TMA entry speed, choice of runway, departure runway, and departure time	Compares mixed integer linear programming and simulated annealing
Zhou et al., 2017	Minimize the sum of interaction with obstacles	Flight path	Branch and Bound

2.2.3 Runway Sequencing Problem

The Runway Sequencing Problem (RSP) involves sequencing flights on the available runways, subject to aircraft observing separation constraints. Runway use may be segregated, allowing only arrivals or departures at a given runway, or mixed, where both arrivals and departures may be allocated on a given runway. An example of a mixed runway configuration, is having every departure aircraft being followed by 3 arrival aircraft. While mixed runways can potentially achieve a higher throughput by maximizing capacity, it is more taxing on ATC and hence, the segregated mode is utilized more commonly. Studies that include scheduling routes for aircraft beyond just arrival or departure, that is, studies that considered airport surface operations such as (Bosson and Sun, 2016; Desai et al., 2022; Ma, 2019), have also been included for completeness, although airport surface operations are currently not a part of our research. However, we may still draw insights from these studies, given their node-based airport surface formulation is conceptually similar to our node-based airspace formulation for the WFR in Chapter 5.

As little room for model variability exists within the limited scope of the problem, all studies have set the objective function as the minimizing the sum of time deviation from the preferred or scheduled time. The exceptions to this are (X.-B. Hu and Chen, 2005) and (Prakash, Piplani, and Desai, 2018), whose objective functions minimize the airborne delay and makespan respectively, which are atypical of the RSP, and presumably align with different stakeholder interests. While the majority of studies have chosen to utilize MILP to solve the RSP, given the computationally challenging nature of the problem, different approaches have been undertaken to reduce the state space, and by extension the computational times. A common approach in the studies reviewed is to iteratively reduce the problem into smaller parts until they can be solved directly, and subsequently recombining the results to obtain the final solution. This divide-and-conquer method was seen in studies (Prakash, Piplani, and Desai, 2018; Prakash, Desai, and Piplani, 2021; Desai et al., 2022) that utilize a data splitting algorithm, under the constraint that an aircraft may at most shift k positions before or after its initial position, aptly named Constrained Position Shifting (CPS). The original flight dataset was divided into several pairs of disjoint subsets and solved independently, and subsequently recombined. In contrast, (Ma, Sbihi, and Delahaye, 2019) approached the problem from a different perspective, comparing the performance of a MILP formulation with that of simulated annealing under a fixed time constraint. Both methods were allotted a computational time limit of 10 seconds, with the optimal results obtained by solving the MILP to completion.

Separation constraints were applied consistently in all studies reviewed, with various additional constraints imposed in some studies. For instance, (Prakash, Piplani,

and Desai, 2018; Prakash, Desai, and Piplani, 2021; Desai et al., 2022) have the CPS constraint applied, (Ma, Sbihi, and Delahaye, 2019) constrained the maximum departure time, (Xiangwei, Ping, and Chunjin, 2011) imposed a landing time window, and (X.-B. Hu and Chen, 2005) imposed a constraint where each flight was required to clear a sector by a specified latest time.

Additionally, all studies reviewed have the position and time on the runway as the optimization variable, which is standard for the RSP. Only in (Bosson and Sun, 2016; Desai et al., 2022) do the modeling of variables differ markedly, with the inclusion of routing, as they also consider airport surface operations. Later, as part of our model of the AFR in Chapter 4, we introduce an extra dimension to the RSP, by allowing flights the freedom of selecting a runway. A multiple runway sequencing model is most closely related to the study by (Xiangwei, Ping, and Chunjin, 2011).

TABLE 2.3: Literature Review for the Runway Sequencing Problem

Reference	Objective Function	Variables	Algorithm
Bosson and Sun, 2016	Minimize the sum of time deviation	Route, position and time on airport surface	Mixed integer linear programming
Desai et al., 2022	Minimize the sum of weighted time deviation	Route, position and time on airport surface	Mixed integer linear programming with group-and-release strategy
X.-B. Hu and Chen, 2005	Minimize the sum of airborne delay	Position and time on runway	Self-developed aircraft scheduling algorithm
Ma, Sbihi, and Delahaye, 2019	Minimize the sum of runway usage time and ground delay	Position and time on runway	Compares mixed integer linear programming and simulated annealing
Prakash, Desai, and Piplani, 2021	Minimize the sum of time deviation	Position and time on runway	Mixed integer linear programming with branch and bound
Prakash, Piplani, and Desai, 2018	Minimize the makespan	Position and time on runway	Mixed integer linear programming with branch and cut
Xiangwei, Ping, and Chunjin, 2011	Minimize the sum of time deviation	Position and time on runway	Mixed integer linear programming

Chapter 3

Problem Description

To whom do the benefits of FF-ICE R1 belongs to, and to what extent, would airspace operations benefit from incremental participation in the program? This section explores how we quantify the value of information sharing and collaboration for the implementation of FF-ICE R1, across various levels of participation among FIRs in the Association of Southeast Asian Nations (ASEAN) region, with the addition of the Sanya and Hong Kong region, which we recall as *ASEAN Plus*. We will first extend the discussion from Chapter 1, of the processes through which information is shared and the systems supporting it, justify why a decentralized model is appropriate, and follow up with a description of the decentralized models applied to answer the key question — *What is the value of information sharing and collaboration?*. We then conclude the chapter with by listing the data sources that allow us to construct a realistic simulator of flights in the ASEAN Plus region that can answer the question.

We work towards answering the key question, through modeling the decentralized ATFM problem with a rolling horizon framework, where multiple flights are updated iteratively at each time step. Flight plans are fixed with a predetermined sequence of waypoints for each flight, making the primary challenge to ensure that the trajectories, defined as the Target Time Over (TTO) for each flight at each node along its flight plan, satisfy separation constraints between all flights at all nodes. At any given time step, flights may be either en route or queuing for departure: Two main types of decisions are considered: (i) Speed regulation: the FIR controller may direct an aircraft to speed up or slow down from its desired speed over each leg of the trajectory, within limits determined by the aircraft type and its phase of flight (ascent, cruise, or descent); (ii) Delay management: the FIR controller may require a more significant delay than what can be achieved through speed regulation. In practice, this represents delaying a flight's departure, vectoring the aircraft to increase the distance to the next waypoint, or placing the aircraft in a holding pattern.

To address this problem, we develop a modeling framework consisting of two primary modules, which are further subdivided into individual algorithms and mechanisms (see Figure 3.1):

- UPDATE module: Updates the simulator and prepares for the next timestep.
  - Rolling Horizon Concept (RHC): Updates the current time window iteratively using a rolling time step (e.g., windows of 2 hours updated every 5 minutes). The RHC pulls in new active flights from a flight database, removes completed flights from the dataset, and reconciles flight paths. The reconciliation process is necessary because flights are updated independently in each FIR, potentially leading to inconsistent TTOs. This procedure is explained in Section 3.1.
  - Information Sharing: Updates flight information across FIRs for flights participating in the FF-ICE R1 information sharing program. The shared information may be used to enact collaborative ATFM initiatives such as ground delay programs and speed modulation.
- SOLUTION module: Solves the problem of regulating flow at both airports and waypoints.
  - Aircraft Flow Regulator (AFR): Assigns aircraft to channels, and schedules arrival and departure times for active flights nearing arrival or departure. Note that to make the approach easily generalizable, instead of modeling each airport's runway system (which can sometimes be a complex and unique system), the AFR adopts a more general framework. Each airport is modeled with three channels representing arrival-only, departure-only, and mixed operations. These channels are calibrated using capacity envelopes generated from historical data. The runway assignment procedure is explained in Section 3.3.
  - Waypoint Flow Regulator (WFR): Analyzes active flights and uses their initial planned TTOs, which may come from the previous time step (unless it is the flight's first active cycle). The WFR optimizes the TTOs at subsequent waypoints, independently for each FIR, by adjusting the speed and holding times for each flight leg to satisfy separation requirements.
  - Discrete Event Simulation (DES): Solves both the AFR and WFR simultaneously by utilizing an event queue to schedule flight legs sequentially. The DES will determine the entire flight trajectory, including the assignment of the arrival and departure channel, as well as the TTO at all nodes along the flight path.

Once flights are solved by the SOLUTION module to satisfy separation requirements, the output data is then fed back to the UPDATE module for flight paths to be reconciled, the list of active flights to be updated, and collaborative information to be

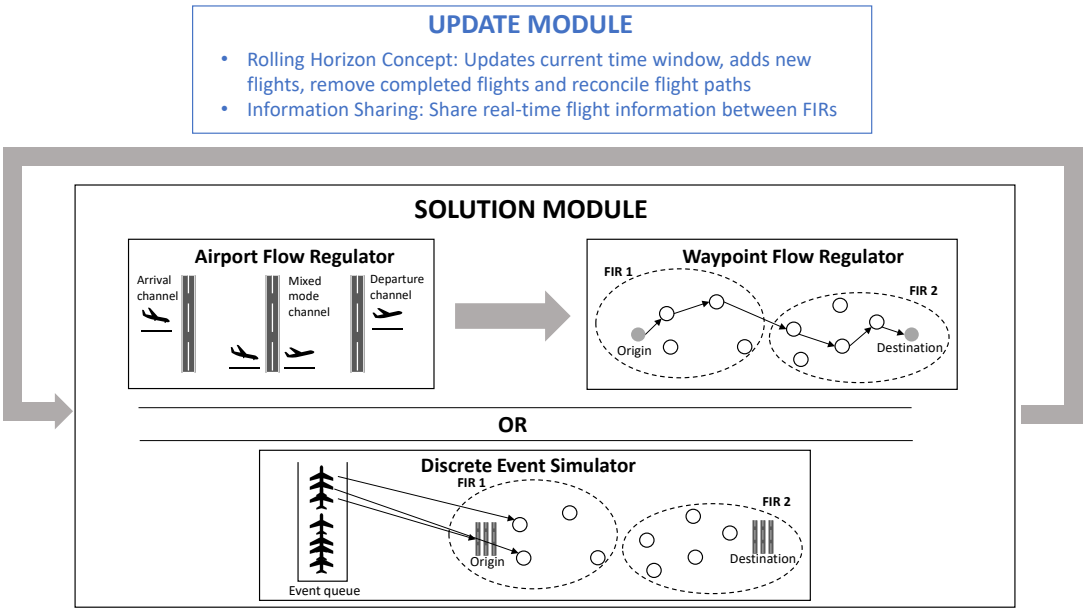

FIGURE 3.1: Modeling Framework for the Integration of the UPDATE and SOLUTION Modules and Their Submodules

shared. This process is repeated until a user-defined termination event is reached, or all flights are completed. Note that in this problem, we do not consider flight level separations. Our viewpoint is therefore flow-based, limiting the number of aircraft traversing each waypoint per unit of time. We continue the chapter by providing a description for each of the submodules, with the RHC and information sharing submodules explained fully in this chapter. The AFR, WFR, and DES are more algorithmically complex, hence a separate chapter is devoted to each of the algorithms.

3.1 Rolling Horizon Concept

This section extends the Rolling Horizon Concept (RHC) introduced in Chapter 2, with a particular focus on its implementation in the FF-ICE simulator.

Due to inconsistencies arising from decentralized decisions made by independent FIRs, we have opted for the RHC approach to accurately model this operationally complex task, with the reconciliation of flight trajectories to ensure feasible flight plans at regular simulated time intervals. Decentralized decisions often result in optimized TTOs at various waypoints within each FIR, however, inconsistent intraflight TTOs may arise when considering prior decisions made by upstream FIRs. To resolve these discrepancies, the problem is modeled as a rolling horizon framework, where TTOs are iteratively updated and reconciled at each step, as new information becomes available.

This approach aims to mimic how information is updated over time in real-world contexts, where both Air Traffic Control (ATC) and airlines may adapt their operations as they receive real-time information. Algorithm 1 outlines the steps of the RHC, using the AFR and WFR approach, while Figure 3.2 illustrates the underlying rationale and provides a simplified representation of how decentralized operations are modeled. The figure depicts two flights across two FIRs, both scheduled to arrive at FIR 1 simultaneously. Note however that in practice, the problem involves far more flights, FIRs, and time intervals, making it difficult to comprehensively capture in a single figure.

In this example, we assume that both aircraft have not yet entered FIR 1 or FIR 2, allowing all flight legs to be modified. The arrows in the lower half of each box (in green) represent the sequence of nodes traversed by Flight 1, while the arrows in the upper half of each box (in orange) represent the sequence of nodes traversed by Flight 2. There is only a single arc between any two nodes with minimum travel time of 10 minutes, and the minimum required separation for flights visiting the same node is 5 minutes.

At time step 1, which covers the time window from 00:00 to 02:00, the RHC starts by applying the AFR and WFR algorithms to independently optimize decisions for each FIR. Both nodes 3 and 4 have separation constraints violated, yet the decisions are made independently for each FIR since node 3 is managed by FIR 1 and node 4 is managed by FIR 2. In this example, the WFR delays Flight 2 in FIR 1 and Flight 1 in FIR 2, each by five minutes. The clock then advances (e.g., by five minutes), representing the time taken to update information across FIRs. When a new period begins, FIR 2 receives updated information that Flight 2 was delayed by FIR 1. As a result, Flight 2 is now expected to arrive at node 4 at 00:35 (00:30 + 5 minutes delay, considering the 10-minute minimum travel time from node 3 to node 4). This allows FIR 2 to re-optimize its previous decisions, adjusting Flight 1's timing back to 00:30 to avoid conflict with the updated timing of Flight 2.

It is important to note that in this simplified example with only two flights, FIR 2 can revise its decision for Flight 1 and restore its original schedule. However, in practice, the clock moves continuously, and other flights with updated timings may no longer be adjustable. These changes can introduce conflicts with flights already affected by earlier decisions, potentially preventing Flight 1 from returning to its original schedule. These dynamic updates highlight the decentralized nature of the problem, as independent decision-making can lead to situations where reversibility is no longer possible.

In summary, the RHC works as such. The simulation advances by a fixed time interval, advancing the simulation clock and rolling horizon. Active flights which do not fall into the new rolling horizon are removed, and new flights that fall into the new

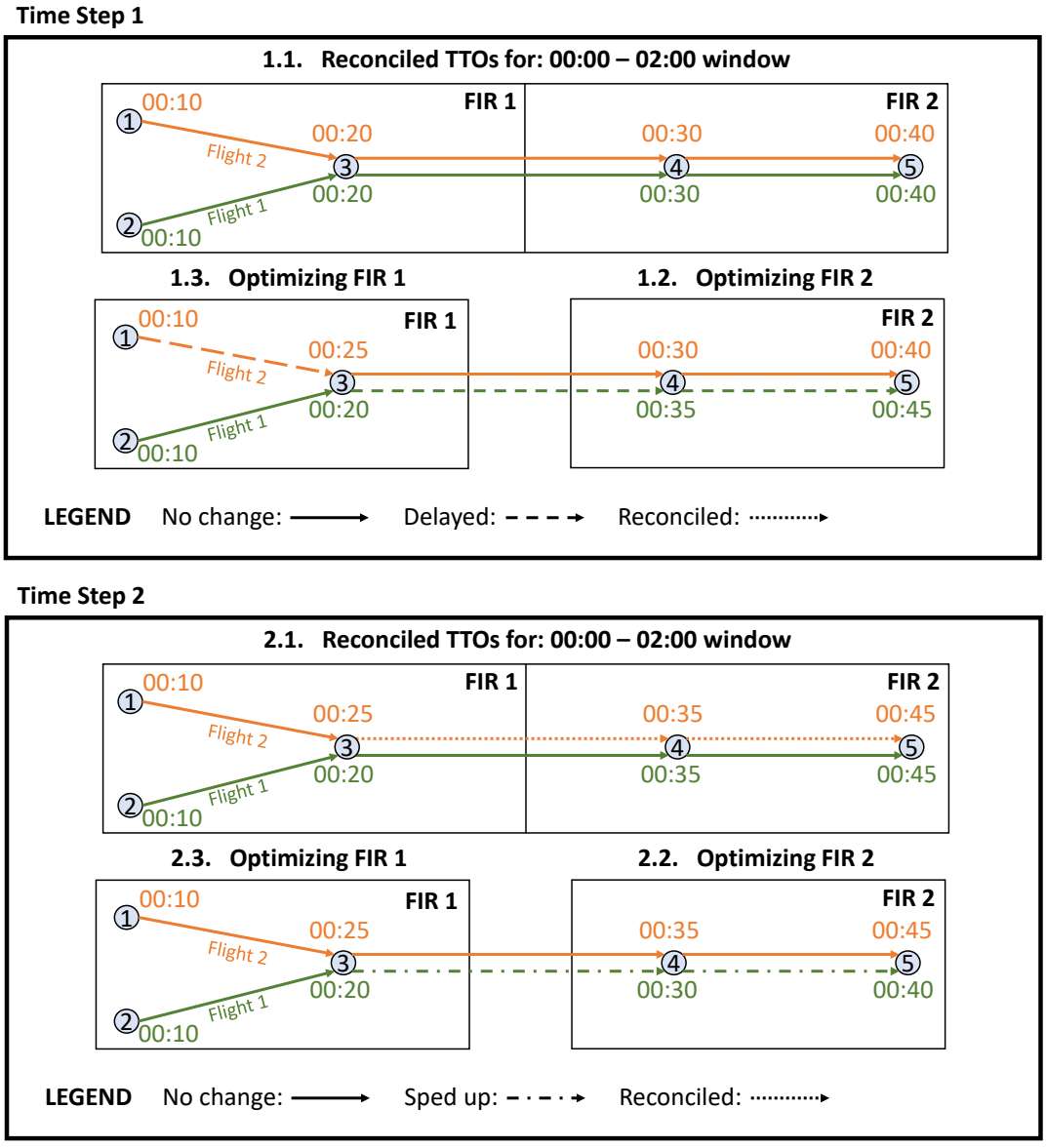

FIGURE 3.2: Example of the Rolling Horizon Concept for Two Time Steps

Algorithm 1: RHC ($\mathcal{F}_{db}$, $\mathcal{F}$, $\mathcal{F}_C$, n, t)

```
// F_db represents the complete set of flights in the database
// F represents the flights in current sliding window
// n represents the number of time steps to run for
// t represents the previous simulation time
// Δ represents the time shift (5 mins in our example)
while F ≠ ∅ AND n > 0 do
    // remove completed flights
    for flight in F do
        if flight sufficiently far in the past then
            F_C = F_C ∪ flight
            F = F − flight
        end
    end
    // add new flights
    for flight in F_db do
        if departure time > t + Δ then
            F = F ∪ flight
        end
    end
    // reconcile flights and share information
    for flight in F do
        use forward recursion to reconcile TTOs for entire trajectory for flight
        if flight is information sharing then
            use backward recursion to update backTTOs for entire trajectory for
              flight
        end
    end
    // designate channels for flights
    for airport in airports do
        run AFR on airport
    end
    // optimize each FIR, in a decentralized manner
    for FIR in FIRs do
        run WFR on FIR
    end
    n = n − 1
    t = t + Δ
end
```

rolling horizon are added to the list of active flights. We then reconcile flight legs, enforcing a minimum positive travel time between nodes. A transition from the current time step to the next, also involves updating the backTTO. This backTTO is conceptualized to assist a flight in adjusting its en-route and departure times. By setting the target time at each node in its flight path to meet the backTTO, delays at the destination airport can result in collaborative ground delays at the origin airport, in place of airborne holding, effectively reducing operational costs. Here, we define a generic function for forward recursion, as $TTO_i = TTO_{i-1} + d_i$, where the TTO at any node i depends on the TTO of the previous node $i-1$ and a duration d_i that is defined as the travel time from node $i-1$ to node i. Note that the forward recursion starts from the departure node, where $TTO_0 = departure\ time$. The backward recursion is similarly defined as $TTO_i = TTO_{i+1} - d_{i+1}$. Note that the backward recursion starts from the arrival node, where $TTO_N = arrival\ time$, and works backwards. Here, N is defined as the index of the last node.

The updated trajectories, along with the new flights from the database, are then handed over to the SOLUTION module, which advances each aircraft along its flight path, transits aircraft control from one FIR to another, allowing each FIR to update the trajectories under its control. These decisions are in turn is handed back to the UPDATE module, looping until termination of the program. The implementation details, system definition charts, and user interface are given in Appendix C.

Choosing an appropriate time shift for the RHC is the next step, and we thus refer to studies utilizing the RHC as mentioned in our literature review in Section 2.2. The studies by (Henry, Delahaye, and Valenzuela, 2022) and (Rapaya, Notry, and Delahaye, 2021) have look ahead time shifts of 7 minutes and 30 minutes respectively. Furthermore, the seminal paper on ATFM (Balakrishnan and Chandran, 2014) has also discussed the pros and cons of various time discretization values used in ATFM studies. Naturally, a shorter time shift better reduce any trajectory inconsistencies, however, the additional computational burden must be accounted for. One important consideration is that aircraft are not able to precisely follow a trajectory due to several factors such as the capabilities of the Flight Management System, weather and wind conditions, pilot behavior, etc. Moreover, from an operational standpoint, it is normal for ATCs to accommodate small deviations from the expected capacity using tactical maneuvers such as vectoring and speed changes (Balakrishnan and Chandran, 2014). Hence, this uncertainty in the ability of aircraft to conform to a precise 4D trajectory, coupled with the operational standard of ATCs, does provide leeway in allowing for a reasonably large time shift of a few minutes, with minimal impact on tactical flight operations.

3.2 Information Sharing

As described in Chapter 1, the primary aim of this dissertation is to investigate the value of information sharing and collaboration. In particular, to provide a preliminary quantitative analysis of the FF-ICE R1 concept, through modeling and simulating air traffic in the ASEAN Plus region, under FF-ICE R0, R1, and mixed mode operations.

We define an information regime, as the nature of information each flight provides to each FIR. These include arrivals, departures, and en route flights. The information we are concerned with include the original flight plans, the flight plan updated by departure and en route changes, and delays due to congestion at the arrival airport. The information may be specified as three distinct definitions of Target Time Over (TTO):

- The R0 TTO: the target time over a waypoint (or airport channel), under the original flight plan. This TTO, for all flights, is always available to all FIRs.
- The R1 TTO: the target time over a waypoint (or airport channel), updated by departure times and en route changes. This TTO, for both R0 and R1-level flights is available to any FIR which has current control of the aircraft. This TTO for R1-level flights are available to FIRs of which it passes through.
- The R1 BackTTO: the target time over a waypoint (or airport channel), back calculated from the desired arrival time at the destination airport. This TTO for R1-level flights is available to all FIRs through which it passes, and hidden for all R0-level flights.

Each flight through the region of interest is assumed to have a flight plan with a TTO assigned to each location on the plan. Initially, these TTO times would reflect an on-time departure and travel along each leg at the preferred speed for the designated equipment type and phase of flight. In the SOLUTION module, each FIR updates the planned TTOs for each flight, only for the locations within its boundaries as conditions and information change. In particular, under R0, information about a delayed departure is not communicated from the origin FIR to other FIR's visited on the flight plan. Consequently, in a subsequent FIR, if an aircraft fails to show up at a boundary entry location at its scheduled TTO, then that FIR will simply advance the TTO at that location to the current time (expecting the aircraft to arrive shortly) and adjust the other TTOs along the flight path within that FIR accordingly. Thus, in the absence of information sharing, the assembled flight plan, knitted together from the TTOs from different FIRs, can suffer major information inconsistencies.

A feature of the R1 release for FF-ICE is that actual departure times would be shared among FIRs operating at release level R1. Consequently, each such FIR at R1 would update the planned TTOs for locations within its boundaries with the updated departure

time. These updates happen at each timestep as part of the UPDATE module, updating TTOs and backTTOs for all flights participating in the information sharing program. For modelling simplicity, we extend this feature as follows. If a flight operates between two airports, both of which are in FIRs at release level R1, we refer to it as an *R1-level flight*. For a R1-level flight, any update to departure time or planned leg travel time is used to update the TTO for all subsequent locations on the flight plan in a consistent manner. That is, even if the flight passes through an FIR at level R0, the TTOs in that region will be updated with upstream information. This clearly violates the concept of R0-level sharing, but we justify the assumption by noting that most of the congestion in the network occurs at airports. Only a few waypoints experience congestion delays, and they are typically in the FIR of the origin or destination airport. Consequently, intermediate FIRs will experience little to no measurable benefit from such updated information. The alternative, that of modelling separate flight plans from the viewpoint of each FIR, is unnecessarily complex, and will not be considered in this study.

On the other hand, a major benefit of release level R1 is that it could facilitate collaborative ground-delay programs. To compute this, we work backwards from the planned arrival TTO for each R1-level flight and determine the TTO at each location which would enable the flight to achieve that arrival TTO using preferred flight durations on each subsequent leg. We refer to such a backward-calculated time as the backTTO, which is computed as follows:

$$t_{fj-1} = t_{fj} - \widetilde{t}_{fj}, \tag{3.1}$$

for all $j = 1, 2, \cdots, |P_f| - 1$. Here, t_{fj} denotes the target time over (TTO) at the j^{th} node in path P_f for flight f, and $\widetilde{t}_{fj}$ represents the preferred time of travel for this flight leg.

The backTTO at the origin airport for a flight is also known, in aviation terms, as the Calculated Take-Off Time (CTOT). If the flight departure could be delayed to the CTOT and nothing were to change en route, then the flight would not experience any airborne holding delays. In essence, the value of information sharing and collaboration would be displayed in the ability of the ground holding at the origin airport, to reduce the time a flight is airborne, while incurring as short a delay as possible at the destination airport. To further investigate the effectiveness of collaborative ground-delay programs, we later introduce a smoothing parameter, that adjusts the ground holding by a percentage of the CTOT.

3.3 Airport Flow Regulator

We now transition to the submodules of the SOLUTION module. In the Airport Flow Regulator (AFR) problem, we consider a modification of the Aircraft Sequencing Problem (ASP) for multiple runways, in a decentralized rolling horizon setting, with multiple flights either queuing for arrival or departure at any given time. We seek to formulate an optimization problem to regulate the flow of aircraft at airports.

To avoid confusion with the runway scheduling algorithms, we will recast the flow constraint for an airport in terms of three "channels": the arrival channel, the departure channel, and the common channel. Each channel has a fixed separation time chosen to satisfy the overall capacity envelope for the airport, where a capacity envelope specifies the maximum number of arrivals and departures at an airport, and the maximum number of operations if both arrivals and departures are present. The key idea behind this formulation is that if the flights satisfy the minimum required flight separation times, which are derived on the capacity envelopes, the overall flow of aircraft into and out of the airport will satisfy the capacity envelope. The derivation of these separation times are given in Section 4.1.

The goal of the algorithm is to regulate flow at airports via choice of a preferred runway, and runway time for each arrival or departure aircraft. This would be done through two decisions. The first, is runway selection, where an arrival aircraft is assigned to either the arrival or common channel, and a departure aircraft, the departure or common channel. Note that these three channels may not always be available for scheduling at any given airport. However, there will always exist at least one available runway for arrival aircraft, and one available runway for departure aircraft. The second decision variable is imposing of ground delays for departure aircraft, and airborne holding for arrival aircraft. We choose not to prioritize one over the other a priori. However, delays would preferentially be assigned to departures in practice, due to the higher operational cost of airborne holding as compared to ground holding. A discussion of how such a modification may be implemented, is given in Section 4.3.

Another statement of the AFR's objective is to match each flight to its desired arrival or departure time, while obeying all pairwise separation constraints. These desired times will differ based on the FF-ICE information systems (R0 or R1) for a given scenario. Each flight possesses an earliest and a desired runway time. These are static for R0 systems but dynamic for R1 systems. The desired runway time is a CTOT under R1 and can reflect destination airport (or en route) congestion. We imagine R0 systems reserve runway times for all flights based on the published flight plans, despite the uncertainty of arrivals.

Our interpretation of static earliest and desired runway times is determined under the assumption that flights are not privy to future delays that occur in other FIRs,

hence the desired time of arrival and departure would follow that of the published flight plan for the FF-ICE R0 information regime. In contrast, for the FF-ICE R1 information regime, flights would be privy to future delays that occur in other FIRs. Hence, the desired time of arrival and departure would follow that of the backTTO, which takes into account the updated trajectory information and can be used to enact ground delays. The backTTO, which is updated through the information sharing submodule, is defined as to exist for flights, which in its flight plan, have both its arrival and departure airports in an FF-ICE R1 FIR. We stress again that both the AFR and WFR are interlinked and each algorithm's inputs and outputs are used to update the other. A sample interaction is portrayed in Figure 3.3. The AFR is given authority over decision of runway choice and potential ground delays, as delineated by the double-lined box, and the WFR is given authority over TTOs along the entire trajectory, as delineated by the solid-lined box.

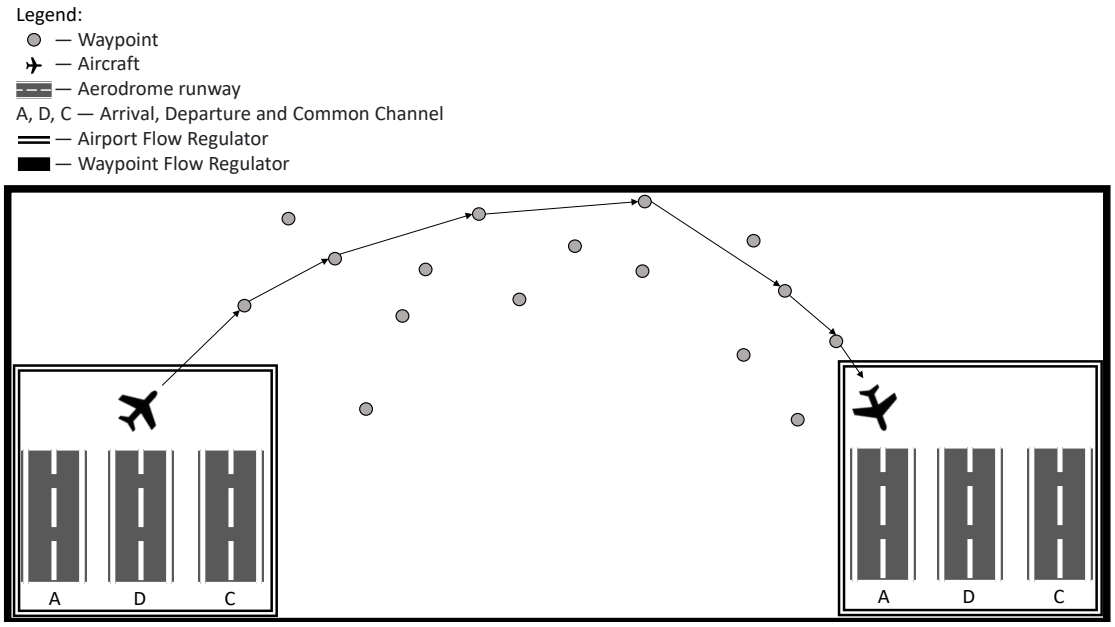

FIGURE 3.3: Iteration between the AFR and WFR

At the end of the AFR, each flight is assigned to one channel, either the departure or common channel at the origin airport, and one channel, either the arrival or common channel, at the destination airport. Once the arrival and departure times at any airport are determined by the AFR, these become the starting values for the WFR. For flights under the FF-ICE R0 regime, these arrival and departure times are computed based on the current TTOs, and are not enforced by the WFR. On the other hand, for flights under the FF-ICE R1 regime, these times are computed based on the backTTOs and are enforced by the WFR for departure flights, as a proxy for collaborative ground holding programs. An important point is that once a flight has been assigned to its channels

at the origin and destination airport, these channels can be treated in the same way as waypoint nodes in the network. This style of modelling the channel assignment and ground delay programs, allow us to consider the airport flow regulator separately from the WFR, in Chapter 5.

Two algorithms have been applied to the AFR problem. They are a novel queue pressure algorithm and the classic First-Come-First-Served (FCFS) algorithm. The AFR, together with the applied algorithms and their results, on a regular and reduced capacity scenario for published flight schedules in the ASEAN Plus region, are further detailed in Chapter 4.

The AFR would play a critical role in displaying the value of information sharing and collaboration, where we hypothesize that with more information, the en route airborne holding may be replaced by ground holding at an airport, leading to a reduction in operational costs.

3.4 Waypoint Flow Regulator

In the broader setting, we consider an ATFM model in a decentralized rolling horizon framework, with multiple flights either en route or queuing for departure at any given time. Assuming the flight plans are fixed in terms of the sequence of waypoints specified for each flight, the challenge at any given time is to ensure that the trajectories, which we define as the TTO of each flight at each node over the remainder of its flight plan, allow for adequate separation times between flights at each node. Here, a node refers to either an airport or waypoint in our flight network. We do not consider flight level separations or in-trail separations between waypoints. Hence, our viewpoint is flow-based, where a limit on the number of aircraft traversing each node per unit time is imposed. The problem setting is decentralized in the sense that each FIR is responsible for updating the trajectories of aircraft within its domain. We are led to such a formulation when formulating simulations of different information sharing regimes in decentralized air traffic management regions such as Southeast Asia.

Within the control of a single FIR, we imagine two forms of planning control over each trajectory. The first is a speed regulation: we imagine the FIR controller can require each aircraft to speed up or slow down from its desired speed over each leg of the trajectory, within limits determined by the type of aircraft and its phase of flight (ascent, cruise, or descent). The computation of speed limits are presented in Appendix B. For this dissertation, we allow speed changes throughout each phase of flight but recognize that a more realistic flow model would penalize each change. The second form of control is more extreme, it requires adding a delay to a trajectory leg of likely more than what could be accomplished with speed regulation. In practice, this could correspond

to vectoring the aircraft to increase the distance to the next waypoint or putting the aircraft into a holding pattern. We refer to this second control as a hold command and penalize its use in the optimization objective more heavily than speed regulation. The penalty for holding commands can be made specific to the waypoint. In this way, the traffic simulation manager can steer the optimization algorithm to concentrate holding or vectoring actions to specific waypoints in the airspace.

Flight plans originate and terminate at airports, and we do not treat airport nodes differently from waypoint nodes. Recall that in the AFR overview in Section 3.3, the AFR is liable for the channel selection at airports, and the initialized TTOs, which may include holding times, particularly for TTOs at the departure node. For the WFR, it is sufficient to recognize that a hold command applied to the first node of a trajectory would correspond to a request to employ a ground delay for the departing aircraft at the airport node. Since ground delays result in lower fuel consumption and lower ATC workload than airborne delays, we anticipate that the traffic manager will specify much lower penalties for holds at nodes which represent airports than at waypoint nodes.

Two algorithms have been applied to the WFR problem, Gradient Descent Ascent, and Simulated Annealing. The WFR, together with the applied algorithms and their results, on a regular and reduced capacity scenario for published flight schedules in the ASEAN Plus region, are further detailed in Chapter 5.

We posit that the WFR would be less critical than the AFR in showing the value of information sharing and collaboration, as the AFR has greater control of flight movement at the airport, including changing the runway and introducing ground holding. Nevertheless, the WFR can capture better resolution of where airborne holding takes place, and potentially the role of speed regulation. It could also capture the impact of waypoint constraints.

3.5 Discrete Event Simulation

A Discrete Event Simulation (DES) abstracts out the optimization aspect and, in place, runs a rule-based simulation. The setting will match that of the AFR and WFR, with the goal of determining channels at airport nodes, and TTOs for all nodes, inclusive of both the waypoint and airport nodes. The DES will also utilize the TTOs and back-TTOs in the same manner as the AFR/WFR model, with values are provided by the UPDATE module. We model the DES as an integrated flight scheduler, with the capability to both select channels at airports, and TTOs at all nodes, effectively performing the role of both the AFR and WFR. The results would not be directly comparable, between the AFR/WFR and the DES model, due to the airport channel and TTO choice

being separated in the AFR/WFR model. Most importantly and central to this dissertation, the DES yields promising results that are useful in distilling the value of information sharing and collaboration, particularly from the use of collaborative ground delays. The DES also provides a realistic operational scenario from the perspective of ATCs. For ATC operations under regular conditions, decisions are typically rule-based and straightforward (Ma, Delahaye, and M. Liang, 2024; Chandra, Choubey, and Verma, 2025). Deviation from a fixed schedule typically occurs in the event of inclement weather or emergencies such a request for priority landing because a passenger onboard requires immediate medical attention. Since such events are rare and overly complex to model, they would not be considered in this simulation. Furthermore, a comparison of the AFR/WFR against the DES model provides insight on how a separation of concerns, between the airport channel and TTO selection, may affect delays.

In the main loop of the DES, we process all events in non-decreasing order of earliest available time. We then employ a First-Scheduled-First-Served (FSFS) algorithm, where in contrast to FCFS, flight events are processed in order of their scheduled time, rather than the time of which they arrive at a given node. This scheduling strategy is applied exclusively to R1-level flights. The FSFS is not new. We have found it similarly applied in other studies including (Ball et al., 2001; Bertsimas and S. Gupta, 2011; Wambsganss, 2001). We believe a FSFS policy conducted in tandem with ground delays are necessary because no flight would appreciate having ground delays imposed on it. However, if there is a benefit of node access priority, which reduces airborne delays or holding time, airspace users are then incentivized to adopt the collaborative policy.

Events may refer to either flight events or resource events, where a resource refers to a unique node-channel pair. Flight events first assign a TTO to the current flight leg that is being processed, subsequently adds a resource event to mark when the node will next be available, and finally adds an event to mark the next leg in its flight path as pending, if any. Resource events signify times at which a resource is available. An event indicating a specific resource is available, will trigger a check to assign the next available flight to this resource, if any. At the end of processing all events, the DES will have generated conflict free TTOs for all flights within the current simulation step, under the FSFS logic for R1-level flights, and FCFS logic otherwise.

3.6 Data Sources

Our flight network, consisting of the nodes and arcs in the ASEAN Plus region, are derived from the Aeronautical Information Publication (AIP) documents published by each countries' civil aviation authority. Next, we determined the FIR boundaries, using

geographical data from ICAO. The flight schedules were obtained from OAG and contain historical information on the origin and destination for each aircraft, together with the scheduled arrival and departure times, for a period in late October 2023. Using the nodes, arcs, and the origin and destination airports, we knitted together flight paths for all aircraft, using Dijkstra's shortest path algorithm. We provide a plot of the network of nodes and arcs, together with FIR boundaries, in Figures 5.1 and 5.2, later in Chapter 5. Finally, flight profiles for each equipment type were obtained from the EUROCONTROL Aircraft Performance Database (EUROCONTROL, 2024). The dataset used for our experimental runs are published flight plans for flights arriving to, or departing from, the ASEAN Plus region for seven days starting from 1 Oct 2023.

Chapter 4

Airport Flow Regulator

On a field visit to an Air Traffic Control (ATC) tower at Changi Airport, Singapore, we had the opportunity to chat with an ATC officer, whose role is to strategically handle aircraft movement at Changi Airport, prioritizing safety and efficiency. Throughout the tour, we got to see the day-to-day operations of ATC officers, leading us to develop a clearer understanding of a day in the life of an ATC officer, the tools they use, and some of the challenges they face. ATC officers operate on a shift work basis, and the first thing ATC officers see upon starting their shift is a stream of arriving and departing flights. It is their primary role to provide instructions to pilots on which runway to use for take off or landing, necessary ground or airborne holding procedures, and offer guidance on potential disruptions in the surrounding airspace. Often, it is the stream, either arrival or departure, with the greatest "pressure", that is prioritized for runway use. Here, stream pressure is defined as the number of flights requesting use of the runway in the near future. This interaction motivated our formulation of the Airport Flow Regulator (AFR), and in particular, the queue pressure algorithm.

Most of the time, air traffic control (ATC) officers adhere to the conventional First-Come-First-Served (FCFS) sequencing of aircraft. As the name of the algorithm implies, aircraft that are ready for arrival or departure, will be sequenced on the appropriate runway according to the requested runway time that was provided to the ATC officer. This time is often known as the estimated time of arrival or departure. However, the FCFS algorithm is known to be inefficient in practice, causing unnecessary system delays (Prakash, Desai, and Piplani, 2021). We highlight that the key difference between scheduled time and estimate time, is that the scheduled time is based on the flight plan, and is fixed in advance, while the estimated time may be updated as more accurate information is received on the flight status. In this chapter, scheduled time and estimated time would be used for scheduling flights under the FF-ICE R0 and FF-ICE R1 information regime, respectively.

As described in Section 3.3, the AFR will be modeled as a flow problem, rather than a runway sequencing problem. The concept for both problems are identical, except that the flow regulation problem models all flights with identical required separation

times, while the runway sequencing problem has greater fidelity, with aircraft weight classes influencing the required separation times. The flow regulation formulation was selected as it aligns more closely with the overarching theme of this dissertation on ATFM, making it a more appropriate modeling choice in this context. As such, the distance and time based separation for aircraft of various weight classes (ICAO, 2016), would be replaced by our computations of a minimum required separation time, derived based on the published flight plans for each airport. The AFR would decide which channel, either the arrival, departure or common, is to be assigned, as well as the ground or airborne holding time, if any, for all active aircraft. While no objective function is explicit in both the Queue Pressure (QP) and FCFS algorithm, the performance of both algorithms would be measured by the makespan of the available channels, where a lower makespan is preferred, as this indicates improved capacity utilization.

We first proceed with the derivation of minimum required separation times from capacity envelopes, then the mathematical formulation, followed by the QP and FCFS approaches. Finally, we compare the performance of the QP algorithm against the FCFS algorithm for a regular and reduced capacity scenario based on published flight schedules in the ASEAN Plus region.

4.1 Capacity Envelope

A capacity envelope represents the amount of flow that can pass through a resource per unit time. This is often approximated as a convex polygon that defines the maximum number of arrivals, departures and total operations for airports, and throughput for sectors. In studies done by Balakrishnan (Balakrishnan and Chandran, 2014) and separately, by Tan (Tan, 2021), capacity envelopes have been modeled both at the sector level, and the airport level. Capacity envelopes are dictated by factors including but not limited to weather conditions, configuration of active runways, and ATC workload. For the purpose of this chapter, we focus only on capacity envelopes for airports, since most of the airspace congestion occurs near airports. A notable exception is the Afghanistan airspace whose capacity limits traffic between Europe and Southeast Asia. However, we also note that the same computations can be identically applied to waypoints or airspace sectors.

Figure 4.1 depicts the capacity envelope of Changi Airport's runway activity in 15-minute time increments. The horizontal and vertical axis represents the number of arrivals and departures per 15 minutes, respectively. The size of the circles represent the frequency of time blocks where the arrival-departure combination was observed. The capacity envelope is then the convex hull of the plotted points. For modeling reasons, we do not use the convex hull, but instead use the intersection of the maximum flow of

arrivals, departures and total operations. An illustration of the intersection is given in the shaded area of Figure 4.2.

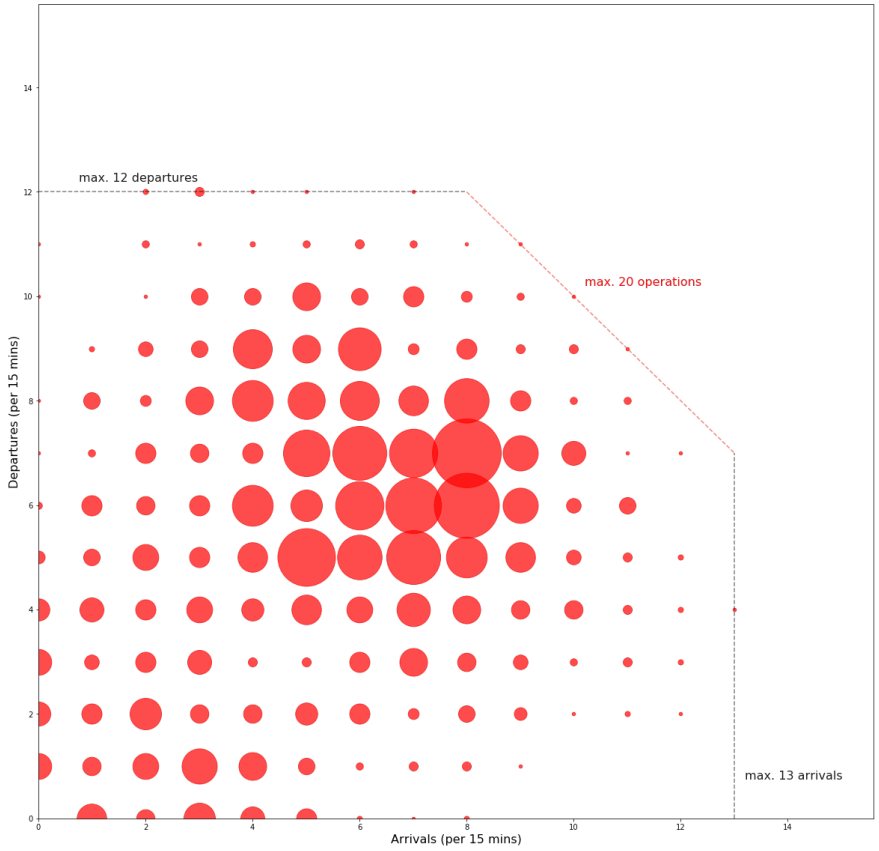

FIGURE 4.1: Runway Activity Density Plot for Changi Airport, Singapore (SIN/WSSS) (Tan, 2021)

For Figure 4.2, we denote x and y as the number of arrivals and departures per unit time, respectively. The parameters $\overline{x}$, $\overline{y}$ and $\overline{z}$ then denote the maximum number of arrivals, departures and total operations per unit time respectively. We note that if the maximum number of operations $\overline{z}$ is greater than the sum of the maximum number of arrivals and departures $\overline{x} + \overline{y}$, then the diagonal constraint $x + y \leq \overline{z}$ will be redundant and the feasible shaded area will be just a rectangle. Conversely, if the maximum number of operations per unit time is lower than either the maximum number of arrivals or departures, the diagonal line could be shifted so far to the left that the feasible shaded area gets transformed into either a trapezium or triangle. We note, however, that this should never happen in practice. These possibilities are captured using the minimum function in the following mathematical formulation of the capacity envelope in the next paragraph.

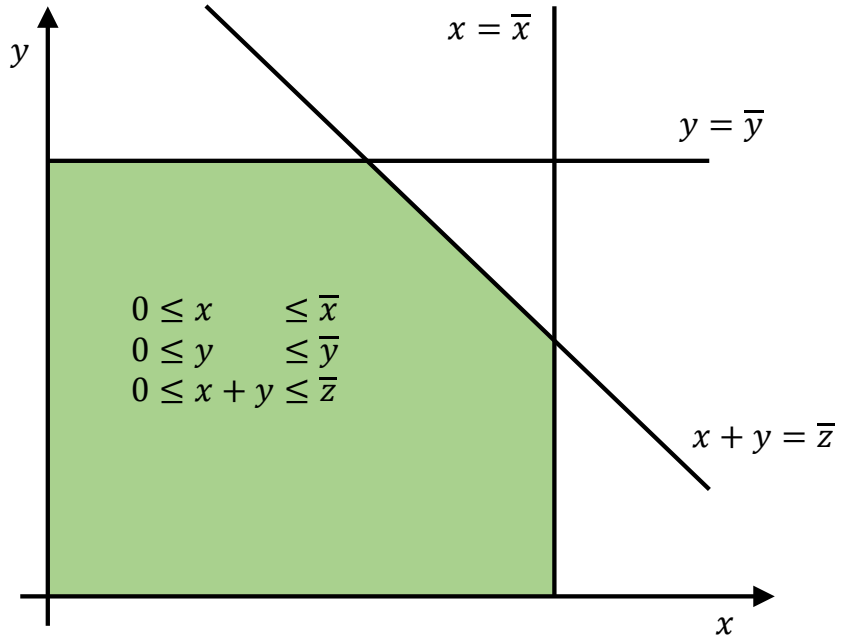

FIGURE 4.2: Capacity Envelope as Intersection of Lines

We propose for the capacity envelope to be mathematically equivalent to a three-runway system (or a three-channel system using prior terminology), using flow regulation at the pre-tactical planning phase. We supplement the following notation: a, b and c shall denote the minimum required separation time between aircraft on the arrival, departure, and common channel respectively. Also, the parameters $\overline{x}$, $\overline{y}$ and $\overline{z}$ shall denote the maximum number of arrivals, departures and total operations per unit time respectively, where we define a single unit of time, or time interval to be h. For each time interval h, the following equations hold:

$$\frac{h}{a} + \frac{h}{b} + \frac{h}{c} = \min\{\overline{z}, \overline{x} + \overline{y}\}, \tag{4.1}$$

$$\frac{h}{b} + \frac{h}{c} = \min\{\overline{x}, \overline{z}\}, \tag{4.2}$$

$$\frac{h}{a} + \frac{h}{c} = \min\{\overline{y}, \overline{z}\}. \tag{4.3}$$

The first equation can be interpreted as: the sum of the maximum number of aircraft utilizing the arrival $\frac{h}{a}$, departure $\frac{h}{b}$ and common $\frac{h}{c}$ runway are equal to the minimum

of the maximum total number of operations $\min\{\overline{z}, \overline{x} + \overline{y}\}$. The other maximum departures and arrivals are represented similarly by the other two equations. We subsequently rearrange the equations to solve for the separation times:

$$a = \frac{h}{\min\{\overline{z}, \overline{x} + \overline{y}\} - \min\{\overline{x}, \overline{z}\}}, \tag{4.4}$$

$$b = \frac{h}{\min\{\overline{z}, \overline{x} + \overline{y}\} - \min\{\overline{y}, \overline{z}\}}, \tag{4.5}$$

$$c = \frac{h}{\min\{\overline{x}, \overline{z}\} + \min\{\overline{x}, \overline{z}\} - \min\{\overline{z}, \overline{x} + \overline{y}\}}, \tag{4.6}$$

where division by zero is interpreted as the unavailability of the corresponding channel. These channel separation times can now be used as parameters in the AFR algorithm. Note that for the common channel, the minimum required separation time is applied equally regardless of whether the leading or trailing flight is an arrival or departure, aligning with our modeling of a flow regulator.

The bounds on the capacity envelope are derived from OAG historical data (OAG, 2024b), in the same manner as done by Tan (Tan, 2021). The takeoff and landing times for each aircraft were used if the data are available, otherwise they are approximated by adding or subtracting a five-minute taxi duration to the scheduled departure or arrival time at the gate, respectively.

The key point is that if the flights are assigned a channel time that satisfy the flight separation times in the three channels, then the overall flow of aircraft into and out of the airport will satisfy the given capacity envelope. The suitability of modeling capacity envelopes as channel separation times are found to be reasonable, with the results provided in Section 7.1.

4.2 Mathematical Formulation

As described in Section 4.1, the capacity envelope has been transformed into an equivalent minimum required separation time in the context of a flow regulator. The minimum required separation times are applied identically to all flights using the specified runway, independent of the weight class or operation type of the leading or trailing aircraft.

We let separation times be denoted by $\delta_p, p \in \{A, D, C\}$ with δ_A, δ_D, and δ_C representing the terms from the previous notation of a, b, and c — the minimum required separation time at the arrival, departure and common channel respectively. We express the time an aircraft uses a channel using the variable t_f^p to represent the time of an aircraft at channel p, and of flight index f. We have $p \in \{A, D, C\}$ for the arrival,

departure or common channel, and $f \in \mathcal{F}$, with $\mathcal{F}$ representing the set of aircraft to be sequenced at the given airport. Note that an aircraft would depart from one airport and arrive at another, pairwise distinct airport. Therefore, in the AFR formulation, for the same flight f, its flight index f is not necessarily the same at both airports.

The AFR problem shares similar attributes to the Job Shop Scheduling Problem (JSSP). For the JSSP, n jobs must be scheduled on m machines in a way that minimizes the makespan, while accounting for the processing time for each job. For the AFR problem, $|\mathcal{F}|$ aircraft must be scheduled on one of three different channels $p \in \{A, D, C\}$, while accounting for the minimum required separation time between each aircraft. Both are NP-complete problems and combinatorial in nature. We draw inspiration from the paper (Adams, Balas, and Zawack, 1988), on a bottleneck procedure for the JSSP, through selecting flights to schedule based on the bottleneck queue, and comparing the results against an FCFS scheduling algorithm.

For the AFR, we make the assumption that no early landing or takeoff are possible, and no upper bound is placed on the time of landing or take off. The second point is justified by the fact that we do not allow flight cancellations. Additionally, since in both the queue pressure and FCFS algorithms, flights are scheduled based on their position on either the arrival or departure queue, and the separation time is independent of aircraft weight category or operation type, no particular flight will be inequitably delayed. When the mixed channel is selected, both algorithms only decide whether to schedule an arrival or departure flight, that is at the front of its respective queue. Since the order of either the arrival or departure flights are never changed, the order precedence relationship for both queues hold. The proof of the invariability of the order precedence relationship is provided in Proposition 4.2.1.

Proposition 4.2.1. *If the scheduled arrival or departure time for flight i is earlier than that of flight j, where flight i and j are of the same operation type, flight i will be scheduled by either algorithm at a time not later than flight j. This maintains the order precedence relationship for flights of the same operation type under both scheduling algorithms.*

Proof. Given that flight i and j are of the same operation type, they will be placed in the same queue. Assume this to be the arrival queue, with the scheduled arrival time for flight i to be earlier than that of flight j. For the arrival queue, the flight with the earliest scheduled time in the queue, is invariably prioritized to be scheduled next by both scheduling algorithms, when a channel suitable for arrivals is chosen. Given that we pick the channel with the earliest available time to schedule a flight, s_p and say, we fix flight i's channel time at time s_p. Subsequently, the next earliest available time to schedule an arrival flight, say time $s_{p'}$, will never be earlier than s_p, that is, $s_p \leq s_{p'}$. This implies that when the scheduled arrival time of flight i is less than that of flight j,

flight i will always be assigned a channel time no later than that of flight j. The proof for the departure queue follows the same line of reasoning. □

The parameters for the AFR are the earliest available aircraft time $\underline{t}_f$, the earliest available channel time s_p for each channel $p \in \{A, D, C\}$, and the minimum required separation times δ_p for each channel $p \in \{A, D, C\}$. The decision variables are the indicator variable for whether flight f uses channel p, $\mathbb{1}_{fp}$, and the time at which flight f uses channel p, t_f^p. Here, we define $\mathbb{1}_{fp}$ as:

$$\mathbb{1}_{fp} = \begin{cases} 1 & \text{if flight } f \text{ is assigned to channel } p, \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

The first set of constraints in the AFR problem is that all aircraft observe the minimum required separation between aircraft, for all channels. This may be formulated as:

$$|t_f^p - t_{f'}^p| \geq \delta_p \ \ \forall f \in \mathcal{G}_p, p \in \{A, D, C\}, \tag{4.8}$$

where $\mathcal{G}_p = \{f : \mathbb{1}_{fp} = 1\}$, that is, the set of flights of which have been assigned to channel p.

The next set of constraints is that each flight must be assigned to exactly one channel. This is captured with the following constraint formulation:

$$\sum_{p \in \{A,D,C\}} (\mathbb{1}_{fp}) = 1 \ \ \forall f \in \mathcal{F}, (f, p) \in \mathcal{G}. \tag{4.9}$$

The next set of constraints prescribe the possible assignment of flights to channels. Let $\mathcal{F}_A$ and $\mathcal{F}_D$ denote the set of arrival and departure flights, respectively, we then have that:

$$\mathbb{1}_{fA} = 0 \ \ \forall f \in \mathcal{F}_D, \tag{4.10}$$

$$\mathbb{1}_{fD} = 0 \ \ \forall f \in \mathcal{F}_A, \tag{4.11}$$

which designates that arrivals are prohibited from being assigned to the departure channel, and similarly, that departures are prohibited from being assigned to the arrival channel.

All flights are also constrained to be scheduled only at or after its earliest aircraft time $\underline{t}_f$ using:

$$t_f^p \geq \underline{t}_f \ \ \forall p \in \{A, D, C\}, f \in \mathcal{F}. \tag{4.12}$$

The final set of constraints are the channel availability constraints. Let $\mathcal{P}^0$ denote the set of channels at an airport that are unavailable, where an unavailability is defined as when the denominator of the equations for a, b and c in Section 4.1 is equal to zero.

The channel availability constraints can then be described as:

$$\mathbb{1}_{fp} = 0 \;\; \forall f \in \mathcal{F}, p \in \mathcal{P}^0. \tag{4.13}$$

The objective function, would be to minimize the makespan, mathematically stated as:

$$\underset{\mathbb{1}_{fp}, t_f^p}{\text{minimize}}\; g(\mathbb{1}_{fp}, t_f^p) = \sum_{p \in \{A,D,C\}} (s_p - s_0), \tag{4.14}$$

where s_0 denotes the current simulation time, that is, the earliest time the first flight may be sequenced, and s_p here denotes the earliest available time to schedule the next flight on channel p, after the AFR algorithm has run to completion for the given airport.

The AFR problem may then be summarized as to determine values for the choice of channels $\mathbb{1}_{fp}$ and flight channel time t_f^p for all flights $f \in \mathcal{F}$, subject to all constraints specified above, such that the makespan is minimized.

4.3 Queue Pressure

We devise a queue pressure algorithm to regulate flow at airports, based on the idea of identifying and prioritizing scheduling of bottlenecks. This bottleneck concept has been applied in (Adams, Balas, and Zawack, 1988), and is also a natural way of scheduling tasks. Say, for example, the post office is assigned the task of processing parcels, and the driver is assigned the task of delivering these parcels to their destination. If we assume that the post office is very quick at processing parcels, and has more parcels ready for delivery than the driver can handle, the driver is the bottleneck of this system. A natural response would be to target improvements at the bottleneck, for example, hire more drivers. In the same vein, we devise the queue pressure algorithm along the overarching theme of identifying which queue, either the arrival or departure queue, is a bottleneck, and prioritize the scheduling of the next aircraft from this queue.

We first split the flights by their operation type into two queues, the arrival and departure queue, which are both to be sorted by the earliest available aircraft time $\underline{t}_f$. We remove flights from their respective queues once they have been scheduled.

Given that there are still flights to be scheduled, in either the arrival or departure queue, or both, the queue pressure algorithm first selects the channel p with the earliest available time s_p:

$$p = \underset{p}{arg\,min}\{s_p\} \tag{4.15}$$

Here, unavailability of a runway will have its earliest available time s_p set at time infinity, so it will never be selected at this step.

If the selected channel p is either the arrival or departure channel, and if the corresponding queue is non-empty, assign the next flight from the corresponding queue to this channel. Set the earliest available channel time at the latter time between the earliest channel available time s_p and the earliest available aircraft time $\underline{t}_f$, of the first aircraft in queue q, plus the required separation time at this channel δ_p,

$$s_p = max\{s_p, \underline{t}_f\} + \delta_p \text{ if the queue } q \text{ is non-empty.} \tag{4.16}$$

Also, set the holding time of this aircraft, which is to be passed into the waypoint flow regulator, as the time that $\underline{t}_f$ must be delayed by to be scheduled on the channel at the time s_p. The holding time is conditioned to be non-negative, that is, the holding time will be set at $\max\{0, s_p - \underline{t}_f\}$. As the holding time is not used by the AFR, we do not notate it here, and we just point out that it first appears in Section 5.1 as h_{fj}.

If either the arrival or departure channel is selected, and its corresponding queue is empty, we set the earliest available channel time to a large value such that it does not get selected again for the current iteration of the algorithm,

$$s_p = INF \text{ if the queue } q \text{ is empty.} \tag{4.17}$$

If the common channel is selected, more thought is required, as it may be feasible to schedule either an arrival or departure flight. First, if either the arrival or departure queue is empty, the choice is obvious. We schedule an aircraft from the non-empty queue. However, when both queues have remaining aircraft to be scheduled, we first perform a preliminary check to see if the first aircraft in a queue may be assigned without delaying aircraft from the other queue. This also leads to an obvious choice, to schedule the first aircraft from the queue that does not delay aircraft from the other queue. We summarize this choice as scheduling an aircraft from the departure queue if the following condition holds:

$$max\{s_p, \underline{t}_f\} + \delta_p \leq \underline{t}_{f'}. \tag{4.18}$$

Here, f and f' represent the flight index of the first flight in the departure and arrival queue respectively. We schedule an aircraft from the arrival queue under the same condition, but instead let f and f' represent the flight index of the first flight in the arrival and departure queue, respectively. The scheduling is done in the same manner as above.

The queue pressure is computed by provisionally assigning flights from a single queue q, on any of its available channels $\mathcal{P}^q$, until a gap in time occurs, while keeping track of the sum of delays. We denote the flights that are in queue q, with no gaps to yet occur, as $\mathcal{F}^q$. A gap is defined as occurring when an aircraft with its earliest available

aircraft time $\underline{t}_f$ is strictly later than the earliest available channel time s_p, that is, $\underline{t}_f > s_p$. A delay is defined as $s_p - \underline{t}_f$ where $\underline{t}_f \leq s_p$. Hence, we define the queue pressure, for queue q as:

$$\pi_q = \sum_{p \in \mathcal{P}^q} \sum_{f \in \mathcal{F}^q} \left(s_p - \underline{t}_f \right), \tag{4.19}$$

where s_p is updated at every provisional assignment in the same manner as above.

If the preliminary check fails, we employ the concept of queue pressure to select the queue from which to schedule the next flight from. Only when we get to this point, does the queue pressure algorithm differ significantly from the FCFS algorithm. For each queue, we provisionally assign aircraft to either the common channel, or its dedicated arrival or departure channel, if available, and compute the queue pressure π_q for each queue $q \in \{A, D\}$. We then schedule the first flight from the queue with the highest queue pressure. If the queue pressure of both queues are equal, we then compare the earliest aircraft time of the first flight in both queues and schedule the flight with the earlier, earliest aircraft time. If the earliest aircraft time for both queues are equal, then we arbitrarily schedule the flight from the arrival queue first. This is because, all else being equal, arrivals should be prioritized over departures due to airborne holding for arrivals, being operationally more expensive than ground holding for departures.

Figure 4.3 illustrates computation of queue pressure for the departure queue. The solid bars represent the minimum required separation time, the striped bars represent the delays, and the stacked striped bars represent the sum of delays, that is, the queue pressure. The aircraft symbols at the right represent the earliest time an aircraft can be scheduled $\underline{t}_f$, with a lower position representing an earlier time. The gray aircraft have been scheduled in a previous iteration, and black aircraft are being provisionally assigned in the current iteration. The top of each solid bar delineates the earliest available channel time at each iteration s_p. To illustrate a computation for delay, the highest horizontal line, at the top of the black bar above flight 11, represents the value of s_p at the appropriate iteration, and the second highest horizontal line, aligned at flight 13, is the earliest aircraft time $\underline{t}_{13}$, and the delay at this iteration is equal to $s_q - \underline{t}_{13}$. Also, the left brace with the word "GAP" denotes that a gap occurred, when the departure channel was selected, and the next aircraft to be assigned would not be delayed, that is, $\underline{t}_{14} > s_p$. Intuitively, this means that any flight, from flight 14 and onward, are not part of the current queue and do not contribute any queue pressure at the current assignment step.

We summarize the queue pressure algorithm in Algorithm 2. Algorithm 3 is called if the mixed channel is chosen, and assigns flights based on the queue pressure computation, which is called by Algorithm 4. We also have Algorithm 5 and 6 to assign flights in the global and provisional computation environment, respectively. Here, we denote

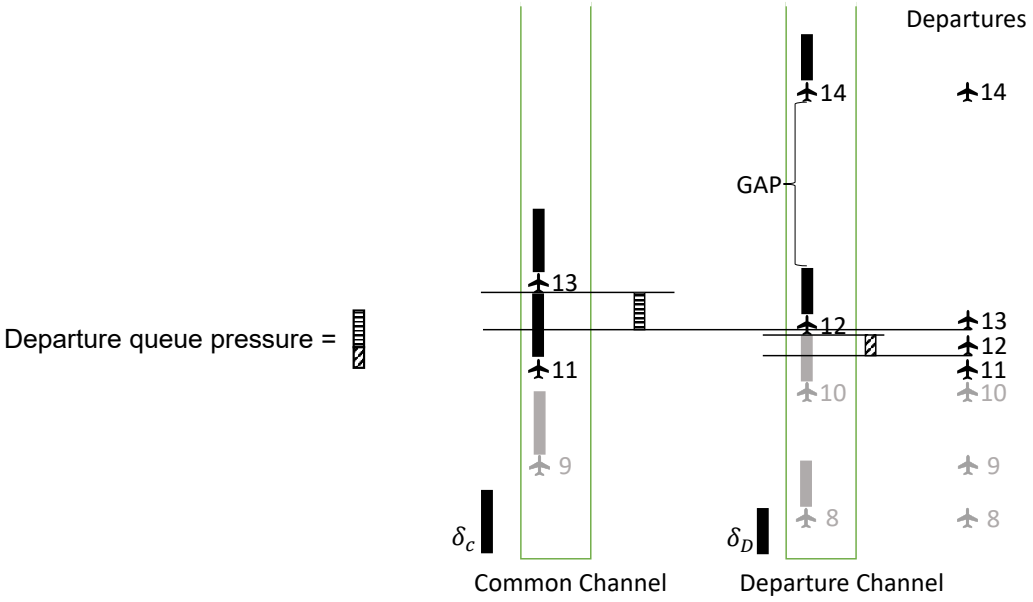

FIGURE 4.3: Illustrative Computation of Queue Pressure for the Departure Queue

$\mathcal{F}$, $\mathcal{F}_A$, and $\mathcal{F}_D$ to contain the information regarding a flight's earliest available aircraft time $\underline{t}_f$, and the operation type $q \in \{A, D\}$. Denote $\mathcal{F}_{new}$ as a construct in the global environment to record the channel assigned and hold time for each flight. We also require $tmp\_\mathcal{F}_p$ to represent the provisional queue, such that provisional flight assignments are done separately from actual flight assignments. Also, let s_p for all channels $p \in \{A, D, C\}$ be global variables available to all algorithms. We treat tmp_s_p in the same way, except it s_p is used for actual flight assignment, while tmp_s_p is used for provisional flight assignment.

We end this section by addressing the possibility of prioritizing arrivals over departures, as mentioned in Section 3.3. A variation on the queue pressure scheduling rule would be to apply a weighting factor to give priority to the arrival queue, since the fuel costs of an airborne delay are higher than those of a grounded aircraft. For example, one possible modification would be multiplying the arrival queue pressure by a weighting factor greater than 1, such as to increase the priority of the arrival queue.

Algorithm 2: `Queue Pressure Algorithm(`$\mathcal{F}$`)`

$\mathcal{F}_{new} \leftarrow \emptyset$;
$\mathcal{F}_A \leftarrow \mathcal{F}$ for which operation type $q = A$;
$\mathcal{F}_D \leftarrow \mathcal{F}$ for which operation type $q = D$;
sort($\mathcal{F}_A$); sort($\mathcal{F}_D$);
while $\mathcal{F}_A \neq \emptyset$ *OR* $\mathcal{F}_D \neq \emptyset$ **do**
  $p = \underset{p}{arg\,min}\{s_p\}$; `// an unavailable runway would have` $s_p = INF$
  if $p == A$ **then**
    if $\mathcal{F}_A == \emptyset$ **then**
      $s_A = INF$
    else
      `Assign Flight(`$\mathcal{F}_A, A$`)` ; `// Assign flight from arrival queue to arrival channel`
  else if $p == D$ **then**
    if $\mathcal{F}_D == \emptyset$ **then**
      $s_D = INF$
    else
      `Assign Flight(`$\mathcal{F}_D, D$`);`
  else
    if $\mathcal{F}_A == \emptyset$ *OR* $max\{s_p, \underline{t}_f\} + \delta_p \leq \underline{t}_{f'}$ **then**
      `//` f `as first flight in departure queue and` f' `as first flight in arrival queue`
      `Assign Flight(`$\mathcal{F}_D, C$`);`
    else if $\mathcal{F}_D == \emptyset$ *OR* $max\{s_p, \underline{t}_f\} + \delta_p \leq \underline{t}_{f'}$ **then**
      `//` f `as first flight in arrival queue and` f' `as first flight in departure queue`
      `Assign Flight(`$\mathcal{F}_A, C$`);`
    else
      `Queue Pressure Assign(`$\mathcal{F}_A, \mathcal{F}_D$`);`
end
return $\mathcal{F}_{new}$

Algorithm 3: `Queue Pressure Assign(`$\mathcal{F}_A, \mathcal{F}_D$`)`

$\pi_A =$ `Get Queue Pressure(`$\mathcal{F}_A$`);`

$\pi_D =$ `Get Queue Pressure(`$\mathcal{F}_D$`);`

if $\pi_A > \pi_D$ **then**
  `Assign Flight(`$\mathcal{F}_A, C$`);`

else if $\pi_A < \pi_D$ **then**
  `Assign Flight(`$\mathcal{F}_D, C$`);`

else if $\underline{t}_f < \underline{t}_{f'}$ **then**
  `// with` f `as first flight in arrival queue and` f' `as first flight in departure queue`
  `Assign Flight(`$\mathcal{F}_A, C$`);`

else if $\underline{t}_f < \underline{t}_{f'}$ **then**
  `// with` f `as first flight in departure queue and` f' `as first flight in arrival queue`
  `Assign Flight(`$\mathcal{F}_D, C$`);`

else
  `Assign Flight(`$\mathcal{F}_A, C$`);`

Algorithm 4: `Get Queue Pressure(`$\mathcal{F}_q$`)`

$f \leftarrow$ index of first flight in queue q, that is, $\mathcal{F}_q$;

$\pi_q \leftarrow 0$;

$tmp\_s \leftarrow s$; `// initializes the provisional earliest channel time at all channels`

$tmp\_\mathcal{F}_q \leftarrow \mathcal{F}_q$; `// initializes the provisional flight queue`

while $tmp\_\mathcal{F}_q \neq \emptyset$ *AND* $\underline{t}_f > tmp\_s_p$ **do**
  `// we also disable the dedicated runway that is unavailable to queue` q `by temporarily setting the corresponding` s_p `to a large value for the current run of Queue Pressure Algorithm`
  $p = \underset{p}{arg\,min}\{tmp\_s_p\}$;
  $\pi_q = \pi_q + (tmp\_s_p - \underline{t}_f)$;
  `Provisionally Assign Flight(`$tmp\_\mathcal{F}_q, p$`);`
  $f =$ index of first flight in $tmp\_\mathcal{F}_q$;
end
return π_q

Algorithm 5: `Assign Flight` $(\mathcal{F}_q, p)$

f = index of first flight in $\mathcal{F}_q$;
$s_p = max\{s_p, \underline{t}_f\} + \delta_p$; `// update the earliest channel time`
$\mathcal{F}_q = \mathcal{F}_q \backslash \{f\}$; `// remove flight from` $\mathcal{F}_q$
$\mathcal{F}_{new} = \mathcal{F}_{new} \cup \{f\}$; `// save channel assignment and holding time in a construct that will be returned at the end of the Queue Pressure Algorithm`

Algorithm 6: `Provisionally Assign Flight` $(\mathcal{F}_q, p)$

f = index of first flight in $\mathcal{F}_q$;
$tmp\_s_p = max\{tmp\_s_p, \underline{t}_f\} + \delta_p$; `// update the earliest channel time`
$\mathcal{F}_q = \mathcal{F}_q \backslash \{f\}$; `// remove flight from queue` q

4.4 First-Come-First-Served

First-Come-First-Served (FCFS) is a well-established and intuitive scheduling algorithm commonly used in various operational contexts where simplicity and fairness are key considerations. As the name implies, under this scheduling algorithm, whichever queue contains the aircraft with the earlier earliest aircraft time, will be chosen to be assigned in the current iteration. It is commonly used to benchmark scheduling algorithms against, and in the case of the runway scheduling problem, papers such as (Bosson and Sun, 2016; Ma, Sbihi, and Delahaye, 2019; Desai et al., 2022) compare the results of their experiments against the FCFS algorithm.

The FCFS algorithm will follow the same rules in Algorithm 2, except, on the third line from the bottom, we replace the `Queue Pressure Assign` $(\mathcal{F}_A, \mathcal{F}_D)$ algorithm with the `FCFS Assign` $(\mathcal{F}_A, \mathcal{F}_D)$ algorithm, as presented in Algorithm 7.

Algorithm 7: `FCFS Assign(`$\mathcal{F}_A$`,`$\mathcal{F}_D$`)`

if $\underline{t}_f < \underline{t}_{f'}$ **then**
  `// with` f `as first flight in arrival queue and` f' `as first flight in departure queue`
  `Assign Flight(`$\mathcal{F}_A, C$`);`
else if $\underline{t}_f < \underline{t}_{f'}$ **then**
  `// with` f `as first flight in departure queue and` f' `as first flight in arrival queue`
  `Assign Flight(`$\mathcal{F}_D, C$`);`
else
  `Assign Flight(`$\mathcal{F}_A, C$`);`

4.5 Results and Comparison of the QP and FCFS Algorithm

The dataset used for our experimental runs are published flight plans for flights arriving or departing from the ASEAN Plus region on 1 Oct 2023. For the results below, we reiterate that a flight is completed from the perspective of the FIR of interest as soon as the current simulation clock exceeds the TTO of the last leg. We model a look ahead period of two hours, such that all flights with at least one active or frozen leg during the next two hours will be added into consideration for the respective optimization algorithm. For flights with all legs frozen, we still include them as other flights, and still maintain the required separation from these frozen flights where possible. We set the time shift for the sliding window at five minutes, that is, we reschedule flights for the next two hours, at every five minute interval. As such, for the results in Tables 4.1 and 4.2, we are scheduling flights arriving or departing from the ASEAN Plus region on 1 Oct 2023, from 04:00:00 to 06:00:00 UTC, or 12:00:00 to 14:00:00 GMT +8. The regular airspace capacity is determined quantitatively by collecting information for the average number of flight traversals in each FIR, similar to the method employed in (Tan, 2021). The reduced airspace capacity is computed by synthetically reducing the original airspace capacity to 0.80 of its original levels, rounded up to an integer value. These capacity values are then used to compute the required separation time between aircraft using the method in Section 4.1.

The PC specifications used to run all the code are Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz with 16 GHz installed RAM. The code was written in R. The results for the AFR, for both the queue pressure and FCFS algorithms are presented in Tables 4.1 and 4.2, and contain the following columns:

- **Scenario and Algorithm**: The scenario, either the original or reduced capacity, and the algorithm used, either queue pressure or FCFS.

- **Makespan (A, D, C, Total)**: The sum of makespan for the arrival (A), departure (D) and common (C) channel, and the total makespan summed over the three channels. Expressed in hours.
- **Squared Deviation (A, D, C, Total)**: The sum of squared deviation for each flight assignment on the arrival (A), departure (D) and common (C) channel, and the total sum of squared deviation, summed over the three channels. Expressed in hours squared.
- **Run Time**: Computational run time taken to run the specified algorithm and scenario, for the entire ASEAN Plus region, computed using the R library microbenchmark. Expressed in seconds.

The makespan is for channel p is equal to $s_p - s_0$, where s_0 denotes the current simulation time, that is, the earliest time the first flight may be sequenced, and s_p here denotes the earliest available time to schedule the next flight on channel p, after the AFR algorithm has run to completion for the given airport. We additionally computed the sum of squared deviation, to investigate if the either algorithm reduces disproportionately larger deviations from the earliest aircraft time $\underline{f}_f$. The squared deviation for any flight is computed as $(\max\{0, s_p - \underline{t}_f\})^2$, that is, the squared value of the assigned holding time, as defined in Section 4.3. The values reported in Table 4.1 are summed over all airports, and for Table 4.2, summed across all arrivals and departures.

TABLE 4.1: Optimization Results for the AFR on 1 Oct 2023 04:00:00 UTC, Within ASEAN Plus Region Dataset, with the Makespan Metric

Scenario and Algorithm	Makespan (A)	Makespan (D)	Makespan (C)	Makespan (Total)	Run Time (s)
Regular Capacity, Queue Pressure	162.11	120.10	134.14	416.35	2.30
Regular Capacity, FCFS	162.70	120.14	133.56	416.40	1.31
Reduced Capacity, Queue Pressure	171.36	135.91	150.28	457.55	2.11
Reduced Capacity, FCFS	171.69	136.03	149.70	457.42	1.38

We first provide our analysis for the makespan, in Table 4.1. The slight differences in makespan values indicate minor changes in sequencing. Looking down the column for Makespan (Total), we observe that the choice between the queue pressure algorithm and the FCFS algorithm has negligible impact on the total makespan. The difference in

TABLE 4.2: Optimization Results for the AFR on 1 Oct 2023 04:00:00 UTC, Within ASEAN Plus Region Dataset, with the Sum of Squared Deviation Metric

Scenario and Algorithm	Squared Deviation (A)	Squared Deviation (D)	Squared Deviation (C)	Squared Deviation (Total)	Run Time (s)
Regular Capacity, Queue Pressure	15.49	10.10	9.38	34.97	2.08
Regular Capacity, FCFS	15.66	9.78	9.42	34.86	1.43
Reduced Capacity, Queue Pressure	32.91	39.65	27.64	100.20	2.29
Reduced Capacity, FCFS	33.14	39.59	27.65	100.38	1.35

values for Makespan (A), Makespan (D) and Makespan (C) columns also do not differ enough for the queue pressure and FCFS algorithm to conclude that one algorithm is better than the other in minimizing makespan. We also observe that the runtimes for the queue pressure algorithm is similar to, and only slightly longer than the FCFS algorithm.

Similarly for the squared deviation computational results, we observe that the difference in total squared deviation is not significant, leading us to conclude that either scheduling algorithm will be equally effective for the airport scheduler. Again, we notice small differences in values for squared deviation in all cases, and this validates that the queue pressure algorithm schedules flights differently from the FCFS algorithm, but no overall gain was to be made.

We conclude this chapter, by highlighting that the study is done from an ATFM perspective. Here, the separation times that were derived from a capacity envelope, are applied equally to all flights using the same channel or runway, regardless of aircraft weight categories. The additional scheduling rules based on queue pressure, used to select between the arrival and departure queues for assigning flights to a common runway, produced results comparable to those obtained using the FCFS algorithm. Given that the results of the FCFS algorithm are competitive with the queue pressure algorithm, we have opted to use the FCFS algorithm for the AFR component of the FF-ICE simulator. The primary motivation for this choice is the FCFS algorithm's simplicity in interpretation and implementation, with the added advantage of slightly faster runtimes.

4.6 Summary of Notation

We do not include notation for the capacity envelope in Section 4.1 as they are illustrative and not part of the AFR formulation. All other notations are summarized in Table 4.3.

TABLE 4.3: Notation for the AFR

$\mathcal{F}$	Set of all flights
$\mathcal{F}_A$	Set of arrival flights
$\mathcal{F}_D$	Set of departure flights
f	Flight $f \in \mathcal{F}$
p	Channel $p \in \{A, D, C\}$
q	Queue $q \in \{A, D\}$
$\mathcal{P}^q$	The set of channels at an airport that are available to queue q, where a queue represents an operation type, either arrivals (A) or departures (D)
$\mathcal{F}_q$	Flights that are in queue q, $q \in \{A, D\}$
$tmp\_\mathcal{F}_q$	Flights that are in queue q, $q \in \{A, D\}$, and are used only for provisional scheduling in the Queue Pressure Algorithm
$\mathcal{F}^q$	Flights that are in queue q, with no gaps
$\mathcal{G}_p$	The set of flights where the indicator variable $\mathbb{1}_{fp}$ is equal to 1, that is, when flight f has been assigned to channel p, $\mathcal{G}_p = \{f : \mathbb{1}_{fp} = 1\}$.
$\mathcal{P}^0$	The set of channels at an airport that are unavailable
δ_p	Minimum required separation time on channel p
t_f^p	Time flight f uses channel p
s_p	Earliest available channel time for channel p
tmp_s_p	Earliest available channel time for channel p, and is used only for provisional scheduling in the Queue Pressure Algorithm
s_0	Current simulation time, that is, the earliest channel time at the beginning of an iteration of AFR
$\underline{t}_f$	Earliest aircraft time for flight f
$\mathbb{1}_{fp}$	The indicator variable for if flight f uses channel p
π_q	Queue pressure for queue $q \in \{A, D\}$

Chapter 5

Waypoint Flow Regulator

The complementary algorithm to the Airport Flow Regulator (AFR), is the Waypoint Flow Regulator (WFR), both of which work together to achieve the goal of regulating aircraft flow in the context of Air Traffic Flow Management (ATFM). We recall from Section 3.4, the goal of the WFR is to issue speed regulation and hold commands to aircraft such that the minimum required separation between any pair of aircraft at all nodes are observed. We also penalize deviations from the preferred time to exit the current FIR, denoted backTTO, and holding times according to the node type.

This work begins at the construction of the network of air navigation routes for the ASEAN Plus region, based on flight schedules (departure and arrival times) obtained from OAG (OAG, 2024b). We build the network of nodes and routes by (i) processing information from the Aeronautical Information Publication (AIP) documents published by each country's civil aviation authority to identify the nodes (or waypoints) (see Figure 5.1), and arcs connecting waypoints along routes (see Figure 5.2) (ii) applying Dijkstra's algorithm to determine the shortest path between any airport pair, thereby defining the flight path routing for each flight.

We first formulate the optimization problem mathematically and provide a brief overview of the concepts central to mathematical optimization. Subsequently, we propose two algorithms to solve this problem, namely, Gradient Descent Ascent (GDA) and Simulated Annealing (SA). The formulations, results and additional considerations are included for each algorithm. Lastly, a summary of notation used in this chapter is provided in Section 5.6.

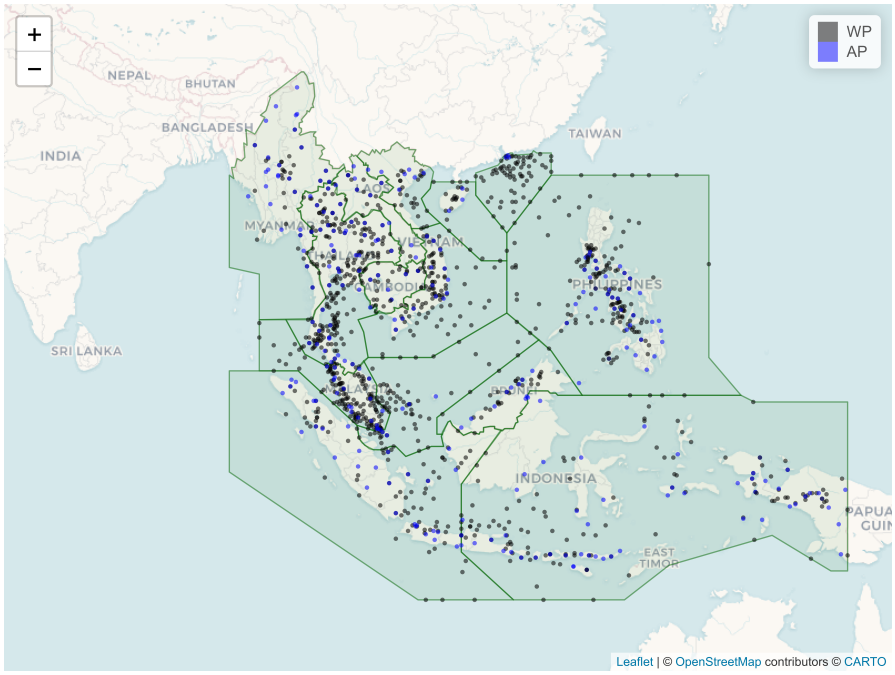

FIGURE 5.1: Flight Network Nodes Within the ASEAN Plus Region, with FIR Boundaries

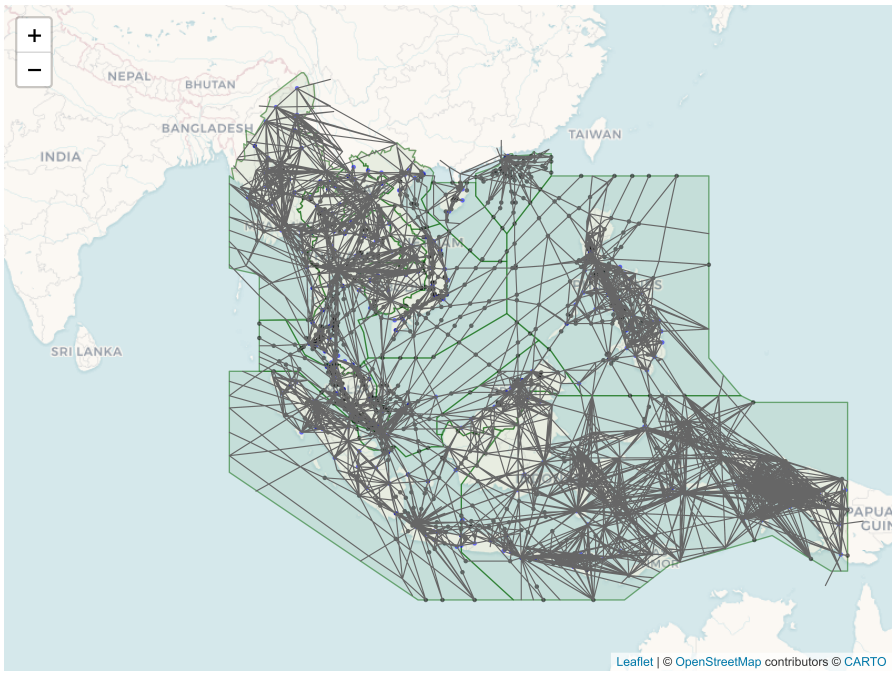

FIGURE 5.2: Flight Network Arcs Within the ASEAN Plus Region, with FIR Boundaries

5.1 Mathematical Formulation

For each input dataset, we are given the initial flight trajectory for all flights in a single Flight Information Region (FIR). A flight trajectory consists of the path a flight will take, from entry to exit of this FIR, and the entry time of a flight into this FIR. We base our dataset on flight schedules (arrival and departure times) downloaded from OAG (OAG, 2024b), and on flight plans derived using Dijkstra's shortest path algorithm applied to a representative network we created of air navigation routes for the ASEAN Plus region with connections to all origin-destination pairs in the flight schedule. Each flight plan specifies the sequence of nodes in our network traversed from its departure airport to its arrival airport. A node in the regional network may represent either an airport or a waypoint. For notational convenience, since the channels have been earlier determined by the AFR, a node in this chapter may refer to a waypoint, or a channel, one of arrival, departure, or mixed, at an airport. Each node in the regional network is designated by the FIR to which it belongs. Let $\mathcal{N}$ denote the set of nodes for the FIR of interest. For the FIR of interest, and each flight in the schedule, we restrict attention to the portion of the flight plan visiting nodes within that FIR. Let $\mathcal{F}$ denote the set of flight plans which intersect with $\mathcal{N}$. Let P_f denote the sequence of nodes for flight $f \in \mathcal{F}$. We assume that the sequence is contiguous in terms of flight legs, that is, a flight does not leave an FIR and then re-enter the same FIR on a later leg of its journey. Let $j = 1, 2, \cdots, |P_f|$ index the nodes of the FIR-restricted flight plan in sequence and let $n_j \in \mathcal{N}$, or simply $n \in P_f$ when no confusion should exist, denote a particular node in the sequence.

We assume that each flight f has an earliest time, t_f^0, at which it is estimated or known to have entered the airspace of the FIR. This could be the scheduled or actual departure time from an airport within the FIR, or it could be the estimated or known time the flight traversed the last node of a flight plan in an adjacent FIR, prior to entering the FIR of interest. Accordingly, we append a node $j = 0$ to the beginning of each flight plan P_f and associate with it the fixed time, t_f^0. No decision variables are associated with this node, so n_0 affects only the right-hand side of a constraint.

Similarly, we assume that each flight has a target time to land or otherwise exit the FIR. We associate this time with the last node in the flight plan and designate it as $\widetilde{t}_{|Pf|}$. In our simulation model, such times are initialized using flight schedules and updated dynamically by the airport flow regulator algorithm (for arrival times within the FIR) or, depending on the information sharing regime, passed in from the neighboring FIR if the flight transits out of the FIR of interest.

Each node $j = 1, 2, \cdots, |P_f|$, marks the *end* of a flight leg. The first leg, $j = 1$, can be thought of as the trip from the departure gate to the runway of the origin airport. We assume that the leg from n_{j-1} to n_j has a preferred duration $\widetilde{d}_{fj}$ particular to the

flight f. The preferred duration of the first leg is zero ($\widetilde{d}_{f1} = 0$). In our simulation, we compute an ideal time profile for each trajectory as a whole from origin to destination with reference to the rate of ascent, cruising altitude, cruising speed, and rate of descent specific to the equipment type. We compute an upper time profile (resp. lower time profile) by scaling the cruising speed down by 10% (resp. up by 5%). For any leg, we use these three profiles to derive the preferred duration $\widetilde{d}_{fj}$, as well as the maximum $\overline{d}_{fj}$, and minimum $\underline{d}_{fj}$ leg durations. Appendix B documents the calculations.

The trajectory of each flight $f \in \mathcal{F}$ is determined by the flight plan P_f, and for each node $j = 1, 2, \cdots, |P_f|$, the target time over (TTO) at the j^{th} node in path P_f, denoted by t_{fj}.

For each node $j = 1, 2, \cdots, |P_f|$, we introduce two decision variables: a speed regulation factor, β_{fj}, and an extraordinary hold delay, h_{fj}. The TTO values are determined with the following forward recursion:

$$t_{f0} = t_f^0; \tag{5.1}$$

$$t_{fj} = t_{fj-1} + h_{fj} + \beta_{fj}\underline{d}_{fj} + (1 - \beta_{fj})\overline{d}_{fj}; \tag{5.2}$$

where $h_{fj} \geq 0$ and $0 \leq \beta_{fj} \leq 1$ for all $j = 1, 2, \cdots, |P_f|$. Observe that increasing the speed regulation factor, β_{fj}, is associated with *reducing* the duration of the j^{th} leg.

Also, observe that the extraordinary hold delay, h_{fj}, is applied to the trajectory after the flight has traversed node $j - 1$. Thus, if we consider the first node in the FIR network, $j = 1$, then the first node in the flight path, $j = 0$, is a dummy node representing either a node in an adjacent FIR or an airport departure node. Consequently, h_{f1} can be interpreted as a request to the neighboring FIR to hold a flight from entering the FIR of interest or a request to the airport scheduler to impose an additional ground delay on the flight.

We impose no upper limits on the extraordinary hold delay variables, h_{fj}. Instead, we introduce an extraordinary hold cost parameter c_n associated with nodes $n \in \mathcal{N}$ to be applied to these delays. We anticipate the traffic manager would designate five classes of nodes with progressively increasing cost parameters: airport nodes (for which delays represent ground delays which cost less than airborne delays), neighboring FIR nodes (for which delays represent external airborne holding or requests for collaborative ground delays), nodes at entrances to designated holding areas within the FIR of interest, final approach nodes (for which delays represent vectoring), and all other waypoints (for which delays are strongly discouraged). In the experiments reported below, for example, we set the default values for holding costs for departure airports and final approach nodes at 1 and 10, respectively. Holding costs at all other nodes were set at 100.

The decision problem becomes a flow regulation problem when we impose flow constraints on the number of aircraft traversing specific nodes in the network. The flow capacity, and by extension, the minimum required separation time at each node, is determined in the same manner as in Section 4.1. We consider simple flow constraints, such as an upper bound on the number of traversals by aircraft of any type through a given node during the current planning horizon. We enforce this by imposing a minimum separation time between any pair of flights whose flight path takes them through the constrained node. We let Δ_n denote the minimum separation time for flights traversing the node n.

To express the flow constraint generically, let n denote a node with a flow constraint and let f and f' denote any pair of flights whose flight paths both include the node n. Let j index node n in P_f and let j' index node n in $P_{f'}$. Let $\mathcal{C}$ denote the set of all such tuples (f, f', j, j') of potential conflicts. Denote the corresponding TTOs at this node for these flights by t_{fj} and $t_{f'j'}$, respectively. We require $\left|t_{fj} - t_{f'j'}\right| \geq \Delta_n$. We enforce this generically by defining a penalty function $\delta\left(\left|t_{fj} - t_{f'j'}\right|\right)$ for any strictly decreasing function $\delta(\cdot)$ and imposing the constraint:

$$\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) \leq \delta\left(\Delta_n\right), \tag{5.3}$$

where $n = n_j = n_{j'}$.

An additional complication arises from the fact that these trajectories are updated in a rolling horizon fashion. At any point in time, some of the flights are en route. Consequently, not all TTOs are free to be modified. Our convention is that the TTO for the ending node of a leg becomes *frozen* as soon as the simulation clock exceeds the TTO of the beginning node of that leg. That is, once an aircraft has transited a node, the TTO of the next node in its flight plan is frozen in time. The following proposition follows, which is a necessary condition for the forward TTO recursion.

Proposition 5.1.1. *For any given trajectory, there does not exist a free leg before a frozen leg.*

Proof. From the definition of a frozen leg, the TTO at the end node of the leg must be immutable in the current and future time steps and current FIR of consideration. Assume we have a free leg before a frozen leg. From the definition of a free leg, we are allowed to modify its duration via changing its speed, or assigning a hold time. Say we increase the duration of this free leg. The forward TTO recursion would also increase the TTO at the end node of the frozen leg, hence a contradiction. The same argument can be made for decreasing the duration of the free leg. □

Let $\mathcal{R}$ denote the set of all flight legs:

$$\mathcal{R} = \bigcup_{f \in \mathcal{F}} \bigcup_{j=1}^{P_f} (f, j)\,, \tag{5.4}$$

and let $r = 1, 2, \cdots, |\mathcal{R}|$ index this set. Recall that we refer to a leg by its *end* node (the first leg begins with path index $j = 0$ and ends at $j = 1$). Let $\overline{\mathcal{R}}$ denote the set of frozen legs and $\mathcal{R}^O = \mathcal{R} \backslash \overline{\mathcal{R}}$, the set of free legs. For a frozen leg, $(f, j) \in \overline{\mathcal{R}}$, let $\bar{t}_{fj}$ denote the fixed TTO of the end node of the frozen leg. We require, for all $(f, j) \in \overline{\mathcal{R}}$,

$$t_{fj} = \bar{t}_{fj}. \tag{5.5}$$

A flight is *completed* from the perspective of the FIR of interest as soon as the last leg is frozen, and the current simulation clock exceeds the frozen TTO of the last leg. To ensure an active flight satisfies the minimum required separation from any completed flight, we will only archive a completed flight once all flight legs are completed, and the time now, is greater than the TTO of all flight legs belonging to this flight, plus its minimum required separation at each resource:

$$t_{now} > t_{fj} + \delta_{fj} \quad \forall j \in P_f. \tag{5.6}$$

Such flights are removed from consideration (i.e. deleted from $\mathcal{F}$). The remaining flights are considered to be *active* even if all legs are frozen because there is at least one leg whose frozen time may affect separation requirements.

Proposition 5.1.2. *Removing separation constraints where both legs involved are frozen, does not increase the state space of which we can search for an optimal solution. The state space remains equivalent.*

Proof. For flight legs that are frozen, they may not be modified in the current time step and current FIR of consideration, effectively behaving as parameters rather than variables. Alternatively, we declare that freezing a leg is equivalent to removing this variable from the state space and converting it into a fixed parameter. For a separation constraint with both flight legs frozen, there are no variables involved, hence does not change the state space as constraints normally do. As there is no change to the state space, such separation constraints are redundant to the current optimization model, and we may remove them without modifying the state space. □

Given that the state space is equivalent, we can reduce the number of separation constraints by restricting attention to a subset of the separation constraints, $\mathcal{C}^I$, where

at least one leg is free:

$$\mathcal{C}^I = \left\{ \left(f, f', j, j'\right) \in \mathcal{C} \mid (f,j) \in \mathcal{R}^O \text{ and/or } \left(f', j'\right) \in \mathcal{R}^O \right\}. \tag{5.7}$$

Similarly, frozen legs are not subject to the forward TTO recursion. Let j^O index the first leg in a given flight which is free. Then, the forward recursion

$$t_{fj} = t_{f(j-1)} + h_{fj} + \beta_{fj}\underline{d}_{fj} + (1-\beta_{fj})\,\overline{d}_{fj} \tag{5.8}$$

applies if and only if $j \geq j^O$. The recursion works due to Proposition 5.1.1, as we observe that if a free leg existed prior to a frozen leg, any change in the speed or holding time of the earlier free leg would be propagated to the frozen legs and modify their TTOs. This would contradict the fact that the TTOs of frozen legs should be frozen.

In summary, we define the Waypoint Flow Regulation Problem (WFRP) as follows. The problem is to determine values for (β_r, h_r) for all $r \in \mathcal{R}^O$ to

$$\underset{\beta,h}{\text{minimize}}\; g(\beta,h) = \sum_f \left(t_{f|P_f|} - \widetilde{t}_{f|P_f|}\right)^2 + \sum_{(f,j)\in\mathcal{R}} c_{n_j} h_{fj}, \tag{5.9}$$

subject to frozen leg constraints:

$$t_{fj} = \overline{t}_{fj}\;\forall\,(f,j) \in \overline{\mathcal{R}}, \tag{5.10}$$

initial node constraints:

$$t_{f0} = t_f^0, \forall\; f \in \mathcal{F}, \tag{5.11}$$

trajectory constraints:

$$t_{fj} = t_{fj-1} + h_{fj} + \beta_{fj}\underline{d}_{fj} + (1-\beta_{fj})\,\overline{d}_{fj}, \tag{5.12}$$

with decision variable bounds:

$$h_{fj} \geq 0, \text{ and } 0 \leq \beta_{fj} \leq 1, \tag{5.13}$$

for all $f \in \mathcal{F}$ and $j = j^O, j^O+1, \cdots, |P_f|$; and all separation requirements:

$$\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) \leq \delta\left(\Delta_{n_j}\right), \forall\left(f, f', j, j'\right) \in \mathcal{C}^I, \tag{5.14}$$

where $n_j = n_{j'}$.

Proposition 5.1.3. *A feasible solution to the WFRP is guaranteed to exist when the extraordinary hold times are unbounded from above.*

Proof. A sketch of the proof can be seen by setting every speed regulation variable to achieve the preferred duration, $\widetilde{d}_{fj}$, and every extraordinary hold time variable to zero except for the first free leg of each flight: $h_{fj} = 0, \forall j \geq j^O$. The free TTOs for each flight can be shifted arbitrarily far to the right by increasing h_{fj^o} for that flight. Flights can be considered in turn and shifted sufficiently to the right to satisfy separation constraints with all previously considered flights. □

Next, we define the terms local minima and global minimum, as these definitions are required in Proposition 5.1.7. We denote the feasible region of the WFRP $\mathcal{W}$, and the region near the current solution as a ball centered at $(\overline{\beta}, \overline{h})$ as:

$$B(\overline{\beta}, \overline{h}, \epsilon) = \{\beta | \|\beta - \overline{\beta}\| \leq \epsilon, h | \|h - \overline{h}\| \leq \epsilon\}. \tag{5.15}$$

Definition 5.1.4. *(β, h) is a local minimum of the WFRP if there exists $\epsilon > 0$ such that $g(\beta, h) \leq g(\beta', h')\ \forall(\beta', h') \in B(\beta, h, \epsilon) \cap \mathcal{W}$.*

Definition 5.1.5. *(β, h) is a global minimum of the WFRP if $g(\beta, h) \leq g(\beta', h')\ \forall(\beta', h') \in \mathcal{W}$.*

Definition 5.1.6. *A function f is convex if:*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \tag{5.16}$$

for all $x, y \in domf$ and all $t \in [0, 1]$.

For sets, this may be restated as: A set C is convex if the line segment between any two points in C lies in C:

$$tx_1 + (1-t)x_2 \in C \tag{5.17}$$

$\forall x_1, x_2 \in C, \forall t \in [0, 1]$

Proposition 5.1.7. *The WFRP is a non-convex problem.*

Proof. The separation constraint involves an absolute function on the TTOs, $\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) \leq \delta\left(\Delta_n\right)$. This causes the feasible region to be disjoint. We can draw a line between two points, each from a disjoint segment and it will lie outside the feasible set, hence using Definition 5.1.6, we have shown the WFRP is a non-convex problem. □

Given that the WFRP is a non-convex problem, local minima may not equal to the global minimum. We illustrate this concept with an example, by simplifying the WFRP to two flight legs, t_1 and t_2, with preferred times $\widetilde{t}_1 = 0$, $\widetilde{t}_2 = 3$, and separation constraint $e^{-|t_1 - t_2|} \leq e^{-5}$, using the $\delta(\cdot)$ function defined in Section 5.3.1. Note that in the

WFRP, the values representing time are positive, but we allow them to be negative for this example. Also, we simplify the notation by using $t = (t_1, t_2)$.

In Figure 5.3, the concentric circles indicate the level sets of $f(t) = (t_1-0)^2+(t_2-3)^2$, the function which mimics deviation from preferred time, of which we are trying to minimize. The area shaded in light blue is the feasible set, of which the two aircraft obey separation requirements. The second-largest circle $f(t) = 32$, is a local minimum at $(4, -1)$, as perturbing t_1 or t_2 by a small value ϵ will lead to a larger value of $f(t)$, as represented by the largest circle $f(t) = 35$. However, $f(t) = 32$ is not a global minimum as there exists another local minimum where the circle $f(t) = 2$ touches the feasible set at $(-1, 4)$ and $2 < 32$. Applying this analogy to the WFRP, we state that *local minima exist when aircraft obey the separation constraints, with TTO as close to the preferred time as possible, but are in a non-optimal sequence*. A global minimum exists when aircraft obey the separation constraints, with TTO as close to the preferred time as possible, and are optimally sequenced.

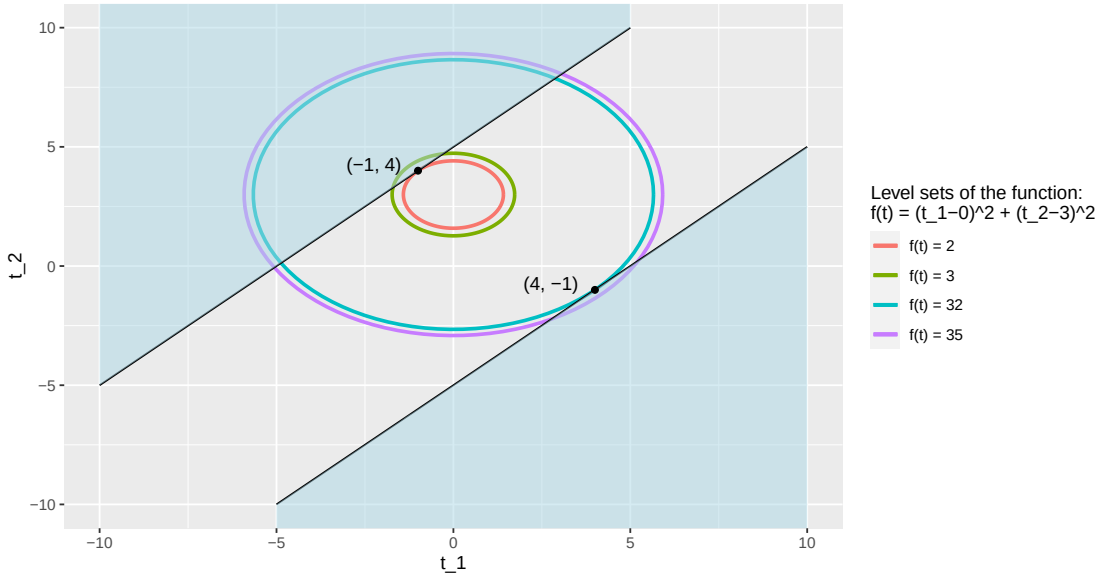

FIGURE 5.3: Geometrical View of Local and Global Minima

The existence of local minima for the WFRP is an important consideration in selecting optimization heuristics, given that a local minimum could be suboptimal, and potentially possess an arbitrarily large optimality gap from the global minimum.

5.2 Description of Mathematical Optimization

The modern form of mathematical optimization, as it is known today, began around the time of Newton, Gauss, Fermat, and Lagrange, in the fifteenth and sixteenth centuries. Some of their contributions to the field of optimization are the method of Lagrange multipliers, Newton's method, Fermat's theorem, and Gauss-Newton algorithm. Mathematical optimization is a method of obtaining the best possible outcome, based on the available choices. Today, it plays a huge role in many industries, not limited to transportation, energy, finance, scheduling, routing, and telecommunications (Festa, 2014). As seen in the discussion of ATFM in Chapter 2, airspace problems are no exception to this, where mathematical optimization has been extensively used to improve the efficiency of aviation operations, and have been proven successful in increasing the airspace and runway capacity.

More formally, mathematical optimization is the process of breaking down a problem into either a discrete or continuous set space of inputs. This set space of inputs is commonly referred to as decision variables. From these inputs, subject to further constraints, select the input set which returns the optimal values for zero or more objective functions. For a maximization problem (resp. minimization), given a set of variables $x \in \mathbb{R}^n$ and an objective function $f(x)$, we aim to find x_0 for which $f(x_0) \geq f(x)$ (resp. $f(x_0) \leq f(x)$) for all $x \in S$ where S is the subset of $\mathbb{R}^n$ which fulfills the specified set of constraints.

Problems with no objective function are also known as feasibility problems, where the goal is to find any values for the variables such that all constraints are satisfied. In other situations, there may be more than one objective function. For these problems, we may search for a Pareto optimal set of possible solutions or, more commonly, use a multi-objective optimization tool such as the popular Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb et al., 2002).

The characterization of inputs, constraints and objective functions are critical to determining the final outcome. For example, the least squares regression problem is well-defined with objective to minimize the sum of squared residuals, where otherwise minimizing the residuals itself would give an incorrect result since the case of which large positive and negative residuals balance each other out would be considered optimal. This is often the most important phase where the decision has substantial impact on the interpretability and application of the problem and its corresponding solution.

There are two main categories of algorithms designed for mathematical optimization: exact methods and heuristic methods. As the name suggests, exact methods return the optimal solution, which can be proven to be globally optimal. Common approaches include Branch and Bound, Integer Programming, Linear Programming and Dynamic Programming. The strength of exact methods lies in its ability to obtain the

global optimal. However, a number of problems, for example, the Traveling Salesman Problem, Flow Shop Scheduling Problem, and minimum k-cut problem, are NP-complete, for which no polynomial-time algorithm is currently known. This implies obtaining the global optimum within a reasonable computational time frame when the problem size is large is not feasible. To answer these important problems, research has been directed towards heuristic methods, which return a solution, with no guarantee of achieving the global optimal and are often, instead, compared against other solutions (Festa, 2014). The strength of heuristic methods lies in their computational efficiency and ability to achieve solutions within an acceptable optimality gap. They have been used extensively in place of exact algorithms for problems that are too large for exact algorithms to provide a solution for real time decision-making. Some examples of heuristic methods are Simulated Annealing, Genetic Algorithms, Gradient Descent and Neural Networks.

The WFR is a nontrivial optimization problem because it involves sequencing flights at nodes, of which is combinatorial in nature. A natural choice of optimization model for smaller instances would be to consider using an integer program, as done in the papers (Balakrishnan and Chandran, 2014; Tan, 2021). However, in the broader context of our research, we are constrained by computational time of the algorithm as we require solving a dozen or so FIRs at each time step of our simulation, with thousands of flights over a 24-hour period. For this reason, we pursue a heuristic approach to solving the WFRP by means of GDA and SA, both of which are presented in the next few sections. We will then discuss which heuristic is most suited for the WFRP and provide our choice of heuristic for the simulator to conclude the chapter.

5.3 Gradient Descent Ascent

The Gradient Descent (GD) family of algorithms is one of the most widely used algorithms for unconstrained mathematical optimization. It makes use of the first order derivative of some differentiable multivariate function, guiding the variables against the direction of the gradient, leading to a local minimization of the objective function. GD is used extensively in the field of Deep Learning, where every state-of-the-art Deep Learning library contains various implementations and algorithms of GD (Ruder, 2016). Some of the most popular extensions include Adam, which combines AdaGrad and RMSProp, designed to work well with sparse gradients and non-stationary settings (Kingma and Ba, 2014), and Stochastic Gradient Descent, designed to work well in the context of high dimensional optimization. Gradient Descent Ascent (GDA) is another flavor of GD, which sees applications in different fields, being more commonly applied in Generative Adversarial Networks and Reinforcement Learning. One key

strength of the GDA lies in its capacity to handle constrained optimization problems through Lagrangian duality. Therefore, in this paper, we leverage GDA's advantages to solve the WFR, which is formulated as a constrained optimization problem.

For a minimization problem, our augmented objective function is $\min_x \max_\lambda \mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x)$, subject to $\lambda \geq 0$. Here, $f(x)$ denotes the objective function, and $g(x)$ denotes set of constraint functions. We can think of this method as penalizing the objective function when $g(x) \geq 0$ as $\lambda^T g(x)$ will be greater than zero, bringing us further away from our goal of minimizing the function with respect to x. Alternatively, a geometrical view is noticing how the term $\lambda^T g(x)$ imposes supporting hyperplanes $g_i(x) \leq 0$ onto the feasible set, which shrinks the set of values which x will take on, and with appropriate step sizes, we will eventually have $x \in S$.

The GDA heuristic is described in Algorithm 8.

Algorithm 8: Gradient Descent Ascent

$x^{(1)} \leftarrow x_0$;
$\lambda^{(1)} \leftarrow \lambda_0$;
for $i \leftarrow 1$ **to** N **do**
  compute $\frac{\partial \mathcal{L}}{\partial x}$ and $\frac{\partial \mathcal{L}}{\partial \lambda}$;
  update $x^{(n+1)} = x^{(n)} - \eta \frac{\partial \mathcal{L}}{\partial x}$ and $\lambda^{(n+1)} = \lambda^{(n)} + \theta \frac{\partial \mathcal{L}}{\partial \lambda}$.
end

For Algorithm 8, let x_0 and λ_0 represent arbitrary initial values of x and λ respectively, and N denote the number of iterations. Let $x^{(n)}$ for any variable x, represent the value of x at the n^{th} iteration of the GDA heuristic. Also, let η and θ represent the step size vector for the primal vector x and dual vector λ respectively.

5.3.1 Choice of Penalty Function

Our choice of the penalty function $\delta(\cdot)$ for use in the separation constraints is driven by the decision to use the GDA algorithm. In general, we want $\mathcal{L}(x, \lambda)$ to be strongly convex with respect to the primal vector x. A strongly convex function possesses the property where if the gradient at a point is close to zero, the point is close to its locally optimal value. Boyd and Vandenberghe (Boyd and Vandenberghe, 2004) provides a proof of convergence for descent methods on strongly convex functions. We selected the Laplace function, $\delta(x) = e^{-|x|}$, plotted in Figure 5.4 for this reason, as it is strongly convex on both the positive and negative x domain. We also considered the reciprocal of squared deviation $\delta(x) = \frac{1}{x^2}$ but anticipated that its lack of continuity at zero would lead to extreme behavior.

A plot of the Laplace function is given in Figure 5.4, with a sample required separation of two units. In the figure, when the x values are close to zero, within the red zone, the constraint is violated. Otherwise, when the x values are further from zero, within the blue zone, the constraint is satisfied.

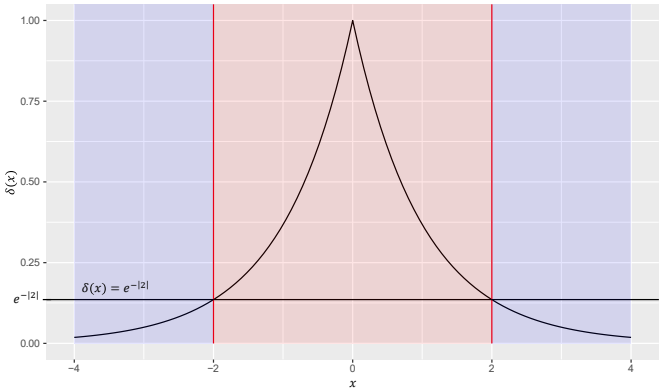

FIGURE 5.4: Graph of $\delta(x) = e^{-|x|}$, With a Sample Required Separation of Two Units

5.3.2 Lagrangian Relaxation

We associate the dual variable λ_{f,f',n_j}, where $n_j = n_{j'}$, with each separation constraint $(f, f', j, j') \in \mathcal{C}^I$, and form the Lagrangian function $\mathcal{L}(\beta, h, \lambda)$ as follows:

$$\begin{aligned}\mathcal{L}(\beta, h, \lambda) = &\sum_{f} \left(t_{f|P_f|} - \widetilde{t}_{f|P_f|}\right)^2 + \sum_{(f,j)\in\mathcal{R}} c_{n_j} h_{fj} \\ &+ \sum_{(f,f',j,j')\in\mathcal{C}'} \lambda_{f,f',n_j} \left(\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) - \delta\left(\Delta_{n_j}\right)\right).\end{aligned} \tag{5.18}$$

Except for the decision variable bounds, all other constraints are expressed as equalities and hence can be eliminated through substitution. The dualized WFRP is thus to find primal vectors (β, h) and dual vector λ to solve:

$$\max_{\lambda} \min_{\beta,h} \mathcal{L}(\beta, h, \lambda) \tag{5.19}$$

subject to the decision variable bounds:

$$h_{fj} \geq 0, \text{ and } 0 \leq \beta_{fj} \leq 1, \tag{5.20}$$

for all $f \in \mathcal{F}$ and $j = j^O, j^O + 1, \cdots, |P_f|$.

5.3.3 Gradient Descent Ascent Applied to the Waypoint Flow Regulator Problem

We compute the partial derivatives of the Lagrangian as:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \beta_{fj}} = (\underline{d}_{fj} - \overline{d}_{fj})\Bigg(& 2\left(t_{f|P_f|} - \widetilde{t}_{f|P_f|}\right) \\ & - \sum_{(f,f',j,j')\in\mathcal{C}^I} \lambda_{f,f',n_j}\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) sgn\left(t_{fj} - t_{f'j'}\right) \\ & + \sum_{(f',f,j,j')\in\mathcal{C}^I} \lambda_{f',f,n_j}\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) sgn\left(t_{f'j'} - t_{fj}\right)\Bigg) \ \forall (f,j) \in \mathcal{R};\end{aligned} \tag{5.21}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial h_{fj}} = \Bigg(& 2\left(t_{f|P_f|} - \widetilde{t}_{f|P_f|}\right) \\ & - \sum_{(f,f',j,j')\in\mathcal{C}^I} \lambda_{f,f',n_j}\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) sgn\left(t_{fj} - t_{f'j'}\right) \\ & + \sum_{(f',f,j,j')\in\mathcal{C}^I} \lambda_{f',f,n_j}\delta\left(\left|t_{fj} - t_{f'j'}\right|\right) sgn\left(t_{f'j'} - t_{fj}\right)\Bigg) + c_{n_j} \ \forall (f,j) \in \mathcal{R};\end{aligned} \tag{5.22}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{f,f',n_j}} = \delta\left(\left|t_{fj} - t_{f'j'}\right|\right) - \delta\left(\Delta_{n_j}\right) \ \forall \left(f,f',j,j'\right) \in \mathcal{C}^I. \tag{5.23}$$

Our aim is to minimize the Lagrangian with respect to β and h, and maximize the Lagrangian with respect to λ, with the following bounds:

$$0 \leq \beta_{fj} \leq 1 \ \forall (f,j) \in \mathcal{R}; \tag{5.24}$$

$$0 \leq h_{fj} \ \forall (f,j) \in \mathcal{R}; \tag{5.25}$$

$$0 \leq \lambda_{f,f',n_j} \ \forall \left(f,f',j,j'\right) \in \mathcal{C}^I. \tag{5.26}$$

We do this by updating the variables β, h, λ, with step sizes η, ι, and μ as follows:

$$\beta^{(n+1)} = \min\left\{1, \max\left\{0, \beta^{(n)} - \eta\frac{\partial \mathcal{L}\left(\beta^{(n)}, h^{(n)}, \lambda^{(n)}\right)}{\partial \beta}\right\}\right\}; \tag{5.27}$$

$$h^{(n+1)} = \max\left\{0, h^{(n)} - \mu\frac{\partial \mathcal{L}\left(\beta^{(n)}, h^{(n)}, \lambda^{(n)}\right)}{\partial h}\right\}; \tag{5.28}$$

$$\lambda^{(n+1)} = \max\left\{0, \lambda^{(n)} + \iota\frac{\partial \mathcal{L}\left(\beta^{(n)}, h^{(n)}, \lambda^{(n)}\right)}{\partial \lambda}\right\}; \tag{5.29}$$

where $x^{(n)}$ for any variable x represents the value of x at the n^{th} iteration of the GDA heuristic.

Here, note that the penalty function $\delta(x) = e^{-|x|}$ is non-differentiable at $x = 0$. Many techniques exist to overcome such a challenge, particularly within the machine learning community. One example is an adaptation of the ReLU gradient (Bai, 2022), of which gradient is piecewise determined as:

$$f'(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x \geq 1. \end{cases} \tag{5.30}$$

This can be adapted to our GDA algorithm by adding a case for $x = 0$. In this case, we can assign an arbitrary value to the derivative which will lead to a change in the TTO of the overlapping flights, and such a change will separate the two flights that are in conflict.

Another option is a clip function, commonly used to prevent excessive jumps in variable values (DeepSeek-AI et al., 2025), particularly when computing the gradient. The clip function is defined as:

$$\text{clip}(x, \overline{x}, \underline{x}) = \min\{\max\{x, \underline{x}\}, \overline{x}\}. \tag{5.31}$$

Here, the clip function bounds a value between a particular minimum and maximum value. Applied to our GDA, this would prevent the TTO from changing by an excessive amount at each time step. We have opted for the clip function to deal with the non-differentiability of the penalty function. Given that the primal and dual variables are already bounded, where $0 \leq \beta \leq 1$, $0 \leq h$, and $0 \leq \lambda$, we only need to clip off any large increments to both h and λ. The upper limits for the changes in h and λ at each iteration were set to 500 and 10,000, respectively. The limit for h was chosen to reflect a reasonable maximum adjustment in the holding time per iteration. In contrast, a substantially larger limit was assigned to λ because it is not a primal variable and does not directly influence the objective function. This allows for significant variation in λ without compromising computational stability.

Choosing suitable step sizes is both an art and a science: too large and our GDA heuristic will oscillate, too small and convergence will be slow. In addition to choosing a step size for each variable, we must implement a scheme to shrink the step sizes. This is because oscillation occurs with constant step size for non-strongly concave problems, particularly when there is interaction between the variables over which we are maximizing and minimizing (T. Liang and Stokes, 2018).

To mitigate the impact of these oscillations, we geometrically shrink the step size every 50 iterations, with the total number of iterations for the GDA heuristic set at 500.

For both β's and h's step sizes, we multiply them by 0.6 once every 50 iterations have been completed. Note here that we do not multiply the step size for λ with a shrinking factor, because if we reduce the step size ι, when two flights get too close towards the end of the algorithm, then λ will increase too slowly to have an impact on the variables that determine the flight schedule β and h. This would likely lead to conflicts as flights tend toward their preferred exit time, or reduce holding time, unless the penalty (λ) imposed for conflicts is sufficiently large.

The initial values of step sizes are set according to the general principle to have the change, at each iteration, for β and h to be of magnitude 0.1 and 100 respectively, and for λ to be of magnitude 1. To that end, with some trial and error, we set $\eta = 1/10000$, $\mu = 2$ and $\iota_{fj} = 1000/\left(\delta\left(\Delta_{n_j}\right)\right)^2$. We sought to make ι_{fj} to be unique to each flight leg to normalize the step size with respect to Δ_{n_j}, as our experimental results demonstrated that not scaling ι according to required separation led to unevenness in number of iterations to converge. This occurred because the variable λ_{f,f',n_j} corresponding to a constraint with large separation time will increase significantly slower than λ_{f,f',n_j} corresponding to a small separation time, particularly when exponentiation is involved. These parameters have been found to produce the least number of conflicts when GDA was run across all FIR datasets generated by our simulator.

5.3.4 Practical Issues of a Gradient Descent Ascent Approach

During our implementation of GDA, we faced two notable issues. These are the selection of the step size, and the possibility of convergence to a local minimum, which led to unresolved conflicts at the termination of the GDA algorithm. We briefly mention some examples to illustrate these issues in the GD context, and note that these issues occur in the same manner in GDA.

In the examples below, we are trying to minimize some function $f(x)$ with respect to x. The rightmost red dot represents the starting value of x, and the blue dot(s) represent the final value of x and the corresponding value of $f(x)$.

In Figures 5.5, 5.6, and 5.7, we are minimizing the function $f(x) = x^2$. Figure 5.5 represents how the function $f(x)$ converges to the minimum point when the step size is chosen appropriately, Figure 5.6 represents how function $f(x)$ converges to the minimum point when the step size is too large, and Figure 5.7 represents how function $f(x)$ converges to the minimum point when the step size is too small. In the earlier Section 5.3.3, we have discussed our strategy for selecting appropriate step sizes.

We now illustrate the problem of getting trapped in a local minimum. In Figure 5.8, we are minimizing the function $f(x) = x \sin x, x \in [0, 12]$. Figure 5.8 represents how the value of the function $f(x)$ is trapped at a local minimum, where a better minimum exists in the right valley around the region $x = 11$. In the context of the GDA, the final

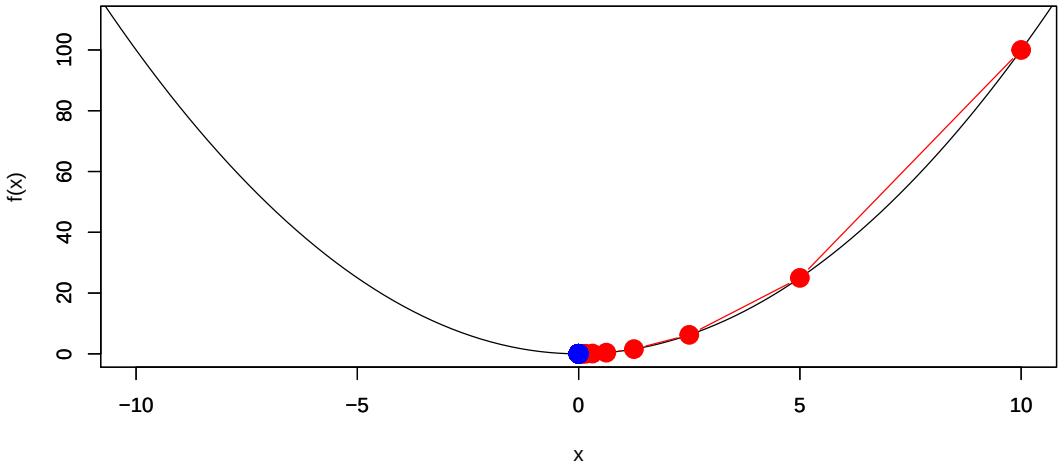

FIGURE 5.5: Graph of $f(x) = x^2$, With a Suitable Step Size

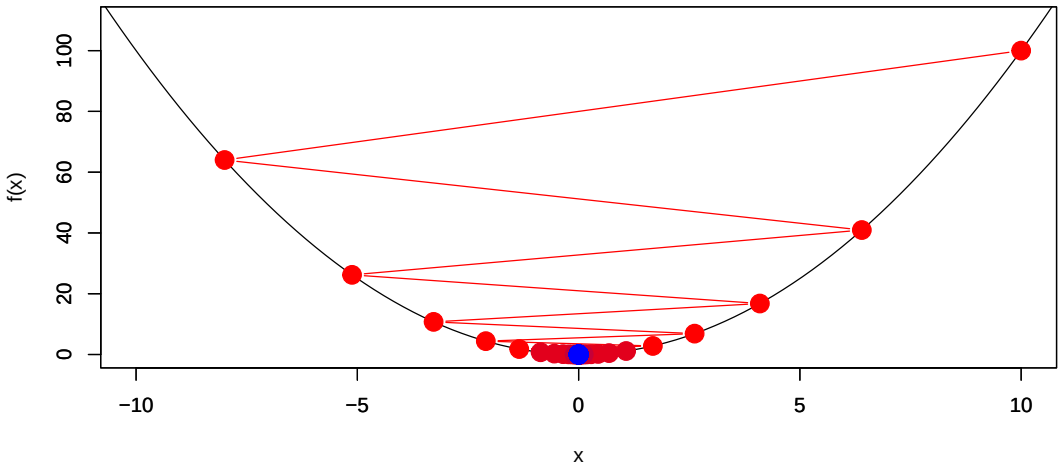

FIGURE 5.6: Graph of $f(x) = x^2$, With a Large Step Size, optimized with GD

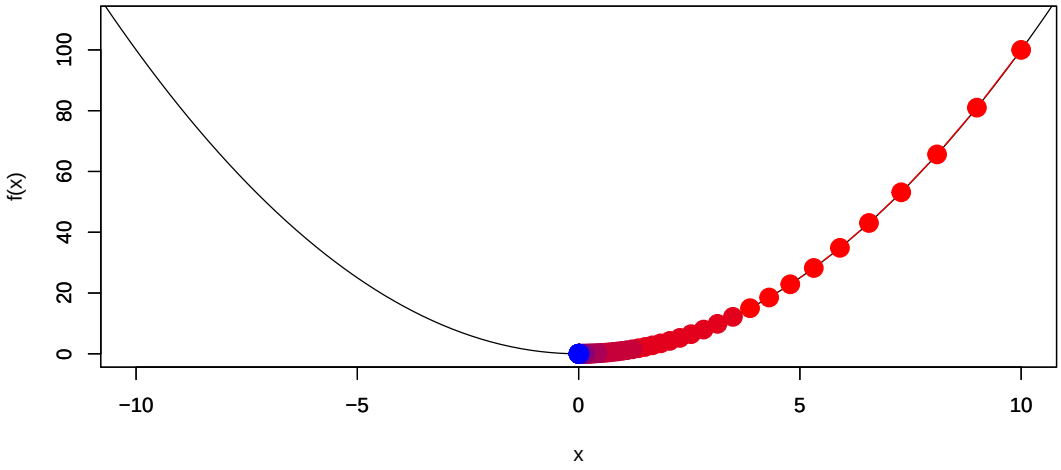

FIGURE 5.7: Graph of $f(x) = x^2$, With a Small Step Size, optimized with GD

state is determined by the initial state and step size. While convergence to a global minimum cannot be guaranteed, our results have been promising. Next, in Section 5.4, we attempt to circumvent this issue through the use of the SA algorithm. Future works may consider an extension to the GDA approach via the use of a more adaptive gradient descent algorithm such as Adam, Momentum, or RMSProp, which have seen success in various applications including backpropagation in neural networks. Such modifications to the traditional gradient descent model have been shown to avoid getting trapped in local optima. Ruder (Ruder, 2016) provides a comprehensive overview of these methods.

In addition, we did consider that holding times may not be equitably distributed across flights, and changing the holding cost to a squared value might resolve this issue. However, we did not record a noticeable difference under both the regular and reduced airspace capacity. The results were virtually identical, and only differing by less than one percent. We attribute this to the dominance of the dual variables λ, which exert significantly greater influence than the other terms in the objective function. This highlights a key challenge in multi-objective optimization, in the difficulty of balancing objectives when large coefficients are used to enforce constraints in the objective.

How the GDA heuristic handles sequencing of flights was also considered. During our test runs, we observed some resequencing of flights, which occurred because of the large initial step sizes allowing flights to swap places in the initial iterations. However, we acknowledge this swapping of flight sequences is a convenient secondary effect of the GDA approach, rather than an explicit feature of the GDA heuristic, hence the

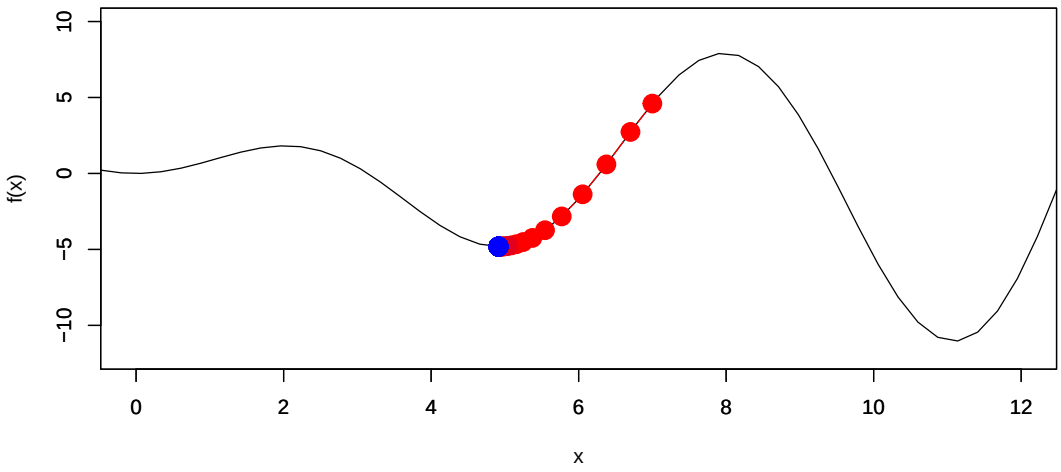

FIGURE 5.8: Graph of $f(x) = x \sin x$, Optimized with GD

solution quality remains primarily dependent on the initial values of β and h.

Another issue we noticed, when utilizing the GDA heuristic, was slow convergence when GDA attempts to separate multiple flights that are packed closely together. We define closely packed flights as scenarios in which multiple flights are separated by intervals that are close to, or slightly less than, the minimum required separation, as illustrated by scenario B in Figure 5.9. With reference to Figure 5.9, when scenario B occurs, the topmost and bottommost flights will move up and down respectively, while the flights in the middle will oscillate while trying to obtain the required separation. This process becomes slow when many flights are tightly packed, as the constraints imposed by preceding and succeeding flights tend to partially offset each other, reducing the effectiveness of each individual adjustment. Furthermore, we have observed instances where the dual variable λ oscillates, as shown in Figure 5.10. This figure portrays the oscillation of the dual variable λ obtained from one of our GDA runs.

Additionally, at the termination of the GDA heuristic, we discovered instances where some variables did not yet converge, leaving free flight legs in conflict with each other, although these could possibly have been resolved. Flights with unresolved conflicts most commonly appear when there are free flight legs between frozen flight legs. With reference to scenario B in Figure 5.9, if the bottom-most and topmost flights are frozen, and the flights in between are free, the flights in between will erratically oscillate between the two frozen flights and the GD heuristic would not be able to resolve the conflicts. This shows that in practice, Proposition 5.1.3 is not realized due to frozen flight legs.

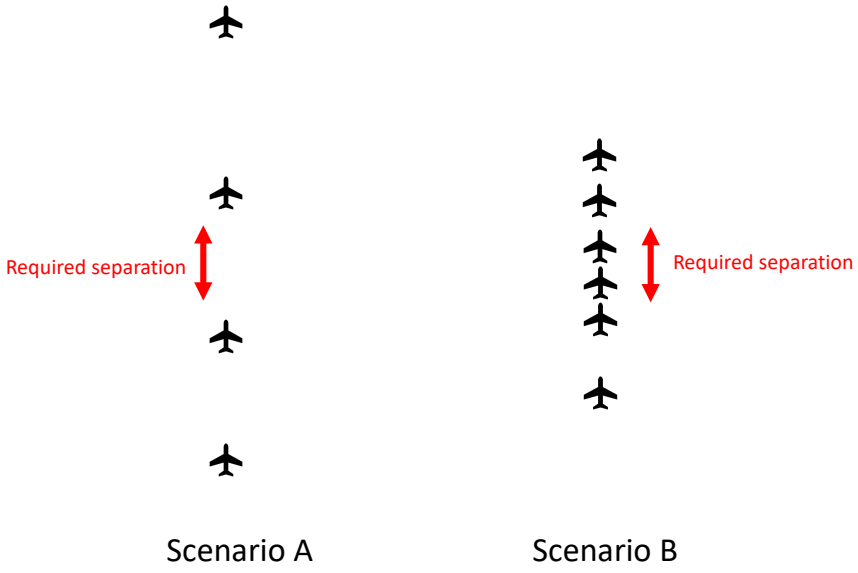

FIGURE 5.9: Scenarios of Aircraft Being Loosely Packed and Tightly Packed

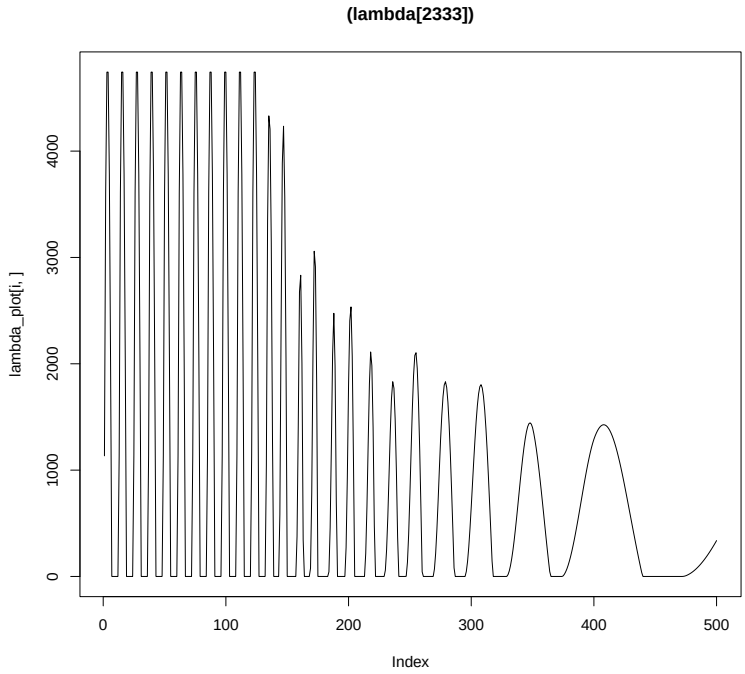

FIGURE 5.10: Oscillation of Dual Variable λ

We address the convergence issue, together with the sequencing issue, by adding a subroutine to generate a good sequence of flights. We define a good sequence of flight as one where a conflict free solution would be generated by the GDA heuristic and is equitable to all aircraft, such that we do not apply an extraordinary delay to any particular aircraft. To that end, we initially sequence flights in a manner that assigns an appropriate number of free flights between any two consecutive frozen flights, prioritizing free flights based on their earliest arrival time to the node of interest. This is a twist on the classical First-Come-First-Served (FCFS) algorithm, distinguished by the presence of additional constraints imposed by the frozen flight legs. The algorithm first splits the free and frozen flight times into separate vectors, and sorts them. The free flight times are the earliest time that the given flight can arrive at a given node, obtained by setting $\beta_{fj} = 1$ and $h_{fj} = 0$. It then precomputes how many free flights can be inserted with sufficient separation into an interval between any consecutive frozen flights. These may be easily computed as $\frac{tFrozen[i]-tFrozen[i-1]}{\Delta_n} - 1$, where $tFrozen$ is a vector of frozen flight times at a given node and Δ_n is the required separation time at the given node. It then sequentially inserts the free flights into the earliest possible interval. Finally, it records the interval the free flight was inserted into, and adds a bound to the hold time for this free flight, such that the GDA will be initialized with this free flight in its appropriate interval. The algorithm to generate the Minimum Holding Time (MHT) vector for free flights is given in Algorithm 9.

Here, $rep(N, 0)$ denotes a vector of length N, filled with value 0, and represents the MHT for all flight legs. t denotes the vector of leg times, $tFrozen$ and $tFree$ denote the vector of frozen and free leg times respectively, and $buckets$ denote the vector of number of flights that may use a given interval, derived from $\frac{tFrozen[i]-tFrozen[i-1]}{\Delta_n} - 1$. We also have $tFrozen[0] =$ *simulation time now* and $tFrozen[len(tFrozen) + 1] = INF$. The second assignment guarantees a feasible solution to the pre-sequencing algorithm, allowing all residual flights to be assigned to the final bucket. The argument for feasibility follows that of Proposition 5.1.3.

Our results demonstrate that the pre-sequencing algorithm successfully generated conflict-free trajectories, albeit with a slightly greater deviation from the preferred times as compared to the case without pre-sequencing. However, the resolution of conflicts takes higher precedence than reduction of deviation time both in our FF-ICE simulator and in practice. Accordingly, we proceed with the inclusion of the pre-sequencing subroutine in the GDA heuristic. Moreover, computation times did not increase significantly with the pre-sequencing algorithm. These results are presented in Section 5.5.

Algorithm 9: Pre-sequencing

$MHT \leftarrow rep(N, 0)$

for *node* n *in nodes* **do**

$tFrozen = t[node == n \ \& \ frozen]$;

$tFree = t[node == n \ \& \ !frozen]$;

idx = *vector matching the indices of tFree to MHT;*

$buckets$ = *vector representing no. of flights that can fit in the interval of tFrozen[i - 1] and tFrozen[i]. The length of this vector is len(tFrozen) + 2;*

sort($tFrozen$); sort($tFree$);

$ptr \leftarrow 1$

for *i in* $1 : len(tFree)$ **do**

```
// prevent assignment to an earlier bucket
```

while *tFree[i] > tFrozen[ptr]* **do**

$ptr = ptr + 1$

end

```
// find earliest bucket with an available slot
```

while *buckets[ptr] == 0 OR tFrozen[ptr] - tFree[i]* $< \Delta_n$ **do**

$ptr = ptr + 1$

end

$MHT[idx[i]] = tFrozen[ptr] - tFree[i]$;

$buckets[pts] = buckets[ptr] - 1$; `// reduce the capacity of this bucket by 1`

$break$

end

end

5.4 Simulated Annealing

In attempts to investigate properties as equations of state for substances consisting of interacting individual molecules by Metropolis in 1953 (Metropolis et al., 1953), the foundations for Simulated Annealing (SA) were established. A few decades later, in 1983, Kirkpatrick (Kirkpatrick, Gelatt, and Vecchi, 1983) coined the term Simulated Annealing, and investigated its use as a heuristic approach to solve the intensely studied, NP-complete, traveling salesman problem. Since then, SA has earned a spot in many textbooks such as (Henderson, Jacobson, and Johnson, 2006; Gendreau and Potvin, 2010; Aarts and Korst, 1997), and is today a very popular heuristic in optimization, and was utilized by multiple papers as seen in the literature review of Chapter 2.

The SA heuristic derives its name from the process of annealing: a process where a solid is heated to a sufficiently high temperature that permits many atomic rearrangements, then cooled in a controlled manner until the material solidifies into a low energy, stable state. With reference to Figure 5.11, when temperature is high, the material is in a liquid state, as illustrated in the left boxes. For a hardening process, the material reaches a solid state with non-minimal energy, that is, a metastable state, shown in the top right box. Under such conditions, the structure of the atoms yields no symmetry. In contrast, when conducting a slow annealing process, the material reaches also a solid state, but in which atoms are organized with symmetry in a crystal state, as displayed in the bottom right box (Delahaye, Chaimatanan, and Mongeau, 2018). This concept is applied in the Simulated Annealing (SA) heuristic, where iterative heating and cooling processes are implemented through probabilistic acceptance and rejection of solutions, leading to convergence toward a stable solution that is only weakly dependent on the initial starting point.

In a combinatorial optimization problem, we seek to find values for each variable, with all variables denoting a state s, that minimizes some function $f(s)$. This function $f(s)$ is often referred to as the cost function, with realistic design scenarios often having numerous parameters and possibly a complex cost function. The SA heuristic, at each iteration, generates a new state s_j and evaluates the energy level, or cost of the new state E_j, where the new state is commonly termed a neighboring state. It then compares the cost of the current state E_i, with the cost of the new state E_j. If the new state is more stable, that is, has a lower cost $E_j < E_i$, the current state is replaced by the new state. Otherwise, the acceptance criterion from the Metropolis–Hastings algorithm is used:

$$P(\text{accept}) = e^{-\frac{E_j - E_i}{T}}, \tag{5.32}$$

where T represents the temperature parameter. When the temperature is high, at the

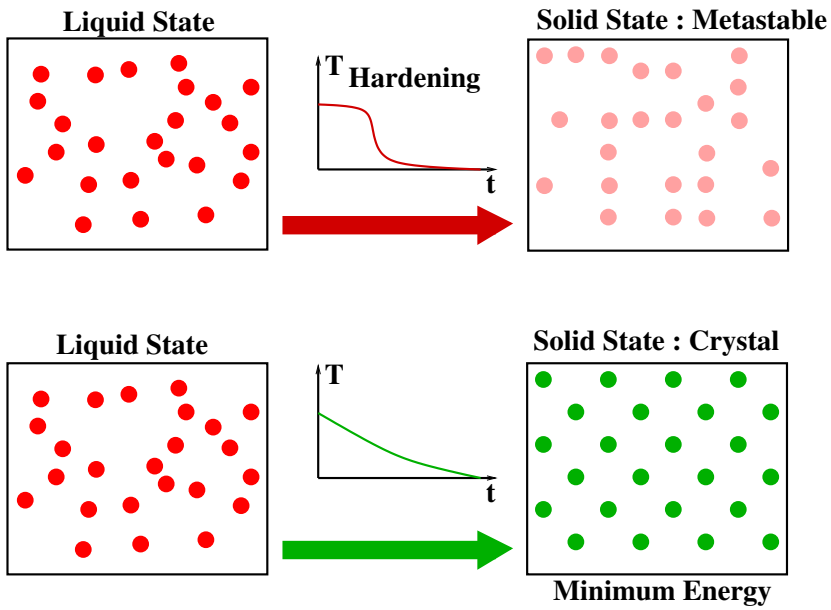

FIGURE 5.11: Visualization of the Annealing Process (Delahaye, Chaimatanan, and Mongeau, 2018)

start of the cooling process, the probability of accepting a worse solution is high, allowing the heuristic to explore, escaping local minima. When the temperature is low, the probability of accepting a worse solution is low, allowing the heuristic to dive deeper and exploit the current minima.

Drawing reference back to our example in Figure 5.8, we performed the same experiment, of minimizing $f(x) = x \sin x, x \in [0, 12]$, but using SA instead of GD. The results are given in Figure 5.12. We see that using SA, the heuristic probabilistically escaped the local minimum, obtaining the global minimum as the final solution. This is an important feature in avoiding getting trapped in local minima, given that our cost function would contain local minima as shown in Proposition 5.1.7. This may also be paraphrased by stating that SA is a global optimization algorithm mainly used for highly combinatorial problems which is the case of the WFR problem, where the sequence of flights has a bearing on the quality of the solution.

An efficient SA heuristic requires appropriately defining four fundamental components (Rutenbar, 1989):

1. **Configurations**: A model of all legal values that variables may take on. These represent the possible problem solutions over which we will search for a good answer. We also term this as the state space of a problem.

2. **Move set**: A set of allowable moves that will permit us to reach all feasible states. Ideally, this move is easy to compute and not too different in energy level from

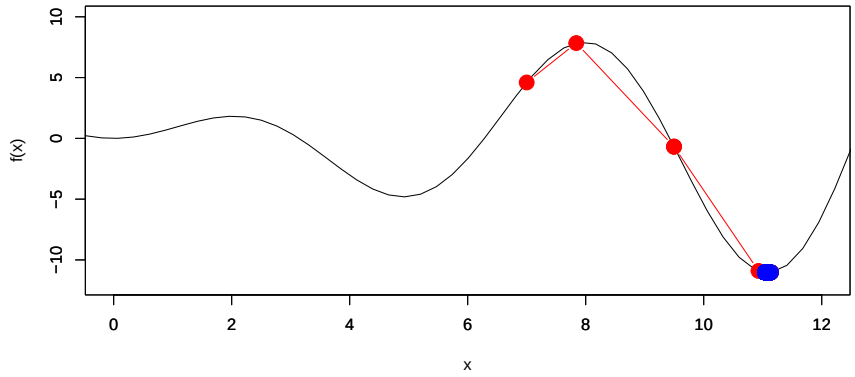

FIGURE 5.12: Graph of $f(x) = x \sin x$, Optimized With SA

the current state, hence the name neighbor. These moves are the computations that we must perform to generate the next state from the current state at each iteration.

3. **Cost function**: To measure how good a given configuration is, and serves to guide the heuristic to a good solution based on the problem requirements.

4. **Cooling schedule**: To anneal the problem from a random solution to a good, frozen state. In particular, we need a high initial temperature, which may be heuristically determined, a temperature decrement rule, and a termination criterion for the SA heuristic.

We discuss in detail how these components are determined in Sections 5.4.1 and 5.4.2.

Let s describe the state of a system, a decision vector. Let $f(s)$ measure the cost of that state, and let T record the current temperature of the annealing process. The SA heuristic is then summarized in Algorithm 10, and explained as follows.

In Algorithm 10, let s_0 and T_0 represent the initial state and initial temperature respectively, $f(\cdot)$ the cost function, ζ a small constant such as 0.0001, and k the number of iterations SA has completed. `random[0,1]` is defined as a value drawn uniformly at random from the interval between 0 to 1. We also define the acceptance function as follows:

$$P(E_i, E_j, T) = \begin{cases} 1 & \text{if } E_i > E_j, \\ e^{-\frac{E_j - E_i}{T}} & \text{otherwise.} \end{cases} \tag{5.33}$$

For application to the WFRP, the cost function $f(s)$, neighborhood generation function `neighbor`(s), and temperature decrement function `cool`(T, k) are described in the succeeding subsections.

Algorithm 10: Simulated Annealing

$s_i \leftarrow s_0$;
$E_i \leftarrow f(s_i)$;
$T_0 \leftarrow$ `heat`;
$T \leftarrow T_0$;
$k \leftarrow 0$;
while $T > \zeta T_0$ **do**
  $T =$ `cool(T, k)`;
  $s_j =$ `neighbor`(s_i);
  $E_j = f(s_j)$;
  if $E_j < E_i$ *OR* $P(E_i, E_j, T) \geq$ `random[0,1]` **then**
    $s_i = s_j$;
    $E_i = E_j$;
  end
  $k = k + 1$;
end

5.4.1 Choice of Cost Function

The cost function draws inspiration from the papers (Ma, 2019; Rapaya, Notry, and Delahaye, 2021), and fruitful discussions with Prof. Daniel Delahaye. The cost function, in line with the formulation in Section 5.1, is a weighted sum of the measure of conflicts and delays, and is hence formulated as:

$$c = w_1 \sum_{n \in \mathcal{N}} S_n + w_2 \sum_{f \in \mathcal{F}} \left(c_f h_f + \left| t_{f|P_f|} - \widetilde{t}_{f|P_f|} \right| \right) \tag{5.34}$$

Here, S_n denotes the score or cost at node n, and is equal to the sum of the number of conflicts and the badness of each conflict. The badness of each conflict is equal to one minus the ratio of separation achieved to the separation required, that is, $badness = 1 - |t_{fj} - t_{f'j'}|/\Delta_{n_j}$. The badness measure is necessary to assist the SA heuristic in resolving conflicts through guidance of achieving a progressively greater separation. The parameters $w1$ and $w2$ are weighting constants, experimentally determined to be set at $w_1 = 10000$ and $w_2 = 1$, with time variables t expressed in hours. The holding cost per hour is set at $c_f = 3$ for airborne holding, and $c_f = 1$ for ground holding.

We impose a very large penalty such that the conflicts penalized as part of the objective will function as hard constraints in the SA algorithm. This ensures that the SA heavily favors conflict-free solutions.

The use of absolute deviation to replace the earlier squared deviation in the GDA

heuristic is purely a design choice. Using the squared deviation was necessary in the GDA as we required a strongly convex function to use descent methods, with the proof of convergence of descent methods provided in (Boyd and Vandenberghe, 2004). However, for the SA heuristic, both functions, absolute and squared, are reasonable alternatives.

5.4.2 Neighbor Generating and Temperature Control Functions

The neighborhood search function $neighbor(s)$ determines the set of moves that will permit the process to potentially reach all neighboring states. We define the neighborhood search function in Algorithm 11, for a single flight f currently in state s. Here, the variables that can be modified, β and h, again represent the flexibility of a flight to undergo speed changes and holding.

Algorithm 11: `neighbor(`s_i`)`

$s_j \leftarrow s_i$

$h_f, \beta_f \leftarrow s_i$

if *l1* < `random[0,1]` < *r1* **then**

  if *first free leg is a departure* **then**

    $h_f =$ `randomInt[0, max ground holding];`

  else

    $h_f =$ `randomInt[0, max airborne holding];`

  end

end

if *l2* < `random[0,1]` < *r2* **then**

  $j =$ `randomInt[0, num free legs];`

  $k = j +$ *num fixed legs*;

  $\beta_{fk} =$ `random[0,1];`

end

$s_j \leftarrow h_f, \beta_f$

return s_j

Early in our investigation, we set the maximum ground holding to 1800 seconds, and the maximum airborne holding to 600 seconds, as these are values that are often used in practice (travelperk, 2025; Stoen, n.d.). However, these values led to many unresolved conflicts for the datasets we had constructed. Subsequently, we set both the maximum ground and airborne holding to 14400 seconds, or four hours. The results as presented later in Section 5.5, will include results only of the latter maximum hold times. The parameters $l1, r1, l2$, and $r2$ are used to probabilistically select which

changes, holding and/or speed regulation, will occur. We set $l1 = 0$, $r1 = 0.6$, $l2 = 0.4$ and $r2 = 1$. This approach allows for the possibility that, in each iteration, either one change or both changes may occur. The function `randomInt[0, val]` returns a random integer uniformly selected from the range 0 and val inclusive.

We now discuss the control procedure for the temperature T. At the start of the SA heuristic, we sought a temperature that can accept a worse solution with approximately 0.8 probability. We adopt the process from (Delahaye, Chaimatanan, and Mongeau, 2018) for determining the initial temperature, as described in Algorithm 12.

Algorithm 12: `heat`

$T \leftarrow 10$;
$acceptCount \leftarrow 0$;
while *(acceptCount / M) < p* **do**
  $T \leftarrow T * \gamma$
  $acceptCount = 0$;
  for $k \leftarrow 1$ **to** N **do**
    $E_i = f(s_0)$;
    $E_j = f(\texttt{neighbour}(s_0))$;
    if $P(E_i, E_j, T) \geq$ `random[0,1]` **then**
      $acceptCount = acceptCount + 1$;
    end
  end
end
return T

For Algorithm 12, let M denote the desired number of iterations, which we set at 500, γ denote the temperature control parameter, which we set at 1.1, $f(\cdot)$ the cost function, and p the initial acceptance probability of a worse solution, which we set at 0.8. The temperature was initialized at 10 as a design choice (any low value may suffice).

For the decreasing temperature scheme, we use a geometric cooling function (Kirkpatrick, Gelatt, and Vecchi, 1983), reducing the temperature by multiplying it by a constant $\alpha < 1$. The cooling function is described by Algorithm 13, with the cooling parameter α set at 0.99. The cooling function `cool(T, k)` is applied on every iteration k to the current temperature T. As mentioned in the previous section, the termination criterion for the SA heuristic is based on the temperature. For our experiment, we lower the temperature geometrically every $M = 500$ iterations, and terminate the heuristic when the temperature is below 0.0001 times of the initial temperature determined by Algorithm 12, that is, we end SA if $T < 0.0001T_0$.

Algorithm 13: `cool(T, k)`

if $k\%M == 0$ **then**
  $T \leftarrow \alpha * T$;
end
return T

5.4.3 Practical Issues of a Simulated Annealing Approach

One problem of the convergence of the SA heuristic is that it converges to the optimal solution, but this only happens asymptotically, that is, when the number of iterations approach infinity. Anily and Federgruen (Anily and Federgruen, 1987) proved the following properties for a SA heuristic, using inhomogeneous Markov chain theory:

1. *Reachability of the set of global optima.* The set of global optima is reached from every starting solution with probability 1.
2. *Asymptotic independence of starting solution.* The dependence of the distribution of the state s_k with respect to the starting solution vanishes as $k \to \infty$.
3. *Convergence in distribution.* s_k converges in distribution.
4. *Convergence to a global optimum.* The algorithm converges to the set of global optima with probability 1.

These were proved under the following necessary and sufficient conditions:

1. The acceptance probability function must, for any iteration k, allow any hill-climbing transition, i.e., moving to a solution with higher cost, to occur with positive probability.
2. The acceptance probability function must be bounded and asymptotically monotone, with limit zero for hill-climbing solution transitions.
3. In the limit, the stationary probability distribution must have zero probability mass for every non-globally optimal solution.
4. The probability of escaping from any locally (but not globally) optimal solution must not approach zero too quickly.

In practice, the convergence conditions will be approximated by parameters such as the number of iterations and cooling schedule, and will terminate in finite time. Given the finite time condition, condition 3 will be violated, hence convergence to a global optimum is not guaranteed in practice.

A potential improvement that one could consider, is to increase the rate of convergence via selective simulated annealing, as described in (Lavandier et al., 2021). In their work, they first identified the flight with the worst cost, and applied a threshold to identify all high-cost flights within 80% of the worst cost. Subsequently, during the neighbor generation step, only the flights that were considered high-cost were considered for mutation. In this manner, they were able to speed up convergence to the optimum, as the SA heuristic does not waste computational resources to mutate flights that provide little to no contribution to the overall cost function. This insight led to a significant speedup, which was necessary as they considered a very large number of flights, 8,800 flights in total.

5.5 Computational Results and Comparisons of GDA and SA Algorithms

The data sources used are documented in Section 3.6. Particular to the WFR, the dataset used for our experimental runs are published flight plans for flights arriving to, or departing from the ASEAN Plus region on 1 Oct 2023. For the results below, we reiterate that a flight is completed from the perspective of the FIR of interest as soon as the last leg is frozen, and the current simulation clock exceeds the frozen TTO of the last leg. We model a look ahead period of two hours, such that all flights with at least one active or frozen leg during the next two hours will be added into consideration for the optimization algorithm, and we set the time shift for the sliding window at five minutes, that is, we revise the free legs of flights for the next two hours, at every five minute interval (see Rolling Horizon Concept in Section 3.1). For the results in Tables 5.1 and 5.2, we are optimizing flights arriving or departing from the ASEAN Plus region on 1 Oct 2023, from 03:00:00 to 05:00:00 UTC, or 11:00:00 to 13:00:00 UTC +8, which is a busy period for flights in the ASEAN Plus region.

The regular airspace capacity is determined quantitatively using OAG data to obtain the capacity envelope for number of arrivals, departures and total operations at each airport node, and the maximum number of operations was used as the capacity for waypoint nodes. These computations are similar to the method employed in (Tan, 2021). The reduced airspace capacity is computed by synthetically reducing the original airspace capacity at arrival and departure nodes to 0.75 of their original levels, rounded up to an integer value. These capacity values are then used to compute the required separation time between aircraft by dividing the number of seconds in an hour by the number of flights in an hour. The total number of active flights for the regular and reduced capacity airspace scenarios are fixed at 2897 and 2990 respectively.

The PC specifications used to run all the code are Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz with 16 GHz installed RAM. The code was written in R for GDA, and Java for SA. The main results are presented in Tables 5.1 and 5.2. The results tables contain the following columns:

- **Scenario and Algorithm**: Either the regular or reduced airspace capacity scenario, and whether pre-sequencing was utilized in the GDA.
- **No. Of Conflicts**: The number of conflicts remaining, that fail to observe the required separation time, summed over all FIRs in the ASEAN Plus region.
- **Sum Of Absolute Deviation**: The sum of absolute deviation time from preferred time at the last leg of flight in the FIR (i.e. FIR exit), summed over all FIRs in the ASEAN Plus region. $\sum_f \left| t_{f|P_f|} - \widetilde{t}_{f|P_f|} \right|$. Expressed in hours.
- **Mean Absolute Deviation**: The sum of absolute deviation, divided by the number of active flights. Expressed in hours.
- **Run Time**: Computational run time taken to optimize flight trajectories in the FIR, computed using the R library microbenchmark. Expressed in seconds.

TABLE 5.1: Optimization Results For GDA on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset

Scenario and Algorithm	No. of Conflicts	Sum of Absolute Deviation (h)	Mean Absolute Deviation (h)	Run Time (s)
Regular Capacity, without pre-sequencing	19	362.70	0.125	8.37
Reduced Capacity, without pre-sequencing	30	811.33	0.271	8.41
Regular Capacity, with pre-sequencing	0	371.60	0.128	8.88
Reduced Capacity, with pre-sequencing	0	829.81	0.278	9.27

The results show that both GDA and SA successfully achieve conflict-free solutions. In particular, we observe that the pre-sequencing approach described in Section 5.3.4 is effective in resolving all conflicts when using the GDA. As anticipated, the sum of absolute deviations is higher in the reduced airspace capacity scenario, attributed to the greater minimum separation required between aircraft. Notably, SA outperforms GDA in minimizing the sum of absolute deviations, achieving reductions of 10.5% and 5.6%

TABLE 5.2: Optimization Results For SA on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset

Scenario	No. of Conflicts	Sum of Absolute Deviation (h)	Mean Absolute Deviation (h)	Run Time (s)
Regular Capacity	0	336.14	0.116	88.02
Reduced Capacity	0	786.19	0.263	74.26

for the regular and reduced capacity scenarios, respectively. This performance difference stems from GDA's convergence to local optima solutions. Despite this limitation, GDA's ability to generate solutions quickly makes it relevant in cases that running multiple simulations across extended time periods is required. Interestingly, both models predict a similar order-of-magnitude increase in the sum of absolute deviations when transitioning from regular to reduced capacity. Specifically, the GDA predicts an increase by a factor of 2.23, while the SA predicts an increase by a factor of 2.33.

For visual inspection of the individual trajectories of aircraft and congestion at nodes, a plot of node against TTO is useful. We provide the congestion plot for a partial trajectory of an aircraft going from WARR (Juanda International Airport) to WSSS (Changi Airport) in Figure 5.13. On the congestion plot, the width of the bars represent the required separation time, and the colors indicate whether a flight has been completed, i.e., archived and no longer under consideration by the optimization algorithms, or active. In Figure 5.13, there are no overlaps between bars, indicating that the current trajectory plan is conflict-free. Also, the interpretability of the plot would allow a developer to pinpoint specific patterns due to congestion, or different capacity scenarios. For example, the large gap between the blue bar and red bar on the mixed mode channel, WSSS-M, stands out. To further investigate, we can question why the active flights on the departure channel WSSS-D, are not assigned to WSSS-M, and on inspecting the active flights, we discovered that despite these flights being considered active, they are on the departure channel and have their TTOs and channels frozen. Furthermore, checking the trajectories table, we will also realize that the selected aircraft in blue is an arrival aircraft, and is unable to arrive earlier due to constraints on its speed, and not capacity constraints at WSSS-M. We contend that the congestion plot is an efficient method for developers and ATCs to rapidly evaluate airspace operations.

Additionally, we present in Tables 5.3, 5.4, 5.5, and 5.6, the results segregated by FIR for both the GDA and SA methods. Under the regular capacity scenario, we observe that the FIRs with the largest mean absolute deviations are Hong Kong, Jakarta, and Singapore (for both GDA and SA). This might be attributed to the fact that these FIRs encompass highly congested airports, which require delays even under regular

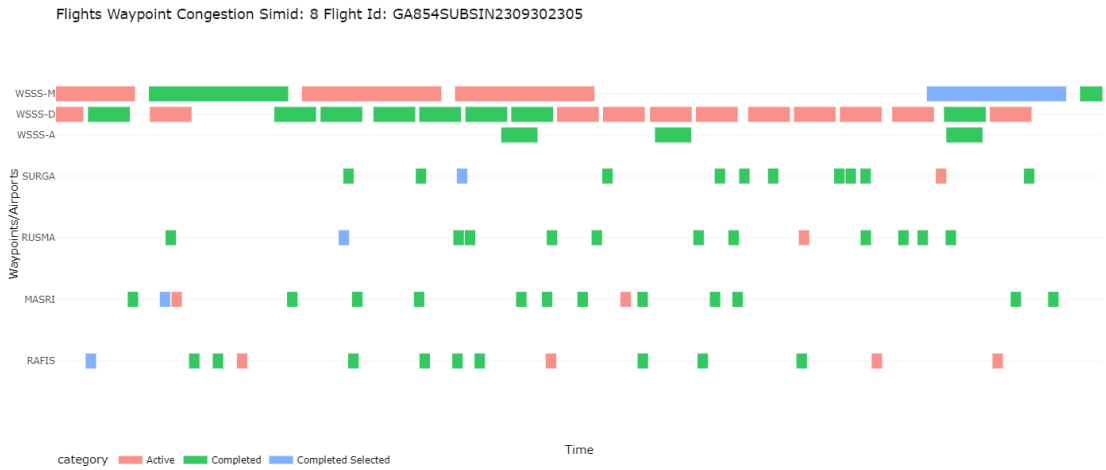

FIGURE 5.13: Congestion Plot of Nodes Leading To Arrival At WARR (Juanda International Airport)

capacity conditions. In the reduced capacity scenario, in addition to Hong Kong and Singapore, the Ho Chi Minh and Kuala Lumpur FIRs also rank among the regions with the largest deviations. This is likely due to their central location in Southeast Asia, making their airspace more susceptible to congestion as capacity decreases. Interestingly, while Jakarta still shows larger mean deviations under reduced capacity, it no longer ranks among the top FIRs. This could be because the Jakarta FIR primarily accommodates traffic passing through Indonesia and, unlike the other FIRs mentioned, does not occupy a central position in the region making it less affected by capacity reductions.

5.5.1 Comparison of Results Against Exact Methods

In previous studies, prior to the formulation and solving the WFR heuristically, via the GDA and SA algorithms, we had experimented with exact methods using the CPLEX solver, interfaced through R. During our computational experiments, we discovered that many FIRs had prohibitively long run times, which motivated the use of heuristics to solve the WFR. More recently, there has been significant advancement of solvers. For example, Gurobi recorded a 91x speed up since version 1.1 (Gurobi, 2025). This prompted us to revisit the use of exact methods as the reduction in run times may render exact methods a feasible choice for the WFR. We implemented the mathematical model from Equations 5.9 to 5.14 in Gurobi, interfaced through Python. The only difference between the earlier formulation and the one programmed in Gurobi is replacing the exponentiation δ function on both sides as in Equation 5.14, by a constraint on the absolute difference in time to reduce the computational burden on the solver, while still

TABLE 5.3: Optimization Results For GDA on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset, Regular Capacity With Pre-sequencing

FIR	Region	No. of Flights	No. of Conflicts	Sum of Absolute Deviation (h)	Mean Absolute Deviation (h)	Run Time (s)
VHHK	Hong Kong	203	0	46.50	0.229	0.73
WIIF	Jakarta	362	0	64.19	0.177	0.92
WSJC	Singapore	239	0	41.96	0.176	0.69
WAAF	Ujung Pandang	428	0	57.85	0.135	0.78
VTBB	Bangkok	289	0	33.72	0.117	0.72
VVVV	Hanoi	156	0	17.22	0.11	0.66
VVTS	Ho Chi Minh	225	0	24.47	0.109	0.90
VLVT	Vientiane	57	0	6.19	0.109	0.35
ZJSA	Sanya	241	0	25.81	0.107	0.56
RPHI	Manila	234	0	20.82	0.089	0.81
WMFC	Kuala Lumpur	231	0	19.26	0.083	0.61
VYYF	Yangon	95	0	6.12	0.064	0.39
WBFC	Kota Kinabalu	90	0	5.75	0.064	0.40
VDPP	Phnom Penh	47	0	1.75	0.037	0.36
TOTAL		2897	0	371.60	0.128	8.88

TABLE 5.4: Optimization Results For GDA on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset, Reduced Capacity With Pre-sequencing

FIR	Region	No. of Flights	No. of Conflicts	Sum of Absolute Deviation (h)	Mean Absolute Deviation (h)	Run Time (s)
VHHK	Hong Kong	215	0	117.47	0.546	0.86
VVTS	Ho Chi Minh	233	0	88.75	0.381	0.77
WMFC	Kuala Lumpur	244	0	73.39	0.301	0.69
VVVV	Hanoi	154	0	45.30	0.294	0.70
WSJC	Singapore	244	0	69.81	0.286	0.69
VTBB	Bangkok	299	0	83.18	0.278	0.77
WAAF	Ujung Pandang	443	0	116.78	0.264	0.80
WIIF	Jakarta	357	0	87.24	0.244	0.93
ZJSA	Sanya	265	0	63.22	0.239	0.57
RPHI	Manila	235	0	46.31	0.197	0.86
VLVT	Vientiane	60	0	10.45	0.174	0.38
VDPP	Phnom Penh	53	0	7.72	0.146	0.38
VYYF	Yangon	94	0	10.47	0.111	0.44
WBFC	Kota Kinabalu	94	0	9.72	0.103	0.43
TOTAL		2990	0	829.81	0.278	9.27

TABLE 5.5: Optimization Results For SA on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset, Regular Capacity

FIR	Region	No. of Flights	No. of Conflicts	Sum of Absolute Deviation (h)	Mean Absolute Deviation (h)	Run Time (s)
VHHK	Hong Kong	203	0	36.89	0.182	7.29
WIIF	Jakarta	362	0	56.65	0.157	5.53
WSJC	Singapore	239	0	31.62	0.132	5.80
WAAF	Ujung Pandang	428	0	48.20	0.113	5.33
ZJSA	Sanya	241	0	24.22	0.100	5.07
VVVV	Hanoi	156	0	15.16	0.097	8.35
VLVT	Vientiane	57	0	5.17	0.091	5.16
VVTS	Ho Chi Minh	225	0	19.95	0.089	7.85
VTBB	Bangkok	289	0	25.35	0.088	6.45
WMFC	Kuala Lumpur	231	0	18.80	0.081	7.04
RPHI	Manila	234	0	15.22	0.065	7.59
WBFC	Kota Kinabalu	90	0	5.44	0.060	5.03
VYYF	Yangon	95	0	5.42	0.057	5.63
VDPP	Phnom Penh	47	0	1.87	0.040	5.90
TOTAL		2897	0	309.96	0.107	88.02

TABLE 5.6: Optimization Results For SA on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset, Reduced Capacity

FIR	Region	No. of Flights	No. of Conflicts	Sum of Absolute Deviation (h)	Mean Absolute Deviation (h)	Run Time (s)
VHHK	Hong Kong	215	0	98.26	0.457	5.97
VVTS	Ho Chi Minh	233	0	76.00	0.326	6.18
WSJC	Singapore	244	0	66.01	0.271	5.37
WMFC	Kuala Lumpur	244	0	62.19	0.255	5.62
VVVV	Hanoi	154	0	38.15	0.248	6.42
VTBB	Bangkok	299	0	72.45	0.242	5.23
ZJSA	Sanya	265	0	61.24	0.231	4.90
WAAF	Ujung Pandang	443	0	100.60	0.227	4.94
WIIF	Jakarta	357	0	77.23	0.216	5.58
VLVT	Vientiane	60	0	11.66	0.194	3.95
RPHI	Manila	235	0	41.34	0.176	6.15
VDPP	Phnom Penh	53	0	7.76	0.146	4.76
WBFC	Kota Kinabalu	94	0	9.79	0.104	4.36
VYYF	Yangon	94	0	9.70	0.103	4.81
TOTAL		2990	0	732.38	0.245	74.26

TABLE 5.7: Optimization Results Using the Gurobi Solver on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset, Regular Capacity

FIR	Sum of Absolute Deviation (h)	No. of Conflicts	Run Time (s)	Sum of Absolute Deviation (h)	No. of Conflicts	Run Time (s)
WIIF	46.32	0	10	46.22	0	600
WAAF	43.86	0	10	43.86	0	600
ZJSA	22.88	0	6.81	22.88	0	8.05
WSJC	29.23	0	10	29.23	0	600
WMFC	16.47	0	3.05	16.47	0	3.31
VVTS	15.58	0	8.89	15.58	0	8.73
VHHK	31.98	0	10	31.98	0	600
RPHI	10.77	0	10	10.67	0	134.9
VVVV	12.48	0	4.3	12.48	0	4.56
VLVT	5.08	0	0.19	5.08	0	0.25
VTBB	19.48	0	10	19.48	0	600
VDPP	1.70	0	0.17	1.70	0	0.13
WBFC	5.06	0	0.25	5.06	0	0.31
VYYF	5.37	0	0.52	5.37	0	0.64
TOTAL	266.26	0	84.18	266.05	0	3160.88

ensuring conflict-free trajectories. The following formulation is used instead:

$$\left|t_{fj} - t_{f'j'}\right| \geq \Delta_{n_j}, \forall \left(f, f', j, j'\right) \in \mathcal{C}^I, \tag{5.35}$$

and subsequently reformulated using Big M constraints as:

$$t_{fj} - t_{f'j'} \geq \Delta_{n_j} - b_{iji'j'}M, \forall \left(f, f', j, j'\right) \in \mathcal{C}^I, \tag{5.36}$$

$$t_{f'j'} - t_{fj} \geq \Delta_{n_j} - (1 - b_{iji'j'})M, \forall \left(f, f', j, j'\right) \in \mathcal{C}^I, \tag{5.37}$$

where $b_{iji'j'}$ is a binary variable that enforces the exclusivity of constraints, ensuring that exactly one of Equation 5.36 or Equation 5.37 holds, while making the other redundant. The value of M here is set to a sufficiently large value of 10^6.

Similar to our earlier optimization via CPLEX, the run time to solve to optimality were prohibitively long, as depicted in Figure 5.14, where the optimization program failed to close the optimality gap for the FIR WSJC even after 1 day (86400 seconds). The issue of non-convergence between the lower and upper bounds has also been reported in (Ribeiro et al., 2018). They suggested a possible solution of splitting constraints with more than one binary variable, to cut off non-integer solutions. However, due to the

TABLE 5.8: Optimization Results Using the Gurobi Solver on the WFR for 1 Oct 2023 03:00:00 UTC, Within ASEAN Plus Region Dataset, Reduced Capacity

FIR	Sum of Absolute Deviation (h)	No. of Conflicts	Run Time (s)	Sum of Absolute Deviation (h)	No. of Conflicts	Run Time (s)
WIIF	67.88	0	10	65.00	0	600
WAAF	89.88	0	10	89.88	0	600
ZJSA	57.27	0	10	57.27	0	600
WSJC	57.31	0	10	57.31	0	600
WMFC	59.51	0	10	59.51	0	600
VVTS	67.87	0	10	67.46	0	600
VHHK	148.39	0	10	90.32	0	600
RPHI	44.79	0	10	30.11	0	600
VVVV	34.27	0	10	34.27	0	11.27
VLVT	9.54	0	0.12	9.54	0	0.18
VTBB	130.63	0	10	59.02	0	600
VDPP	7.58	0	0.16	7.58	0	0.11
WBFC	7.87	0	0.26	7.87	0	0.35
VYYF	9.63	0	0.52	9.63	0	0.78
TOTAL	792.42	0	101.06	644.76	0	5412.69

absence of two binary variables in a single equation, this method is not applicable.

An important feature of the Gurobi solver that facilitates a meaningful comparison between exact methods and the heuristics presented in this chapter is its support for early termination while still returning the best incumbent solution. We report these results in Tables 5.7 and 5.8, where the optimization program was run with a time limit of 10 and 600 seconds respectively. The best solution found within the time limit would be returned, and the variables would be used to compute the TTOs. The solution quality were subsequently evaluated in the same manner as the GDA and SA, on the basis of deviation from preferred time, number of conflicts, and computational runtime.

We observe that the Gurobi solver was able to generate conflict-free trajectories within 10 seconds, for both capacity scenarios, similar to GDA and SA heuristics. Under the regular capacity case in Table 5.7, 12 out of 14 of the FIRs had equal values for the sum of absolute deviation, except for WIIF and RPHI, of which the difference between the solutions under the 10 and 600 second time limit was less than 1%. For the regular capacity case, the results indicate that the 10 second time limit is sufficient to generate TTOs that are close to the optimal solution. The solver subsequently spends

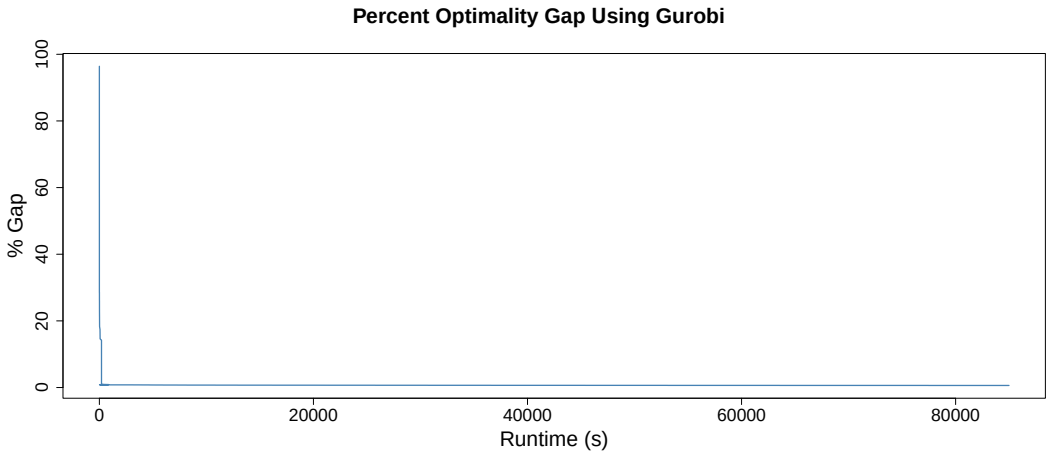

FIGURE 5.14: Percent Optimality Gap Using the Gurobi Solver

the majority of computation time attempting to improve the upper bound to prove optimality; however, this theoretical bound offers limited value for the operational capabilities of the WFR. Given the increased complexity, the results for the reduced capacity case are more interesting, where the results reveal a substantial gap between the sum of absolute deviation time under the 10 and 600 second time limit. In particular, comparing the solutions under the 10 and 600 second time limit runs, the solutions for FIRs VTBB, VHHK, and RPHI had a gap of 121%, 64%, and 49% respectively. This large optimality gap indicates a significant challenge for the Gurobi solver when handling a large number of constraints and variables, given that the solution for these three FIRs with a 10 second time limit imposed, were comparatively worse than the GDA (Table 5.4), where each FIR was solved in under one second.

In Table 5.9, we consolidate and report the results of the proposed GDA, SA and exact methods. While GDA was able to generate conflict-free trajectories with the shortest run time, of about 10 times faster than the next algorithm, the drawback is the lack of flight sequencing leading to larger deviation from preferred time as compared to the SA and exact methods. For the GDA, due to its ability to rapidly generate conflict-free trajectories, it may see use in tactical flight planning, where rapid solve times are preferred, say in the event of a medical emergency and immediate rescheduling is required. For the SA, while it generated solutions with lower deviation than the GDA, its solution for the regular capacity case was still 16% worse than the exact methods under the 10 second time limit. However, under the reduced capacity case, which is harder to solve due to the increase in number of conflicts, it performed better than the Gurobi solver, achieving a deviation that outperformed the exact methods by 8%, given a similar computational time limit. This suggests that SA is more robust when handling highly complex problems, offering a reasonable runtime and good solution

quality, highlighting its suitability for the tactical planning phase. In contrast, exact methods, given sufficient computational time, have demonstrated the ability to significantly outperform both GDA and SA in terms of solution quality. This provides strong evidence of their suitability for producing high-quality solutions during the strategic flight planning phase, in which flights are planned and adjusted months in advance of actual operations. We did not pursue this line of research on optimization methods further, as explained in Section 7.1. Instead, we adopted a Discrete Event Simulation, which better reflects the current operational capabilities of ATC under the model's assumptions of regular conditions (excluding inclement weather and medical emergencies) and circumvents several issues associated with optimization-based approaches.

TABLE 5.9: Optimization Results Summary for GDA and SA Against Exact Methods

Algorithm and Scenario	Sum of Absolute Deviation (h)	Run Time (s)
GDA, Regular Capacity	371.6	8.88
SA, Regular Capacity	310	88.02
Exact Methods, Regular Capacity (Short)	266.3	84.18
Exact Methods, Regular Capacity (Long)	266.1	3160.88
GDA, Reduced Capacity	828.8	9.27
SA, Reduced Capacity	732.4	74.26
Exact Methods, Reduced Capacity (Short)	792.4	101.06
Exact Methods, Reduced Capacity (Long)	644.8	5412.69

We also extend our analysis from the FIR level to the airport level, by computing the simulated arrival deviation times at the destination airport (i.e. arrival delays in relation to schedule). The results of the arrival delays, for the top eight airports by number of flights, is plotted in Figure 5.15. The box plots visualize the distribution of absolute delays for all aircraft that have arrived at their destination airport between 1 Oct 2023 00:00:00 UTC and 1 Oct 2023 03:00:00 UTC, and the labels on the horizontal axis indicate the capacity scenario, either regular or reduced capacity. It is clear from the plots that the delays are greater in the reduced capacity scenario. This is to be expected, that constraining the capacity under the same volume of demand, would lead to greater delays in the system.

In conclusion, to solve the complex WFR problem, we developed two solution approaches: a Gradient Descent Ascent (GDA) and a Simulated Annealing (SA) heuristic and compared the results against exact methods using Gurobi. We evaluated the performance of our modeling framework using real data from the Southeast Asia Region, including Hong Kong and Sanya. Results demonstrate that both approaches quickly converge to conflict-free solutions, with GDA requiring approximately 9 seconds and

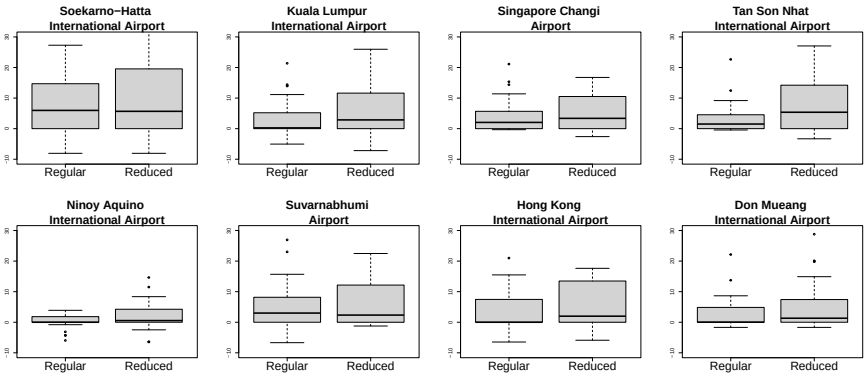

FIGURE 5.15: Boxplots of Arrival Delays Using GDA for the Top Eight Airports Sorted by Number of Aircraft

SA about 88 seconds to simulate a two hour window of operation. Despite its longer runtime, SA provides solutions that are 5–10% better in terms of optimality. We also compared our solutions against the Gurobi solver. While it is superior at solving problems with fewer constraints on the flight schedules, generating better solutions than the GDA and SA within a reasonable amount of time, for a similar computational time to the SA, the reduced capacity scenario presents problems for the Gurobi solver. Solutions of worse quality than both the GDA and SA were generated, indicating difficulty for exact methods to solve highly combinatorial problems within a short amount of time.

Additionally, both the GDA and SA heuristics came with their own set of challenges to overcome in the formulation and implementation process. For the GDA heuristic, the challenge lies in obtaining all the partial derivatives as in Section 5.3.3, and accurately translating them into code. In addition, to obtain relatively fast computational times in R, the partial gradient computation were done using matrices as described in Appendix A. This capitalizes on vectorization, which provided a substantial speed up, given that the code was written in an interpreted language, R. For the SA heuristic, the main challenge lies in devising a good neighbor generating function, as described in Section 5.4.3.

Given the benefits and drawbacks of each heuristic, a particular application may benefit from one or the other. Take, for example, a low-fidelity simulation model for analyzing air traffic movements in large networks. Such a model can be run for many different scenarios, to delve into how flights behave under various conditions and heuristics, including reduced capacity, FIR visibility conditions, and different modes of information sharing. For such a system, the GDA would be preferred due to the significantly shorter runtimes, especially if such information is used during pre-tactical

or tactical planning. However, if greater importance is placed upon the solution quality, such as analyzing future network collaborations or during the strategic planning phase, exact methods would be preferred due to its ability to generate solutions that are optimal. The SA finds itself in the middle, with solutions and computational times better than the GDA, but worse than exact methods. Given the moderate runtimes and robust solution quality, we propose that SA would also be a good choice for the tactical planning phase.

Next, a comparative analysis of two scenarios, one with regular capacity and another with reduced capacity, highlights how the model can already support ATCs with simulations to assess the potential impact of disruptive events on their FIR. While the current model focuses on decentralized ATFM systems, it lays also the foundation for investigating more collaborative regimes. These regimes align with the FF-ICE initiative proposed by the International Civil Aviation Organization (ICAO), which emphasizes both pre-departure collaboration and in-flight trajectory updates.

The results of the integrated simulator utilizing both the AFR and the WFR, for both GDA and SA heuristics, will be later compared against our alternative rule-based algorithm, the Discrete Event Simulation, and are further discussed in Chapter 7.

5.6 Summary of Notation

The notation in Table 5.10 is applicable to the entire chapter, and the notation in Table 5.11 consist of additional notation that apply only to the SA heuristic.

TABLE 5.10: Notation for the WFR

$\mathcal{F}$	Set of all flights
$\mathcal{N}$	Set of all nodes; consists of both waypoints and airports
$\mathcal{R}$	Set of all flight legs. Corresponds to the row number in the dataset
$\overline{\mathcal{R}}$	Set of frozen flight legs
$\mathcal{R}^O$	Set of free flight legs
f	Flight $f \in \mathcal{F}$
n	Node $n \in \mathcal{N}$
r	Flight Leg $r \in \mathcal{R}$
P_f	Acyclic flight path consisting of an ordered set of nodes associated with flight f: $(n_1, n_2, ..., n_{\|P_f\|})$
t_{fj}	Target Time Over (TTO) at the j^{th} node in path P_f for flight f
$\tilde{t}_{fj}$	Preferred TTO at the j^{th} node in path P_f for flight f
$\bar{t}_{fj}$	Fixed TTO of a frozen leg at the j^{th} node in path P_f for flight f
t_f^0	Earliest entry time of flight f into airspace
$\underline{d}_{fj}$	Minimum duration for flight f to transit from node n_j to node n_{j+1} in path P_f, excluding hold times
$\overline{d}_{fj}$	Maximum duration for flight f to transit from node n_j to node n_{j+1} in path P_f, excluding hold times
β_{fj}	Mixing coefficient of minimum and maximum duration, excluding hold times, for flight f to transit from node n_j to node n_{j+1} in path P_f
h_{fj}	Holding time for flight f at node n_j in path P_f
$\tilde{d}_{fj}$	The preferred duration for flight f to transit from node n_j to node n_{j+1} in path P_f
c_n	Cost per unit holding time at node n
$\lambda_{f,f',n}$	Dual variable for separation constraint between flights f and f', at node n
Δ_n	Minimum required separation time at node n
$\delta(\cdot)$	Penalty function. e.g. $\delta(\Delta_n) = e^{-\Delta_n}$
$\mathcal{C}$	Set of flight f and flight f' pairs that pass-through common node n for all nodes $n \in \mathcal{N}$
$\mathcal{C}^I$	Set of flight f and flight f' pairs that pass-through common node n for all nodes $n \in \mathcal{N}$ where one or both flight legs are free

TABLE 5.11: Additional Notation for SA Applied to the WFR

s_i	State of all variables, indexed by i
T	Current temperature of the system
E_i	Energy of the system at state i and is equivalent to the cost function at state i
S_n	Score of node conflicts at node n
h_f	Holding time of first free leg for flight f
c_f	Cost per unit holding time at first free node for flight f
ζ	Constant used to determine the SA termination criterion

Chapter 6

Discrete Event Simulation

The Air Traffic Flow Management (ATFM) problem may also be solved via a rule-based modeling approach, which solves the problem in a more conventional and predictable manner. Rule-based modeling can be described as, conditioned on the current state, and possibly previous states, the system transitions to the next state according to a predefined set of rules. We may also think of the change of state being caused by the model's choice of action.

The rule set for a given event may be deterministic or probabilistic. An example of a deterministic rule set may be to always take one step forward at each time step, whereas a probabilistic rule set may be to take one step forward with a 0.4 probability, and one step backward otherwise. Rule-based simulations are particularly effective when the analyst is able to distill the simulation model into a rule set that is significantly more compact. This simplicity allows the analyst to determine the quality of solutions and identify bottlenecks based on the current rule set. The analyst would also then be able to update the rule set as required or to understand the impact of various rule sets, commonly known as what-if analysis. Rule-based models include Markov chains, Discrete Event Simulation, and differential equations. Some fields of which rule-based modeling is applied to are queuing theory (Forbus and Berleant, 2022), signal transduction of biological cells (Sekar and Faeder, 2012), games (Finnsson and Björnsson, 2008), and healthcare (Kalashnikov, 1994).

6.1 Discrete Event Simulation

A simulation is a representation of a system that is of interest. Rather than waiting for the outcome of a system in the real-world, which may take months or years and may be operationally expensive or dangerous, software simulations can software-based simulations enable controlled experimentation and stress testing. These simulations can evaluate a wide range of scenarios and outcomes, often in a fraction of the time required for real-world observation. If the relationships between components in our system is simple enough, we may be able to use mathematical tools to solve it, such as linear

programming and probability theory. However, when the system is too complex, or when visualizing the evolution of the system is necessary, simulations are preferred.

To illustrate a practical use case of simulation, consider the decision making process for a proposed new runway at an airport. One approach would be to construct the runway and evaluate its effectiveness post-implementation. However, if the anticipated benefits fail to justify the cost, this approach becomes highly inefficient and wasteful, and deconstruction might even be necessary, compounding the costs. Simulation offers a more practical alternative by enabling a virtual comparison of airport operations with and without the new runway. For example, the simulation can estimate the number of flights that would utilize the runway, the reduction in arrival and departure delays, and the increase in airport throughput measured in flights per hour. With these insights, stakeholders can make more informed decisions regarding the project's feasibility without incurring the financial and operational risks of actual construction.

Simulation is one of the most widely adopted tool in operations research and management science, taking second place out of 13 research techniques (U. G. Gupta, 1997). Simulation has wide-ranging applications, some of these are (Law, 2015):

- Designing and analyzing manufacturing systems;
- Designing and operating transportation systems such as airports, freeways, ports, and subways;
- Evaluating designs for service organizations such as call centers, fast-food restaurants, hospitals, and post offices;
- Analyzing supply chains;
- Determining ordering policies for an inventory system;

Simulation may be conducted on both discrete and continuous events. Both models may be used on the same problem, with the difference lying in the system modeling and assumptions, and the level of granularity required of the analysis. A discrete event simulation would model one event happening after another, at discrete points in time, on the assumption that the time between events follow a fixed pattern or are not of interest. In contrast, a continuous event simulation would precisely simulate events at every point in time, assuming that state variables change continuously with respect to time. Let us take a look at the problem of scheduling flights. If we are only concerned with scheduling and analyzing TTOs at nodes, and the movement of an aircraft between nodes is assumed to be at a predetermined speed and trajectory, a discrete event simulation would be a suitable choice. Such a model would fundamentally require only cycling through events of assigning a TTO at each node, while ensuring a minimum required separation between TTOs at every node. This is because the TTOs

at nodes, are not affected by the event of a flight traversing from one node to the next, under the assumption of a predetermined speed and trajectory. On the other hand, if at each moment in the flight, the trajectory is of interest, say we want the velocity of each aircraft at any point in time, a continuous event simulator would be more appropriate as the discrete event simulator described prior does not simulate the traversal from one node to the next at each point in time.

6.2 Discrete Event Simulation Applied to the Complete Decentralized Air Traffic Flow Management Problem

Let t_{now} denote the current time. Let $\mathcal{F}$ denote the set of flights scheduled to arrive, depart, or overfly the air traffic region of interest in the course of a day or the period of interest. Let $\mathcal{F}_A$ denote the set of active flights at the current time. Flights from $\mathcal{F}$ are added to $\mathcal{F}_A$ if their scheduled departure time is less than $t_{now} + H$ for some departure planning horizon, H. Only flights in $\mathcal{F}_A$ are visible to air traffic planners in the region.

Let $\mathcal{N}$ denote the set of all airports and waypoints in the region of interest, together with supplementary nodes representing airports external to the region but referenced in the flight schedule. Each node in $\mathcal{N}$ represents either an airport or a waypoint. Let $\mathcal{A}$ denote the set of arcs connecting these nodes and representing the routes which describe the air traffic network. For each flight $f \in \mathcal{F}$, let P_f denote the shortest path through the $(\mathcal{N}, \mathcal{A})$ network from flight origin airport to flight destination airport. That is, P_f is an ordered sequence $(n_1, n_2, \cdots, n_{|P_f|})$ of nodes $n_j \in \mathcal{N}$ such that n_1 is the origin airport, $n_{|P_f|}$ is the destination airport, and each leg (n_{j-1}, n_j) is an arc in $\mathcal{A}$, for $j = 2, 3, \cdots, |P_f|$. In this study we do not consider en-route changes to the flight paths since information-sharing about such changes is not a feature of either the R0 or the R1 release of FF-ICE. Hence, we assume the flight path P_f for each flight remains unchanged throughout the simulation.

Each node $j = 1, 2, \cdots, |P_f|$, marks the *end* of a flight leg. The first leg, $j = 1$, can be thought of as the trip from the departure gate to the runway of the origin airport. We assume that the leg from n_{j-1} to n_j has a preferred duration $\widetilde{d}_{fj}$ particular to the flight f. The preferred duration of the first leg is zero ($\widetilde{d}_{f1} = 0$). In our simulation, we compute an ideal time profile for each trajectory as a whole from origin to destination with reference to the rate of ascent, cruising altitude, cruising speed, and rate of descent specific to the equipment type. We compute an upper time profile (resp. lower time profile) by scaling the cruising speed down by 10% (resp. up by 5%). For any leg, we use these three profiles to derive the preferred duration $\widetilde{d}_{fj}$ as well as maximum, $\overline{d}_{fj}$, and minimum, $\underline{d}_{fj}$, leg durations. Appendix B documents the calculations.

For any flight $f \in \mathcal{F}_{\mathcal{A}}$, let $\widetilde{t}_{f1}$ denote the scheduled departure time for the flight and let $\widetilde{t}_{fj}$ denote the scheduled TTO at each subsequent node, for $j = 2, 3, \cdots, |P_f|$, according to the recursion:

$$\widetilde{t}_{fj} = \widetilde{t}_{f,j-1} + \widetilde{d}_{fj}.$$

We refer to the scheduled TTO's for a flight as *Release R0 Target Time Over* (R0TTO) because they represent trajectories which can be deduced from pre-departure flight plans filed under release R0.

For any flight $f \in \mathcal{F}$ and node n_j for $j \in \{1, 2, \cdots, |P_f|\}$, let $r(n_j)$ denote the FIR responsible for flights over that node. Let t_{fj} denote the TTO for flight f at that node as computed under the visibility rules for FIR $r(n_j)$.

Let ${}_jt_f$ denote the TTO of the node previous to n_j in the flight plan for flight f, again, under the visibility rules for FIR $r(n_j)$. For the first leg in the flight plan, $j = 1$, we take its previous TTO, ${}_1t_f$, to be the earliest departure time of the flight. This is the scheduled departure time, $\widetilde{t}_{f1}$, but potentially updated by mixing with a CTOT. For subsequent legs, $j > 1$, we set ${}_jt_f = t_{f,j-1}$ if flight f is an R1-level flight (both origin and destination belong to R1-level FIRs) or if the previous node belongs to the same FIR (i.e., $r(n_{j-1}) = r(n_j)$) or if the previous leg has completed ($t_{f,j-1} < t_{now}$). Otherwise, we set ${}_jt_f = \widetilde{t}_{f,j-1} \vee t_{now}$. That is, if there is no information sharing for this flight and it has not yet entered FIR $r(n_j)$, then the previous TTO is set equal to the corresponding R0TTO or t_{now}, whichever is greater.

We mark a leg j as *completed* as soon as $t_{fj} \leq t_{now}$. The flight is completed as soon as the last leg is completed: $t_{f,|P_f|} \leq t_{now}$. The TTOs of completed legs are not subject to change. All other TTOs may be adjusted to satisfy timing constraints. However, to ensure an active flight satisfies the minimum required separation from any completed flight, we will archive a completed flight only after all flight legs are completed, and the time now, is greater than the TTO of all flight legs belonging to this flight, plus its minimum required separation at each resource:

$$t_{now} > t_{fj} + \delta_{fj} \quad \forall j \in P_f. \tag{6.1}$$

We add an archive operation as to remove a flight from active set $\mathcal{F}_A$ and place it in a completed set, denoted by $\mathcal{F}_C$. Note here that the DES only interacts with flights in the active set $\mathcal{F}_A$, so when the condition in Equation 6.1 is satisfied, we have that all flight legs of flight f obey the minimum separation requirements between any flight leg after t_{now}, and therefore can be removed from the active set while preserving the conflict-free nature of the processed flights.

6.2.1 Timing Constraints

A flight plan is said to be *consistent* if ${}_jt_f = t_{f,j-1}$ and *sequence-feasible* if $t_{fj} \geq {}_jt_f + \widetilde{d}_{fj}$ for all $j \in \{1, 2, \cdots, |P_f|\}$. Thus, because a previous TTO may only be an R0TTO time, a sequence-feasible flight may not be consistent across FIR boundaries unless the flight has actually entered the FIR of interest or it is an R1-level flight. Furthermore, for legs within an FIR, we assume no planned speed-up is possible but there is no upper limit to the amount of delay imposed. If $t_{fj} - {}_jt_f > \widetilde{d}_{fj}$, the difference can be composed of a leg slowdown, up to a maximum of $\overline{d}_{fj}$, and a hold time. Although it is not necessary in the discrete event simulation model, we can extract the hold time, h_{fj}, and the travel duration, d_{fj}, associated with leg j of a flight plan using the formulae:

$$d_{fj} = \min\{t_{fj} - {}_jt_f, \widetilde{d}_{fj}\},$$
$$h_{fj} = t_{fj} - {}_jt_f - d_{fj}.$$

Each leg in a flight plan must be assigned to a specific resource channel for the node at the end of that leg. Waypoints have only one resource channel but airports have three. Let $c = 0, 1, 2, 3$ represent a channel assignment of waypoint, arrival, departure, and mixed, respectively. Let c_{fj} denote the channel assignment for leg j of flight f. Departures take place on either a departure or a mixed channel, so $c_{f1} \in \{2, 3\}$. Arrivals take place on either an arrival or a mixed channel, so $c_{f,|P_f|} \in \{1, 3\}$. En route traversals take place at waypoints, so $c_{fj} = 0$ for $j = 2, 3, \cdots, |P_f| - 1$.

Airport capacity envelopes and waypoint capacities can change over the course of the simulation so let δ_{fj} denote the separation time required on channel c_{fj} as of time t_{now} if leg j is not completed. Otherwise, it will be the separation time as of the time at which it was completed. The interval $\left[t_{fj}, t_{fj} + \delta_{fj}\right]$ represents a block of time reserved on channel c_{fj} for flight f. For a visualization of these reservation blocks, we refer the reader to Figure C.13 in the appendices. Each flight can have at most one such reservation for any node so let $b(n, c, f) = [t_1, t_2]$ denote the block of time reserved at node $n \in \mathcal{N}$ on channel $c \in \{0, 1, 2, 3\}$ for flight $f \in \mathcal{F}_A \cup \mathcal{F}_C$ if it exists. A set of flight plans for completed and active flights is said to be *resource-feasible* if all such blocks on the same resource (n, c) are non-overlapping for all resources with capacity constraints. Note that flight plans can be resource-feasible from the viewpoint of each FIR but sequence-inconsistent because of visibility restrictions. Only flights in $\mathcal{F}_C$ are guaranteed to be consistent, sequence-feasible and resource-feasible.

6.2.2 Frozen Legs and Frozen Blocks

Once a leg j is completed, its TTO, t_{fj}, channel, c_{fj}, channel separation time, δ_{fj}, and reservation block $b(n_j, c_{fj}, f) = \left[t_{fj}, t_{fj} + \delta_{fj}\right]$ become immutable and its previous TTO, ${}_jt_f$, becomes fixed at $t_{f,j-1}$. But if $t_{fj} < t_{now} \leq t_{f,j+1}$, that implies the aircraft is somewhere between node n_j and n_{j+1} on the flight plan. Since arrival at node n_{j+1} may be imminent, there is presumably less flexibility to adjust TTO $t_{f,j+1}$. Accordingly, we declare such a leg to be *frozen* and restrict the ability to modify it. We cannot make it immutable in all dimensions because the act of declaring leg j completed may immediately result in a resource-infeasible set of flight plans. We give an example of when a conflict may arise if a frozen leg is immutable. When a R0-level flight completes a leg at the boundary of an FIR, the next leg immediately becomes visible in the next FIR. In particular, $t_{f,j+1}$ was previously dependent on ${}_jt_f$, but now, since the previous leg is visible, we set $t_{f,j+1} = t_{fj} + \widetilde{d}_{fj+1}$. This may result in shifting the corresponding reservation block $b(n_{j+1}, c_{f,j+1}, f)$ into conflict with another frozen reservation block, leading to an irresolvable conflict if frozen blocks were strictly immutable. To resolve such conflicts while maintaining the current solution as far as possible, we fix the channel $c_{f,j+1}$, channel separation time, $\delta_{f,j+1}$, and previous TTO ${}_{j+1}t_f = t_{fj}$ of each frozen leg and run a discrete event First-Come-First-Served algorithm on the frozen legs to update their TTOs, t_{fj}, to restore resource-feasibility. Once the frozen times have been reconciled into a resource-feasible state (ignoring non-frozen legs) we collect the reservation blocks for all completed and frozen legs into a collection of frozen blocks. All non-frozen legs will subsequently be re-scheduled by the DES to achieve a resource-feasible set of flight plans.

A consequence of freezing flight legs between node n_j and n_{j+1} on the flight plan is that the a frozen flight leg overrides any priority scheduling of flights, where the impact is observed primarily at the airports, given that they are the most congested resource. This happens when flights arrive from different waypoints, of which time taken to arrive at the airport is significantly different. Using Figure 6.1 to illustrate, we have Flight 1 approaching the airport via the northern waypoint and Flight 2 approaching the airport via the eastern waypoint, where the time taken to travel from the eastern waypoint to the airport is longer than for the northern waypoint. At the simulation step for 8:25, both Flight 1 and Flight 2 would not have their last flight leg (shown in the figure) frozen, hence Flight 1 would be scheduled in front of Flight 2 at this simulation step. Assuming a minimum required separation time of 5 minutes, Flight 1 would be scheduled for arrival at 8:35, and Flight 2 at 8:40. In the next simulation step at 8:30, only Flight 2 would be frozen. If Flight 1 is later assigned a delay of even 1 second, say to satisfy separation requirements with another flight at Node 1, it would violate the minimum separation requirements at the airport, and instead be scheduled

5 minutes after Flight 2 as depicted in Figure 6.2. This leads to the unintended phenomenon of flights using paths of which last node is further away from the destination airport receiving priority scheduling, which violates both the First-Come-First-Served (FCFS) and First-Scheduled-First-Served (FSFS) queue system. One method we have implemented to circumvent this problem is introducing a dummy airport node, just before the airport node. The final approach node is located at the same location as the airport, and connects to the airport with the distance and preferred leg time set to zero. The flight path will then be modified to connect the current second last node to the final approach node, and the final approach node to the airport. We show in Proposition 6.2.1 that including a final approach node ensures that flights will not encounter the phenomenon of unintended priority scheduling based on distance between its last waypoint and destination airport as described above.

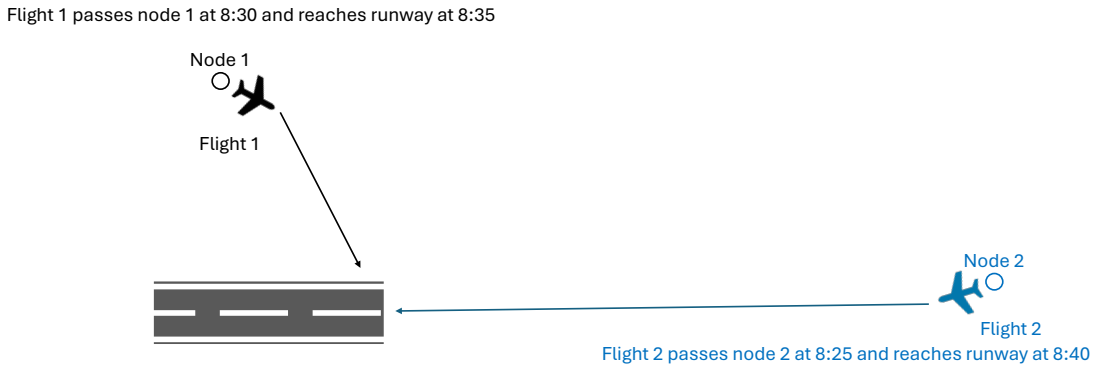

FIGURE 6.1: Negative Example of Frozen Flight Legs

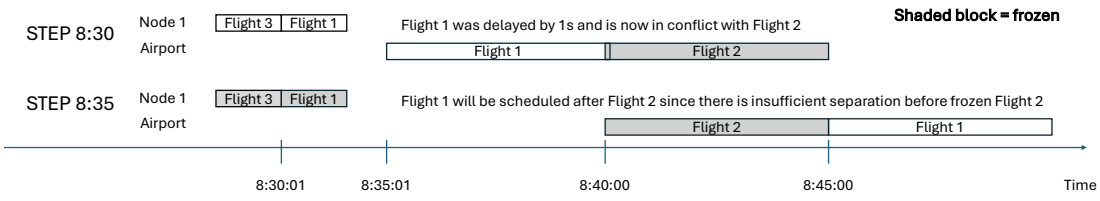

FIGURE 6.2: Negative Example of Frozen Flight Legs, TTO View

Proposition 6.2.1. *Flights will not be prioritized based on its distance from the last waypoint to the destination airport, if all flights are required to pass through a final approach node.*

Proof. Say there are k waypoints that have flights going from these waypoints, to the final approach node. We consider a set of flights $\mathcal{F}$, with increasing time from the current second last (waypoint) node to final approach node, and some flights between the waypoint and final approach node. The TTO at the final approach node would be

frozen for the flights in between the waypoint and final approach node, while for other flights that have yet to reach the current second last waypoint, their TTO at the final approach node remains adjustable. Currently the queue order at the final approach node may not follow FCFS, as we have shown earlier that freezing flight legs could lead to a violation of the scheduling rules. Additionally, the last flight leg, from the final approach node to the airport, must currently not be frozen, otherwise it would not be considered in the main loop of the DES. We recall that a flight leg would only be frozen when it between the current and next node, that is, for this proof, between the final approach node and airport node, or more formally, when $t_{f,|P_f|-1} < t_{now} \leq t_{f,|P_f|}$. Given that there is zero travel time between the final approach node and airport, the flights entering the queue at the airport would be scheduled according to the queue system chosen. As such, flights coming from all k waypoints, will have equal travel time on the last leg (dummy to airport), and not be frozen prematurely, particularly in cases where the last waypoint is abnormally far from the airport. This ensures fairness in the scheduling, where the time at which flights are frozen depends on the queue, rather than the travel time from the last waypoint to the airport. □

6.2.3 Discrete Event Simulation and FSFS Logic

Let $\mathcal{L}$ denote the set of legs in all active and completed flights, $\mathcal{F}_A \cup \mathcal{F}_C$. Let l index the legs in $\mathcal{L}$ and let $(n_l, \widetilde{d}_l, t_l, {}_l t, \widetilde{t}_l, c_l, \delta_l)$ denote the node, preferred leg duration, TTO, previous TTO, R0TTO, channel, and channel separation, respectively, currently assigned to leg l. Let $\mathcal{K}$ denote the set of all node-channel resources in the network. Let k index the resources in $\mathcal{K}$ and let $(n^k, c^k, \delta^k, u^k, b^k)$ denote the status of the resource including the node, channel, separation time, block status, and next frozen block, respectively for resource k. The block status for a resource, u^k, is Boolean variable indicating whether the resource is blocked (TRUE) or free (FALSE). If the resource is blocked, the block status also includes the time which it is blocked until. Let $\mathcal{B}$ denote the set of all frozen blocks, indexed by b, and consisting of data $(k(b), t_1(b), t_2(b))$ representing the resource index, starting time, and ending time, respectively, of the frozen block, b. Let $(\mathcal{B}, b) \leftarrow \mathcal{B} \cup \{(k, t_1, t_2)\}$ denote the operation of adding a frozen block (k, t_1, t_2) to the set $\mathcal{B}$ and returning augmented set $\mathcal{B}$ together with the index for the new block, b. The sets $\mathcal{L}, \mathcal{K},$ and $\mathcal{B}$ capture the current simulation state of legs, resources, and frozen blocks.

A simulation event is a tuple $v = (y, i, t)$ denoting the event type, y, in the set of event types Y, the index i in $\mathcal{L} \cup \mathcal{K} \cup \mathcal{B}$ indexing the data for the event, and t, the time of the event. Let Q denote an ordered set of events, sorted in increasing order of the time index. Ties in the time index are resolved arbitrarily. Let $Q \leftarrow Q \cup \{v\}$ denote the operation of adding a new event v to the queue Q and updating the queue

order. Let $Q[m]$ refer to the m^{th} element of the queue and let $(v, Q) \leftarrow Q(m)$ denote the operation of extracting the m^{th} event from the queue and returning both the m^{th} event, v, and the residual queue, Q. Let Q_M denote the master queue for the simulation consisting of all unprocessed events. Let Q_n denote the queue of events pending for assignment at node $n \in \mathcal{N}$, and let $Q_P = \{Q_n, n \in \mathcal{N}\}$ refer to the collection of pending event queues. Note that when implementing the event queues, we may simply select the event with the earliest event time in $O(n)$, rather than re-sorting the queue at each update in $O(n \log(n))$.

The goal of Algorithm 14 is to convert the current set of flight plans into flight plans which are sequence-feasible and, collectively, resource-feasible for all legs and consistent, at least for frozen legs and legs within the same FIR. The timing of frozen legs may be adjusted to reconcile resource conflicts but completed legs will be unaffected by the procedure. The scheduling logic is decentralized at the FIR level with visibility of flights outside the FIR limited by the information regime. If a resource is blocked by a previous flight, the FIR is assumed to maintain a queue of legs pending for availability of the resource and to assign the resource to the pending leg with the earliest event time — the R0TTO for R1-level flights, and the TTO for R0-level flights. The FIRs thus grant R1-level flights priority via a First-Scheduled-First-Served logic.

Algorithm 14 first reconciles the times of all frozen legs and initializes the set of frozen blocks, $\mathcal{B}$. Then it empties the master queue, Q_M, and pending queues, Q_n, at all nodes n. It also schedules the start of the first frozen block for each node and channel within that node ($c \in c(n)$) by adding a BLOCKSTART event to the master queue marking the start time of the block, $t_1(b)$. It then loops over all legs and identifies the first unfrozen leg for each flight in each FIR traversed by the flight plan. If that leg is a departure, or if the previous TTO is not visible, as captured by `prevTTOIsNotVisible`(l), then a LEGREADY event is prepared for that leg and added to the master queue, Q_M, at time $t_l \leftarrow {}_l t + \widetilde{d}_l$. This effectively simulates each FIR conducting its own decentralized operations. Once the queues have been thus initialized, the main loop of the discrete event simulation commences. Within that loop, the first event is removed from the master queue ($\big(Q_M, (y, i, t)\big) \leftarrow Q_M(1);$) and the simulation time is advanced to the time of the first event. The event is then processed by different routines according to the event type. The BLOCKSTART event marks the beginning of a frozen block. The BLOCKEND event comes at the end of the most recent frozen block on the resource. The RESERVATIONEND event comes after the required separation time of the last unfrozen leg to be assigned to the resource. The BLOCKEND and RESERVATIONEND events both mark the freeing of a resource. Each of these events will trigger a call to the `processPENDING` routine which assigns pending legs to the appropriate resource, if available. When a pending leg is assigned to a resource (within the `processPENDING`

Algorithm 14: `simulateFSFS` $(\mathcal{L}, \mathcal{K})$

$(\mathcal{L}, \mathcal{K}, \mathcal{B}) \leftarrow$ `reconcileFrozenLegs` $(\mathcal{L}, \mathcal{K})$
$Q_M \leftarrow \emptyset$;
for $n \in \mathcal{N}$ **do**
  $Q_n \leftarrow \emptyset$;
  for $c \in c(n)$ **do**
    $b \leftarrow$ `getFirstBlock`$(\mathcal{B}, n, c)$;
    if $b > 0$ **then**
      $t_{next} \leftarrow t_1(b)$;
      $y \leftarrow$ BLOCKSTART;
      $v \leftarrow (y, b, t_{next})$;
      $Q_M \leftarrow Q_M \cup \{v\}$;
    end
  end
end
for $l \in \mathcal{L}$ **do**
  if *and(not(*`isFrozen`(l)*),*`prevTTOIsNotVisible`(l)*)* **then**
    $t_l \leftarrow {}_l t + \widetilde{d}_l$;
    $y \leftarrow$ LEGREADY;
    $v \leftarrow (y, l, t_l)$;
    $Q_M \leftarrow Q_M \cup \{v\}$;
  end
end
while $Q_M \neq \emptyset$ **do**
  $\big(Q_M, (y, i, t)\big) \leftarrow Q_M(1)$;
  $t_{sim} \leftarrow t$;
  switch y **do**
    when BLOCKSTART $k \leftarrow i$; $(\mathcal{L}, \mathcal{K}, Q_M) \leftarrow$
     `processBLOCKSTART`(t_{sim}, k, Q_M);
    when BLOCKEND $k \leftarrow i$; $(\mathcal{L}, \mathcal{K}, Q_M) \leftarrow$
     `processBLOCKEND`(t_{sim}, k, Q_M);
    when LEGREADY $l \leftarrow i$; $(\mathcal{L}, \mathcal{K}, Q_M, Q_P) \leftarrow$
     `processLEGREADY`$(t_{sim}, l, n_l, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$;
    when RESERVATIONEND $k \leftarrow i$; $(\mathcal{L}, \mathcal{K}, Q_M, Q_P) \leftarrow$
     `processRESERVATIONEND`$(t_{sim}, k, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$;
  end
end
return $\mathcal{L}$

routine), a LEGREADY event is also scheduled for the next leg of the flight, provided the visibility requirements are met. In this way, it can be guaranteed that all legs are scheduled to be sequence-feasible and resource-feasible.

In the next section, Section 6.3, we describe the supporting algorithms for the discrete event simulation in greater detail and provide proofs of correctness. The results for the DES are deferred to Chapter 7, where they will be compared against the integrated simulator utilizing both the AFR and the WFR.

6.3 Supporting Algorithms for Discrete Event Simulation

Algorithm 15: `updateFrozenLeg` $(t_{sim}, l, k, \delta, \mathcal{L}, \mathcal{K}, Q_M)$

$t_{earliest} \leftarrow {}_l t + \widetilde{d}_l$;
if $t_{earliest} > t_{sim}$ **then**
  $t_l \leftarrow t_{earliest}$;
  $y \leftarrow$ FROZENSTART;
  $v \leftarrow (y, l, t_l)$;
  $Q_M \leftarrow Q_M \cup \{v\}$
else
  $b \leftarrow b^k$;
  if $b > 0$ **then**
    $t_{next} \leftarrow t_2(b)$;
  else
    $t_{next} \leftarrow t_{sim}$;
  end
  $t_{next} \leftarrow \max(t_{next}, t_{sim})$;
  $t_1 \leftarrow t_{next}$;
  $t_2 \leftarrow t_{next} + \delta$;
  $(\mathcal{B}, b) \leftarrow \mathcal{B} \cup \{(k, t_1, t_2)\}$;
  $b^k \leftarrow b$;
  $t_l \leftarrow t_{next}$;
  $\delta_l \leftarrow \delta$;
  $l' \leftarrow$ `nextLeg(`l`);`
  if `isFrozen(`l'`)` **then**
    ${}_{l'} t \leftarrow t_{next}$;
  end
end
return $\mathcal{L}, \mathcal{K}, \mathcal{B}, Q_M$

Algorithm 15 responds to a FROZENSTART event for frozen leg l assigned to resource k with separation time δ. The current TTO, t_l, is guaranteed to be less than or equal to the current simulation time, t_{sim}. If the earliest time this event can be scheduled based on the previous TTO, ${}_lt$, exceeds the current simulation time, then the FROZENSTART event is rescheduled into the future ($Q_M \leftarrow Q_M \cup \{v\}$). Otherwise, using FCFS logic, this leg gets the next available time on resource k, t_{next}. The value of t_{next} is either the current simulation time or the end of the last frozen block assigned to resource k, b^k, whichever is later. A new frozen block, b, is created to register the assignment of leg l to resource k and the resource is updated to point to this block, b^k. The frozen leg TTO, t_l, and separation time δ_l are updated as well. If there is a next leg, $l' =$`nextLeg(`l`)`, on the same flight, and that leg is also frozen, then previous TTO of the next leg, ${}_{l'}t$ is updated to the TTO, t_{next}, of the current frozen leg.

Algorithm 16: `reconcileFrozenLegs(`$\mathcal{L}, \mathcal{K}$`)`

$Q_M \leftarrow \emptyset$;
$\mathcal{B} \leftarrow \emptyset$;
for $k \in \mathcal{K}$ **do**
  $b^k \leftarrow 0$
end
for $l \in \mathcal{L}$ **do**
  if *isFrozen(*l*)* **then**
    $y \leftarrow$ FROZENSTART;
    $v \leftarrow (y, l, t_l)$;
    $Q_M \leftarrow Q_M \cup \{v\}$;
  end
end
while $Q_M \neq \emptyset$ **do**
  $\big(Q_M, (y, l, t_{sim})\big) \leftarrow Q_M(1)$;
  $k \leftarrow (n_l, c_l)$;
  $\delta \leftarrow \delta_l$;
  $(\mathcal{L}, \mathcal{K}, \mathcal{B}, Q_M) \leftarrow$ `updateFrozenLeg(`$t_{sim}, l, k, \delta, \mathcal{L}, \mathcal{K}, Q_M$`)`;
end
return $\mathcal{L}, \mathcal{K}, \mathcal{B}$

Algorithm 16 conducts a discrete event simulation to reconcile all frozen legs into a resource-feasible plan. The master queue, Q_M and the set of frozen blocks, $\mathcal{B}$, are initialized to the empty set, $\emptyset$, and all references to frozen blocks in the resource set, $\mathcal{K}$ are set to zero. A FROZENSTART event is created for each frozen leg and added to

master queue, Q_M, at its pre-computed TTO, t_l. The events are processed in order from smallest TTO to largest and each event is updated as described in Algorithm 15. The set $\mathcal{B}$ ends with a frozen block for every frozen leg on the corresponding resource.

Algorithm 17: `processBLOCKSTART` (t_{sim}, b)

$k \leftarrow k(b)$;
$u^k \leftarrow (TRUE, t_2(b))$;
$t_{next} \leftarrow t_2(b)$;
$y \leftarrow$ BLOCKEND;
$v \leftarrow (y, b, t_{next})$;
$Q_M \leftarrow Q_M \cup \{v\}$;
return $Q_M, \mathcal{K}$

Algorithm 17 processes a BLOCKSTART event for frozen block, b, The matching resource, k, is identified and the blocked status, u^k, is set to TRUE until the end time of the block $t_2(b)$. $t_2(b)$ is then extracted from the block and used to add a BLOCKEND event to the master queue, Q_M.

Algorithm 18: `isAvailable` (t_{sim}, k)

$\alpha \leftarrow$ not(u^k);
$b \leftarrow b^k$;
if *and*$(\alpha, b > 0)$ **then**
  $t_{next} \leftarrow t_1(b)$;
  $\delta \leftarrow \delta^k$;
  $\alpha \leftarrow \{t_{sim} + \delta \leq t_{next}\}$?TRUE:FALSE
end
return α

Algorithm 18 checks whether the resource k is not blocked at the current time, t_{sim}, and, if it is free, whether or not a reservation equal in length to the current separation time for the resource, δ^k, can be inserted before the next frozen block, b^k, for that resource, if such a block exists ($b^k > 0$).

Algorithm 19 assigns an unfrozen leg l to resource k at TTO t_{sim}. The channel for the leg is fixed at the channel for resource k and the required separation for the leg on that channel is set to δ. The resource is guaranteed to be free at the current time, t_{sim}. The blocked status of the resource, u^k, is thus changed to TRUE, marking the beginning of a reservation, until the end of the reservation $t_{sim} + \delta$. A RESERVATIONEND event is scheduled to occur at time $t_{sim} + \delta$. If the next leg(s) for the flight is visible (that is, if it belongs to the same FIR as the current leg or it is part of an R1-level flight) then the

Algorithm 19: `assignLegToResource` $(t_{sim}, l, k, \delta, \mathcal{L}, \mathcal{K}, Q_M)$

$c_l \leftarrow c^k$;
$t_l \leftarrow t_{sim}$;
$\delta_l \leftarrow \delta$;
$u^k \leftarrow (TRUE, t_{sim} + \delta)$;
$y \leftarrow$ RESERVATIONEND;
$i \leftarrow k$;
$t \leftarrow t_{sim} + \delta$;
$v \leftarrow (y, i, t)$;
$Q_M \leftarrow Q_M \cup \{v\}$;
while *true* **do**
  $l' \leftarrow$ `nextLeg(`l`);`
  if `isVisible`(l, l') **then**
    ${}_{l'}t \leftarrow t_{sim}$;
    $t_{l'} \leftarrow t_{sim} + \widetilde{d}_{l'}$;
    $y \leftarrow$ LEGREADY;
    $v \leftarrow (y, l', t_{l'})$;
    $Q_M \leftarrow Q_M \cup \{v\}$;
  end
  else
    break;
  end
end
return $\mathcal{L}, \mathcal{K}, Q_M$

previous TTO of the next leg, ${}_{l'}t$ is set to the TTO of the current leg and a tentative value for the next TTO, $t_{l'}$, is set to the current TTO plus the preferred duration of the next leg, $\widetilde{d}_{l'}$. This is the earliest sequence-feasible TTO for the next leg. A LEGREADY event is added to the master queue for this next leg using the tentative TTO as the LEGREADY time.

Algorithm 20 identifies the best available channel for a non-frozen leg, l, ending at node n at time t_{sim}, if an appropriate channel is available. If no appropriate channel is available, the algorithm returns NOAVAILABLECHANNEL. If the node n is a waypoint, there is only one channel possibility ($c = 0$). If the resource $(n, 0)$ is available then the best channel is 0. If the node n is not a waypoint, then it is an airport and there are three channel choices. If the leg is an arrival, then the preferred channel is the arrival channel $(c = 1)$. If that resource is available, then the best channel is 1. Otherwise, if the mixed channel $(c = 3)$ resource is available, then the best channel is 3. Similarly, if the leg is a departure leg it will be assigned first to the departure channel $(c = 2)$ if it is available or, second, to the mixed channel $(c = 3)$ if it is available.

Algorithm 21 considers all the pending leg events for node n and assigns them as many of them to appropriate available channels as possible. Recall Q_n is the collection of LEGREADY pending events for node n sorted in increasing order of the scheduled times, R0TTO. They are processed in sequence where m indexes the m^{th} element of Q_n. If the best available channel c_{best} for that element is valid (ie. not(c_{best} == NOAVAILABLECHANNEL)) then the event is removed from the pending events for node n ($\big(Q_n, v\big) \leftarrow Q_n(m)$;) and the leg is assigned to resource k matching (n, c_{best}). In this case, m will now point to the next pending event for node n. Otherwise, there is no available channel for the m^{th} event, so we increment m and repeat until all elements of Q_n have been considered. Of course, this is inefficient since at most three pending events can be assigned before all available channels become blocked. The algorithm can be easily modified to break out of the while loop when such a condition is detected.

Algorithm 22 processes a BLOCKEND event for frozen block, b. The matching resource, k, is identified, and if it was blocked until a later time, we add it back to the event queue at a later time, and exit the function. Otherwise, the blocked status, u^k, is set to FALSE. The next frozen block, b, for resource k is identified and the resource block pointer, b^k, is updated to point to it. If the next block exists ($b > 0$), the start time, $t_1(b)$, is extracted from the block and used to add a BLOCKSTART event to the master queue, Q_M. Since the resource is now free to accept another leg, the final call is to the `processPENDING` routine for the associated node.

Algorithm 23 processes a LEGREADY event for leg l at node n. It first adds a LEGPENDING event to the queue of pending events for node n. Notice that if the leg

Algorithm 20: `getBestChannel` $(t_{sim}, l, n, \mathcal{L}, \mathcal{K})$

```
c_best ← NOAVAILABLECHANNEL;
if isWaypoint(n) then
    c ← 0;
    k ← getResource(n, c);
    if isAvailable(t_sim, k) then
        c_best ← c;
    end
else
    if isArrival(l) then
        c ← 1;
        k ← getResource(n, c);
        if isAvailable(t_sim, k) then
            c_best ← c;
        else
            c ← 3;
            k ← getResource(n, c);
            if isAvailable(t_sim, k) then
                c_best ← c;
            end
        end
    else
        c ← 2;
        k ← getResource(n, c);
        if isAvailable(t_sim, k) then
            c_best ← c;
        else
            c ← 3;
            k ← getResource(n, c);
            if isAvailable(t_sim, k) then
                c_best ← c;
            end
        end
    end
end
return c_best
```

Algorithm 21: `processPENDING`$(t_{sim}, n, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$

```
m ← 1;
while m ≤ |Q_n| do
    (y, l, t) ← Q_n[m];
    c_best ← getBestChannel(t_sim, l, n, L, K);
    if not(c_best ==NOAVAILABLECHANNEL) then
        c ← c_best;
        k ← getResource(n, c);
        δ ← δ^k;
        (Q_n, v) ← Q_n(m);
        (L, K, Q_M) ← assignLegToResource(t_sim, l, k, δ, L, K, Q_M);
    else
        m ← m + 1;
    end
end
return L, K, Q_M, Q_P
```

belongs to an R1-level flight, then it uses the scheduled TTO, $\widetilde{t}_l$, or R0TTO, for this leg. Otherwise, it uses the current simulation time, t_{sim}. In this way, if there is a queue of legs pending, the legs with earlier scheduled times will receive priority, provided they are R1-level flights. This is what we refer to as the First-Scheduled-First-Served logic. If there are no R1-level flights, then the queue operates according to First-Come-First-Served logic. The algorithm then calls the `processPENDING` function to assign as many pending legs to channels at node n as possible.

Algorithm 24 is called to process the RESERVATIONEND event for resource k. The time t_{sim} marks the first time after the last leg to use this resource that the resource becomes free to assign to another leg. However, if the resource was blocked until a later time, the function simply exits without doing anything. Otherwise, the blocked status, u^k, is set to FALSE and the processPENDING algorithm is called in case there are any legs pending for this resource in the pending queue, Q_n, for node $n = n^k$.

Validity of DES Algorithm

Freezing the TTO of legs takes place outside of the DES loop as part of the rolling horizon simulation. For leg $l' \in \mathcal{L}$, let $f' = f(l')$ denote the corresponding flight and $j' = j(l')$ denote the node index within the flight plan. If $j' > 1$, let l'' denote the leg corresponding to the previous node index, $j' - 1$, for flight f'. Let $t_{l'} = t_{f'j'}$ the TTO for leg l' and let ${}_{l'}t$ denote the TTO on record for the previous node. The only properties we expect of the freezing operation are:

Algorithm 22: `processBLOCKEND` (t_{sim}, b)

$k \leftarrow k(b)$;
if $t_{sim} < u^k[2]$ **then**
  $t_{next} \leftarrow u^k[2]$;
  $y \leftarrow$ BLOCKEND;
  $v \leftarrow (y, b, t_{next})$;
  $Q_M \leftarrow Q_M \cup \{v\}$;
  return $Q_M, \mathcal{K}$
end
$u^k \leftarrow FALSE$;
$n \leftarrow n(b)$;
$b \leftarrow$ `nextBlock`$(\mathcal{B}, b^k)$;
$b^k \leftarrow b$;
if $b > 0$ **then**
  $t_{next} \leftarrow t_1(b)$;
  $y \leftarrow$ BLOCKSTART;
  $v \leftarrow (y, b, t_{next})$;
  $Q_M \leftarrow Q_M \cup \{v\}$;
end
$(\mathcal{L}, \mathcal{K}, Q_M, Q_P) \leftarrow$ `processPENDING` $(t_{sim}, n, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$;
return $Q_M, \mathcal{K}$

Algorithm 23: `processLEGREADY` $(t_{sim}, l, n, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$

$y \leftarrow$ LEGPENDING;
if *isR1Flight*(l) **then**
  $t \leftarrow \widetilde{t}_l$;
else
  $t \leftarrow t_{sim}$;
end
$v \leftarrow (y, l, t)$;
$Q_n \leftarrow Q_n \cup \{v\}$;
$(\mathcal{L}, \mathcal{K}, Q_M, Q_P) \leftarrow$ `processPENDING` $(t_{sim}, n, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$;
return $\mathcal{L}, \mathcal{K}, Q_M, Q_P$

Algorithm 24: `processRESERVATIONEND` $(t_{sim}, k, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$

if $t_{sim} < u^k[2]$ **then**
  return $\mathcal{L}, \mathcal{K}, Q_M, Q_P$
end
$u^k \leftarrow FALSE$;
$n \leftarrow n^k$;
$(\mathcal{L}, \mathcal{K}, Q_M, Q_P) \leftarrow$ `processPENDING` $(t_{sim}, n, \mathcal{L}, \mathcal{K}, Q_M, Q_P)$;
return $\mathcal{L}, \mathcal{K}, Q_M, Q_P$

1. Frozen legs are contiguous within flights: if leg l' is frozen and $j' > 1$ for flight f', then the leg corresponding to node $j' - 1$ is also frozen.
2. Frozen legs are consistent: If leg l' is frozen and $j' > 1$ then ${}_{l'}t = t_{l''}$.
3. Frozen legs are sequence feasible: if leg l' is frozen and $j' > 1$ then $t_{l'} \geq t_{l''} + \widetilde{d}_{l'}$.

As noted, a set of frozen legs may fail to be resource-feasible, and so the purpose of `reconcileFrozenLegs()` is to enforce resource-feasibility.

Lemma 6.3.1. *If a set of frozen legs is resource-feasible, then the solution will be unchanged by algorithm* `reconcileFrozenLegs()`.

Proof. The set Q_M is initialized to consist of all frozen legs l' sorted in increasing order of $t_{l'}$. Within the loop **while**$Q_M \neq \emptyset$, one event, $Q_M(1)$, is removed and at most one event (for the leg just removed) is added (in `updateFrozenLeg()`). The proof is by induction on the number of iterations the **while** loop has been performed. The induction hypothesis is that for every event that was removed up to and including iteration n no new event was generated. The legs removed in the first n iterations have TTOs which are unchanged. Furthermore, the set of resource blocks $\mathcal{B}$ after iteration n corresponds to the resource blocks associated with the original TTOs of the legs removed in the first n iterations. By assumption, there are no overlaps among these blocks. The hypothesis is trivially true for $n = 1$ because, assuming $\widetilde{d}_{l'} > 0$ the first leg to be processed cannot have a previous leg at the same or earlier time. Assuming the hypothesis is true for iteration n, we consider the next iteration. The next event to be processed (y, l', t_{sim}) must correspond to a leg, l', which has not yet been processed, by the induction hypothesis. Consequently, $t_{sim} = t_{l'}$, the original TTO for this leg. Since $t_{l'} \geq t_{l''} + \widetilde{d}_{l'}$, the previous leg must already have been processed and its TTO is unchanged. Consequently, in `updateFrozenLeg()` the condition $t_{earliest} > t_{sim}$ will fail and no new event will be generated. If the resource is free at this time, $t_{l'}$, a new block will be created for resource k which matches a block in the original solution. If the resource is not

free at this time, that would imply this block overlaps one of the blocks in $\mathcal{B}$. By the induction hypothesis this would contradict the assumption that the original solution was resource-feasible. Consequently, the resource must be free at time $t_{l'}$ and the induction hypothesis holds for $n + 1$. □

As discussed, once a leg l' is completed, its TTO, t_{fj}, channel, c_{fj}, channel separation time, δ_{fj}, and reservation block $b(n_j, c_{fj}, f) = \left[t_{fj}, t_{fj} + \delta_{fj}\right]$ become immutable and its previous TTO, ${}_jt_f$, becomes fixed at $t_{f,j-1}$. We propose that the output of `simulateFSFS()` is resource feasible, otherwise, there will be an immutable infeasibility in the solution.

Proposition 6.3.2. *The output of `simulateFSFS()` is resource feasible.*

Proof. The set Q_M is initialized to consist of all non-frozen legs l', whose previous TTO is not visible, and all frozen blocks b sorted in increasing order of time. Here, time is denoted by t_{next} for the start time of blocks, and $t_{l'}$ for legs. The set Q_n is the set of LEGREADY pending events for node n, sorted in increasing order of scheduled times, R0TTO. This set is initialized to $\emptyset$. All resources k will have the blocked status u^k initialized to FALSE, that is, resource k is available. The proof is by induction on the number of iterations the loop **while** $Q_M \neq \emptyset$ has been run. The induction hypothesis, is that for every event that was removed up to and including iteration n, no overlapping resource block was generated. The hypothesis is trivially true for $n = 1$, because no resource block was previously generated. Assuming the hypothesis is true for iteration n, we consider the next iteration. The next event to be processed could be one of four events: `BLOCKSTART`, `BLOCKEND`, `LEGREADY`, and `RESERVATIONEND`. Excluding the `BLOCKSTART` event, which does no TTO assignment, the other events make a call to the function `processPENDING()`. For each flight leg l' in the pending queue Q_n, `processPENDING()` then calls `getBestChannel()`. For waypoints, `getBestChannel()` checks if the waypoint is available, and for airports, `getBestChannel()` checks if a dedicated channel is available, otherwise a common channel. Availability is defined by the function `isAvailable()`, where a resource must have its blocked status $u^k = FALSE$, and either possesses no frozen block b^k in the future, or have sufficient separation between the start time of the next frozen block t_{next} and the current flight leg time $t_{l'}$, $t_{l'} + \delta \leq t_{next}$. Only if these criteria are fulfilled, implying no overlap with a block that was scheduled prior to iteration n, will `assignLegToResource()` be called. `assignLegToResource()` then blocks the current resource by setting $u^k = TRUE$ and adds a RESERVATIONEND event to the queue Q_M, at $t = t_{sim} + \delta$. It follows that resource k will not be assigned any flight leg for $t < t_{sim} + \delta$, hence no reservation block in the future will overlap with the current reservation block.

The function `processPENDING()` considers, for node n, up to three separate channels c, and therefore can result in a maximum of three assignments to different resources k. As soon as an assignment is made, that resource becomes blocked $u^k = TRUE$, so each assignment will be made to a unique resource. A reservation block $b(n_j, c_{fj}, f)$ is unique to a single resource k, belonging to channel n and node c. As the assignments are made on unique resources, they are independent of each other and can be considered separately. Suppose one of the assignment is for resource k, and leg l'. If resource k is available, this is due to either having no reservation block in the future, or sufficient separation time $t_{l'} + \delta \leq t_{next}$ with the start time of the next frozen block, and an unblocked resource $u^k = FALSE$. Hence, following the discussion in the previous paragraph, the inner functions of `processPENDING()` will assign a resource feasible reservation block for flight leg l' and set $u^k = TRUE$, blocking any infeasible assignments in later iterations. If no resource is available, the condition not(c_{best} ==NOAVAILABLECHANNEL) fails and no resource block is generated for resource k. Consequently, the resource k will either be assigned a resource feasible reservation block, or no resource block is generated, and the induction hypothesis holds for $n + 1$. □

Corollary 6.3.3. *The function `reconcileFrozenLegs()` does not alter the TTO of completed legs.*

Proof. The completed legs in the output of `simulateFSFS()` are resource feasible due to Proposition 6.3.2. Furthermore, since $t_{now} > t_{l'}$, for any completed leg l', leg l' has previously been processed, is resource feasible and is immutable. Hence, combining this result with Lemma 6.3.1, we obtain the result. □

It is imperative for the algorithm to schedule all flight legs. We must ensure that `PROCESSPENDING()` does not get trapped in an infinite loop where no channel is ever available for a particular flight leg, that is, c_{best} ==NOAVAILABLECHANNEL. We prove that all flight legs will eventually be assigned a TTO in Proposition 6.3.4.

Proposition 6.3.4. *All flight legs will be scheduled at the end of the DES.*

Proof. The key algorithm that will lead to an assignment of flight legs is `processPENDING()`. This algorithm is only called during either of the three algorithms: `processBLOCKEND()`, `processLEGREADY()`, or `processRESERVATIONEND()`. We first cover the trivial case where the flight leg is the first to be assigned onto an empty resource. Here, a LEGREADY event would exist in the queue, and when the LEGREADY event is processed, the algorithm calls `processPENDING()` through `processLEGREADY()`. Because no prior flight leg has been assigned to this resource, we have that the resource is not blocked ($u^k == FALSE$).

Thus `isAvailable()` will return $TRUE$ and the flight leg would be assigned a TTO at this resource.

For the general case where some flights have already been assigned a resource, we consider two cases, where we focus on the assignment of a TTO t_{fj} for flight f on flight leg j, at resource k. We assume without a loss of generality that this is the only event in the queue of LEGREADY events Q_n. The first case is when the there are no frozen flight legs ahead of t_{fj} at resource k. Both functions `processBLOCKEND()` and `processRESERVATIONEND()` pertaining to a feasible resource for the flight leg would have have that the resource is not blocked ($u^k == FALSE$), `isAvailable()` returns $TRUE$, and the flight leg would be assigned a TTO at resource k. `processLEGREADY()` may also have scheduled the flight leg if the resource is available, but this condition is unnecessary for our proposition as either `processBLOCKEND()` or `processRESERVATIONEND()` is guaranteed to be called prior to the termination of the algorithm. We observe this is true as blocking a resource will generate the BLOCKEND and RESERVATIONEND events, thus the resource will be unblocked by the `processBLOCKEND()` and `processRESERVATIONEND()` functions prior to the termination of the algorithm.

The second case is when there is a frozen flight leg ahead at resource k, with insufficient separation to schedule flight leg j for flight f. When this happens, the flight leg is unable to use resource k at this time, and the flight leg will not be scheduled. For any frozen blocks, a BLOCKEND event corresponding to the frozen flight leg exists, and in particular, this BLOCKEND corresponding to the flight leg ahead exists. Assume we have processed any other events up until this BLOCKEND event. If there is a sufficient separation from the current event time to the next frozen block, we can assign a TTO to flight leg j for flight f at the current event time of BLOCKEND. However, if there again is insufficient separation to the next frozen flight leg at this resource, all events are processed, including the BLOCKEND events at this resource, with the edge case of repeating until there are no more frozen flight legs at this resource. Given that we have a finite number of flights, and by extension a finite number of frozen flight legs, flight leg j for flight f is guaranteed a TTO assignment. Hence, applying the logic for all flight legs, no unscheduled flight leg will exist at termination of the DES. □

In the next chapter, we compare the results of the DES to the AFR/WFR model, reason which is the better choice for our simulator, and perform further analysis on the various information sharing and collaboration regimes.

6.4 Summary of Notation

TABLE 6.1: Notation for the DES

t_{now}	Current simulation time
H	Planning horizon
$\mathcal{F}$	Set of all flights
$\mathcal{F}_A$	Set of active flights that are visible to air traffic planners in the region
$\mathcal{F}_C$	Set of completed flights
$\mathcal{N}$	Set of all nodes; consists of both waypoints and airports
$\mathcal{A}$	Set of all arcs
$\mathcal{K}$	Set of all node-channel resources in the network
$\mathcal{Y}$	Set of all event types
f	Flight $f \in \mathcal{F}$
n	Node $n \in \mathcal{N}$
k	Resource $k \in \mathcal{K}$
y	Event type $y \in \mathcal{Y}$
P_f	Acyclic flight path, consisting of an ordered set of nodes associated with flight f: $(n_1, n_2, ..., n_{\|P_f\|})$
$\underline{d}_{fj}$	Minimum duration for flight f to transit from node n_j to node n_{j+1} in path P_f, excluding hold times
$\overline{d}_{fj}$	Maximum duration for flight f to transit from node n_j to node n_{j+1} in path P_f, excluding hold times
$\widetilde{d}_{fj}$	Preferred duration for flight f to transit from node n_j to node n_{j+1} in path P_f, excluding hold times
t_{fj}	Target Time Over (TTO) at the j^{th} node in path P_f for flight f
$\widetilde{t}_{fj}$	Scheduled TTO at each node for flight f at node n_j in path P_f
${}_jt_f$	TTO of the node previous to n_j for flight f. in path P_f
$r(n_j)$	FIR responsible for flights over node n_j
h_{fj}	Holding time for flight f at node n_j in path P_f
c_{fj}	Channel assignment for leg j of flight f
δ_{fj}	Minimum required separation at channel c_{fj}
$b(n, c, f)$	Block of time reserved at for flight f at node n and channel c
u^k	Boolean indicating if resource k is blocked, and if TRUE, the time which it is blocked until
Q_M	Master event queue
Q_n	Event queue at node n

Chapter 7

Value of Information Sharing and Collaboration

The previous chapters have laid the foundation for our Air Traffic Flow Management (ATFM) model, and explored, in detail, the implementation of multiple different algorithms to effectively schedule conflict-free flight plans. This chapter serves to explore the value of information sharing and collaboration under the FF-ICE R1 regime.

We first discuss and compare the results of the segregated Airport Flow Regulator (AFR) and Waypoint Flow Regulator (WFR) algorithms against the Discrete Event Simulation (DES). Next, we give our reasons for proceeding with further experiments using the DES instead of the AFR and WFR algorithms. We then dive into the difference in airborne savings for mixed mode information sharing, for both full and incremental participation of the FF-ICE R1 program. Finally, we uncover the source of these airborne savings, be it by time of the day, number of participating flights, particular airport pairs, or the broader impact on non-participating flights due to the FF-ICE R1 program.

7.1 Comparison of Algorithms

From Table 7.1, we immediately notice that both the GDA and SA algorithms generate an abnormally large amount of delay, both arrival and on ground, as compared to the DES. The GDA also performed worse than the SA, similar to the results obtained in Section 5.5. While the GDA and SA perform well on short time horizons spanning a few hours, their limitations become apparent when extended to simulate an entire day of flight plans. There are two possible reasons for this. The first is what we term *drifting*, which we define as a leg entering a vicious cycle of delays. In the GDA and SA model, both conflict resolution and deviation from preferred time enter the objective function, with conflict resolution having a significantly higher penalty coefficient. This causes conflict resolution to be prioritized, and over hundreds of runs and information updates across FIRs and time steps, the TTOs in the solution begin to drift. Under the

GDA and SA algorithms, conflict resolution tends to occur by incrementally increasing all TTOs rather than decreasing them, leading to progressively greater delays as the simulation progresses. This is contrasted by the DES, where flights are scheduled as early as possible, with little to no gaps between flights, except for the minimum required separation time.

The second reason stems from the relative lack of flight resequencing. In the WFR, a flight's departure TTO is strictly constrained to be no earlier than the TTO assigned during the preceding AFR step. Once the AFR has been executed, its departure times are enforced in the WFR, meaning that a flight can be further delayed but not advanced, even if a time gap opens up due to another flight's delay. This restriction prevents the system from utilizing newly available capacity, resulting in inefficiencies and unnecessary delay propagation. Moreover, because resequencing is uncommon under both the GDA and SA heuristics, conflicts tend to propagate along the flight path, increasing the TTOs of the flight legs. These shifts then delay other flights due to the preserved sequence, again failing to exploit the space vacated by delayed flight legs. This cascading effect can lead to a disproportionate increase in both airborne and ground delay. In contrast, such behavior is mitigated in the DES, where gaps left by delayed flights will be filled by other flights under the FCFS or FSFS queuing system.

In addition to the reasons discussed above, the DES is more clearly aligned with our vision for the FF-ICE simulator. We envision the simulator as a tool that can accurately capture how countries might benefit from enhanced information sharing and collaborative decision-making. The DES framework makes these benefits clearer, more interpretable, and more representative of real-world operations. Unlike optimization-based approaches, DES mirrors current ATC practices, where decisions under regular conditions are guided by simple rules, such as FCFS and FSFS (Erkan, Erkip, and Şafak, 2019; Ma, Delahaye, and M. Liang, 2024). As such, it provides a more realistic and operationally grounded platform for simulating FF-ICE scenarios.

TABLE 7.1: Comparison Table For All at R0 Regime for 1 Oct 2023 Flight Plans

Algorithm	Avg. Flight Time (min.)	Avg. Arrival Delay (min.)	Avg. Ground Delay (min.)
DES	147.02	5.74	2.68
GDA	191.43	60.95	13.44
DES	147.50	5.74	2.69
SA	172.35	34.01	6.07
GDA	191.87	61.16	13.48
SA	172.26	34.15	6.08

7.2 Computational Results

To reiterate from Section 3.6, t the dataset used for our experimental runs are published flight plans, obtained from OAG, for flights arriving or departing from the ASEAN plus region between 1 Oct 2023 and 7 Oct 2023. The necessary data required to build the airspace network include FIR boundaries, waypoint and airport nodes, and flight paths connecting all possible origin and destination airports pairs. The FIR boundaries were derived from ICAO geographical data, and modeled as a series of polygons. The nodes and connecting arcs used in our simulations are obtained by processing Aeronautical Information Publication (AIP) documents published by each country's civil aviation authority, and are presented in Figure 5.1. Using the nodes, and the origin-destination airports pairs, we applied Dijkstra's shortest path algorithm to generate flight paths for all aircraft. The paths in our dataset are plotted in Figure 5.2.

For the results below, we reiterate that a flight is completed from the perspective of the FIR of interest and will be archived as soon as condition 6.1 is satisfied. We model a look-ahead period of two hours, such that all flights with at least one active or frozen leg during the next two hours will be added into consideration for the discrete event simulation algorithm, and we set the time shift for the sliding window at five minutes, that is, we reschedule flights for the next two hours, at every five-minute interval.

The base airspace capacity is determined quantitatively by collecting information for the average number of flight traversals in each FIR, similar to the method employed in (Tan, 2021). The reduced airspace capacity is computed by synthetically reducing the original airspace capacity at arrival and departure nodes to 0.80 of their original levels, rounded up to an integer value. These capacity values are then used to compute the required separation time between aircraft by dividing the number of seconds in an hour by the number of flights in an hour.

We introduce the columns and their respective definitions that are common to many of the result tables in this section:

- **Day**: Selected day for the current simulation;
- **Capacity**: Either the base or reduced airspace capacity scenario;
- **Regime**: The information regime that specifies which FIRs under consideration operate at FF-ICE release level R0 or R1;
- **Collaboration**: The collaboration protocol, that further refines the regime by selecting a subset of flights that are currently at FF-ICE R1, and set all other flights to R0. Example of possible protocols are allowing collaboration only between particular airport pairs, or airlines;

- **No. of Matching Flights**: The number of matching flights between the two simulations currently under consideration;
- **Avg. Flight Time**: The average airborne flight time in minutes, for all matching flights in the selected simulation;
- **Total Flight Time Difference**: The difference between airborne flight time in minutes, summed over all matching flights in the selected simulation pair;
- **Avg. Arrival Delay**: The average delay in minutes, at the destination airport for all matching flights in the selected simulation. An arrival delay is calculated by subtracting the scheduled arrival time (R0TTO) from the actual arrival time at the destination airport (TTO);
- **Avg. Ground Delay**: The average ground delay time in minutes, for all matching flights in the selected simulation;
- **Avg. Fuel**: The average fuel consumed in kg, for all matching flights in the selected simulation. Computations are based on the BADA fuel formulae (EUROCONTROL, 2019).

7.2.1 Smoothing Parameter and Capacity Scenarios

A critical source of benefit from information sharing stems from the congestion that occurs at airports. If all the airports in the region are only lightly congested, as was seen during the COVID-19 pandemic, we are unlikely to see much benefit from information sharing. The underlying reason for why benefits are proportional to congestion, is that airborne savings are primarily derived from GDPs. A flight is assigned a GDP when congestion is predicted at the arrival airport, causing it to depart later and arrive when a landing slot becomes available. Therefore, flights benefit through substituting airborne delays with the more resource-efficient ground delays. As higher levels of airport congestion lead to increased expected delays, more flights will be assigned GDPs, thereby creating greater potential for airborne time savings. Accordingly, as air traffic returns to pre-pandemic levels and continues to grow, we expect the benefits and importance of information sharing to continue on an upward trajectory.

To illustrate the effect of congestion we consider two capacity scenarios, a base capacity case and a reduced capacity case. In the base case, capacity envelopes are determined based on empirical estimates of current activity within the regional airports. In the reduced capacity case, the envelopes are shrunk by 20%. Figures 7.1 and Figure 7.2 show the simulated airport activity at six major airports in the region for the two different capacity levels, together with the capacity envelopes enforced by the simulation. The bubbles represent relative frequency counts of simulated arrival-departure

pairs over 15-minute intervals for an entire day of simulated flights, 1 Oct 2023. In the plots, the radius of the bubbles reflects the number of 15-minute intervals that contain the particular combination of arrivals or departures. Close inspection of the envelopes will reveal that the reduced capacity envelopes are approximately 80% of the base case versions. For example, maximum departures at Singapore Changi airport (WSSS) are 60 per hour in the base capacity case, and 48 per hour in the reduced capacity case, a reduction of 20%. Similarly, maximum arrivals at Jakarta (WIII) are 94 per hour in the base case and 76 in the reduced capacity case, a reduction of 19.7%. We observe that in the base capacity case, operations in the 15 minute time interval are concentrated around the middle of the chart, indicating that airports are operating below the maximum capacity for most of the day. In contrast, the reduced capacity case capture a large proportion of bubbles near the borders of capacity envelope, indicating severe congestion at airports, where flights are scheduled one after another, most likely experiencing delays to precisely satisfy the minimum required separation. In both figures, the bubbles indicate that the simulated flight activity, which includes ground delays and airborne holding, obeys the capacity envelope at all airports with few exceptions. Similar findings across all other nodes reinforce the validity of our model, a three channel system with separation times, as an effective representation of airport capacity envelopes and flow regulation mechanisms.

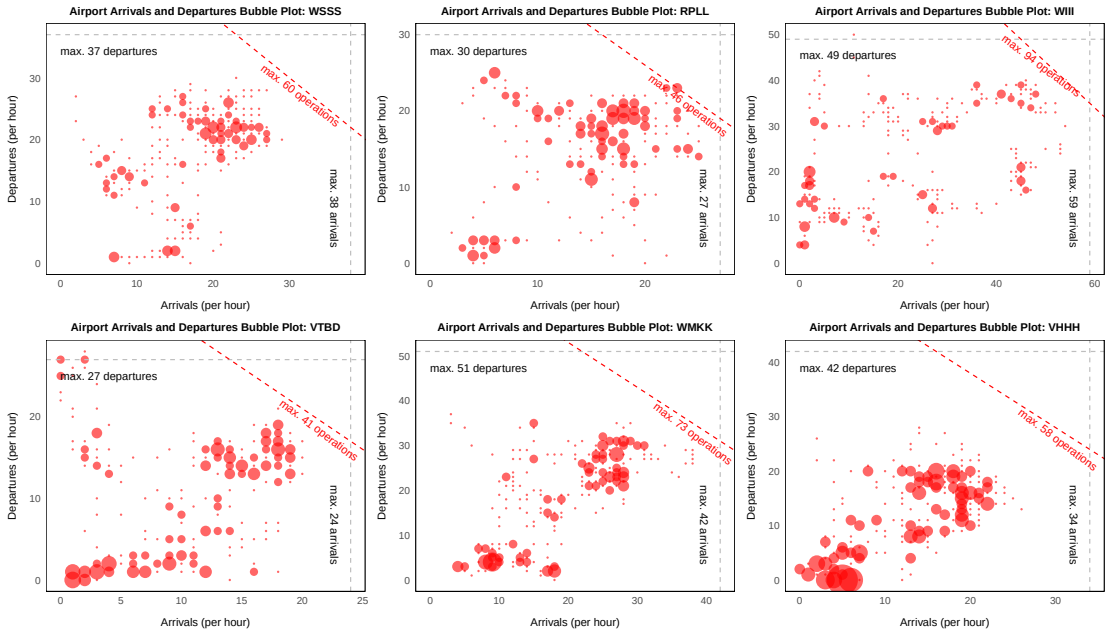

FIGURE 7.1: Frequency Counts and Capacity Envelopes Under the Base Capacity Case for Six Major Airports: Singapore (WSSS), Manila (RPLL), Jakarta (WIII), Bangkok (VTBD), Kuala Lumpur (WMKK), and Hong Kong (VHHH). Frequencies are Based on Arrival-Departure Counts in 15-min. Intervals for 1 Oct 2023

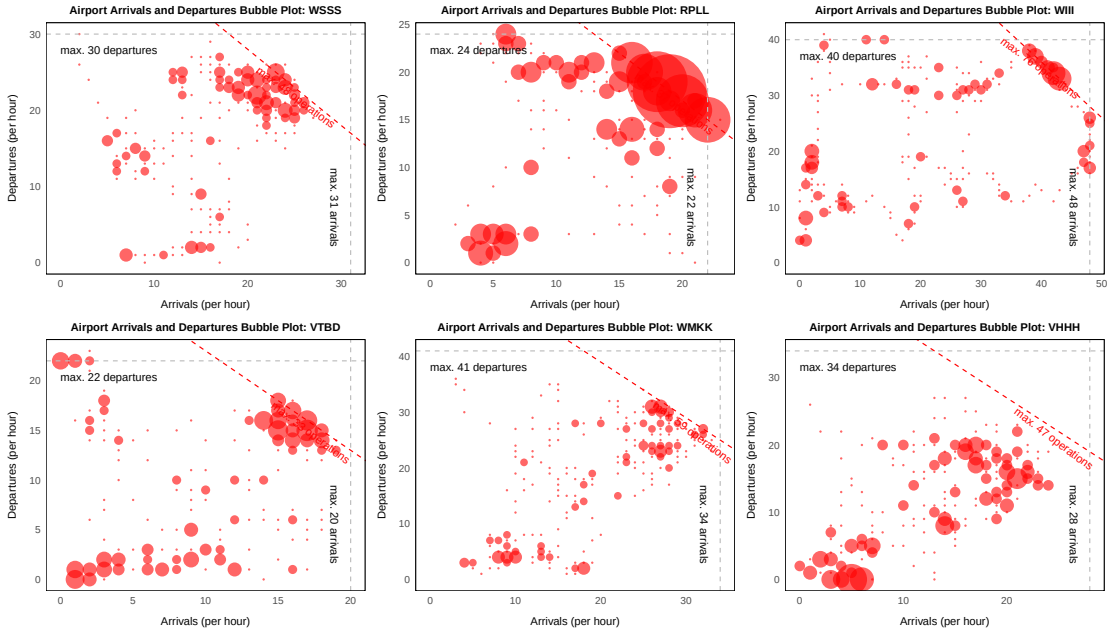

FIGURE 7.2: Frequency Counts and Capacity Envelopes Under the Reduced Capacity Case for Six Major Airports: Singapore (WSSS), Manila (RPLL), Jakarta (WIII), Bangkok (VTBD), Kuala Lumpur (WMKK), and Hong Kong (VHHH). Frequencies are Based on Arrival-Departure Counts in 15-min. Intervals for 1 Oct 2023

We recall from Section 3.2, that the implementation of ground delay programs included the use of a mixing parameter, PROFILEMIX. If a flight is not an R1-flight, that is, if one or both of origin and destination airports are at release level R0, then no adjustment is made to departure time based on a shared CTOT. If the flight is an R1-flight, then we have the origin airport adjust the planned departure time in the direction of the most recent computed CTOT. The PROFILEMIX parameter controls how reactive the origin airport is in responding to changes in CTOT. Our conjecture was that the optimal PROFILEMIX parameter would be less than 100% based on machine learning experience in other domains, such as in the GDA algorithm, where we shift the TTO by a step size in the direction of the gradient, or in exponential smoothing to remove noise and bias in our forecasts (Gardner, 1985). Surprisingly, our computational results do not support that conjecture. Figures 7.3 and 7.4 show the results of varying the PROFILEMIX parameter for both the base and reduced capacity case (comparing information regimes All R0 with All R1). Under both capacity scenarios, the savings in airborne time continue to increase as the mix parameter is increased, with slight variation in the reduced capacity case. Consequently, we elected to run all further experiments setting the PROFILEMIX parameter at 1. That is, the origin airport plans to change departure times 100% to newly calculated CTOTs transmitted to them. These

preliminary results also bring to light the impact of congestion on the value of information sharing and collaboration. Based on the PROFILEMIX parameter with a value of 1, The maximum airborne delay savings jump from 4874 minutes to 14864 minutes, an increase of 205%, for a 20% reduction in capacity. This finding suggests that the value of information sharing and collaboration is strongly correlated with the congestion levels, stressing the need for information sharing and collaborative measures as air travel demand continues to outpace the increase of airspace capacity. We dive into greater detail on the value of information sharing and collaboration under FF-ICE R0 and FF-ICE R1 in the next section.

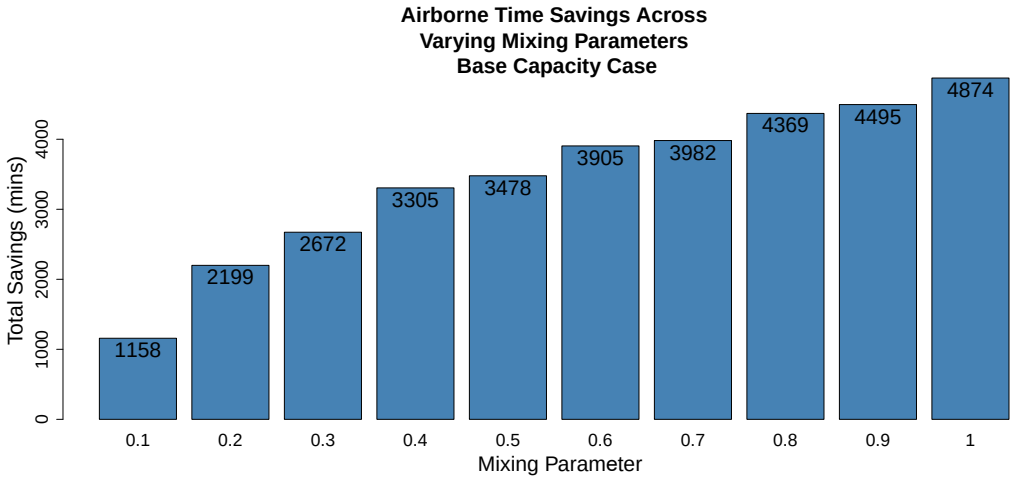

FIGURE 7.3: Total Airborne Savings Under FF-ICE Plotted Against PROFILEMIX, Base Capacity Case. Results for the Simulation Using 1 Oct 2023 Flight Plans.

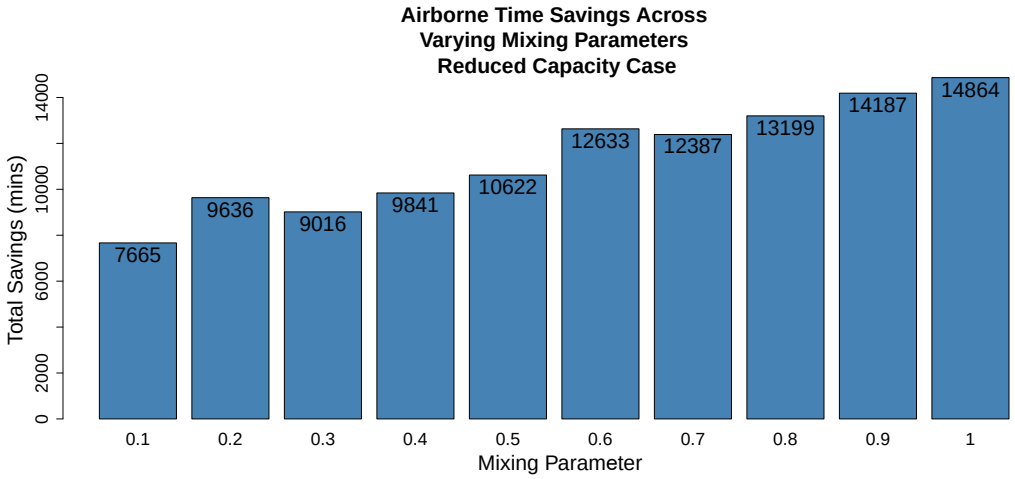

FIGURE 7.4: Total Airborne Savings Under FF-ICE Plotted Against PROFILEMIX, Reduced Capacity Case. Results for the Simulation Using 1 Oct 2023 Flight Plans.

7.2.2 Results of the FF-ICE R1 Concept Over the ASEAN Plus Region

In this section, we analyze and compare the performance of flight trajectories of the current operational capabilities (FF-ICE R0), against the full collaboration and information sharing regime (FF-ICE R1). The analysis focuses on key performance metrics, including the average flight time, ground delay, arrival delay, and fuel consumption. The results obtained are based on simulation runs conducted over the ASEAN Plus region for a period of seven consecutive days, from 1 October 2023 to 7 October 2023, under both the base and reduced capacity scenarios. The raw results are presented in Table 7.2. We first discuss the results and their interpretation by column, and later summarize these results and identify key trends graphically.

Comparing the columns *Avg. Flight Time (R0)* and *Avg. Flight Time (R1)*, we see that in all instances, the average flight time under the FF-ICE R1 regime is lower across all rows of the data. This indicates that across a whole day of simulated flights, collaborative measures and information sharing are effective in reducing the time spent in the air. This is also evident when we compare the results in the *Avg. Fuel* columns, where the fuel consumed under the FF-ICE R1 regime is consistently lower than the FF-ICE R0 regime across all simulations.

Looking at the arrival delays, we see that the both the arrival delays and ground delays are greater under FF-ICE R1 than FF-ICE R0. The ground delays are necessarily higher under FF-ICE R1 due to the GDPs assigned to flights which are expected to encounter delays enroute, or at the destination airport. Due to the GDPs however, new conflicts may be introduced, leading to the flights assigned a GDP not being able to reach the destination airport at its new reserved time slot, and hence were assigned a later arrival time slot than what was expected under the R0 case. As such, our preliminary conjecture is that the new conflicts introduced due to GDPs, are the primary cause of increased arrival delays at the destination airport under the FF-ICE R1 regime. Also, note that the flight time savings is equal to the difference between the increase in the arrival and increase in ground delay. This highlights the idea that flight savings stem from substituting airborne holding with ground holding. For example, looking at the second row of Table 7.2, we derive the following values:

- Increase in average ground delay $= 10.67 - 7.88 = 2.79$ mins;
- Increase in average arrival delay $= 19.62 - 18.49 = 1.13$ mins;
- Difference in the increase of average ground and arrival delay $= 2.79 - 1.13 = 1.66$ mins.

Notice that the difference in the increase of average ground and arrival delay matches the savings in average flight time of $154.90 - 153.24 = 1.66$ mins exactly.

For this study, we focus primarily on the difference in average flight time between the FF-ICE R0 and R1 regimes, as this value represents the key benefit of information sharing and collaborative decision-making, which leads to the greatest operational cost savings. While important in practice, average arrival delays are less relevant in this analysis as we chose to utilize a FCFS and FSFS scheduling system, which is more prevalent in real-world operations (Erkan, Erkip, and Şafak, 2019; Ma, Delahaye, and M. Liang, 2024). Our rule-based algorithm, does not have the ability to speed up flights to meet the scheduled arrival time. Hence, if we were to set the deviation from the scheduled arrival time as an objective, it would be more appropriate to apply optimization algorithms that can both slow down and speed up flights. While such algorithms will be considered in future work, it is beyond the scope of this paper. Furthermore, ground delays are not an appropriate measure of benefits as passengers are less concerned about leaving late as long as they arrive at their destination on time. We do not go into detail on fuel analysis as this is also beyond the scope of the paper, and requires significantly more material to cover the nuances of the different levels of fuel consumption based on the phase of flight, velocity, altitude etc. While our analysis is centered around the airborne savings realized, we still report the other values when appropriate to provide context on how the other values change, relative to the airborne savings achieved.

Using the results from Table 7.2, we plot the total airborne and fuel savings in Figure 7.5, and the total increase in ground and arrival delay in Figure 7.7. We first discuss the results in Figure 7.5. We see that in both plots, and both capacity cases, the graphs all lie above the x-axis, representing a consistent airborne and fuel savings across all days and capacity scenarios, again highlighting that the information sharing and GDPs under FF-ICE R1 are effective policies at optimizing the airspace network. Next, we see that under FF-ICE R1, the reduced capacity case has a wider range of values across the week of data, and approximately three times the savings as compared to the base case. This aligns with our earlier conjecture that the savings would be proportional to the congestion, as more congestion leads to more GDPs being assigned, leading to an increase of savings. Additionally, since more GDPs are triggered, an increase in the variance was also observed, with the increase proportionate to the increase in absolute values. From the figure, we see that the increase in the range and interquartile range between the base and reduced capacity case, is roughly three times, on par with the increase in total savings under FF-ICE R1. Additionally, the graphs for total airborne savings and total fuel savings are nearly identical, as expected, since no specific provisions were made to minimize fuel consumption, and fuel usage is assumed to be directly proportional to airborne time.

TABLE 7.2: Comparison Table For All at R0 Regime Vs All at R1 Regime

Day	Capacity	No. of Matching Flights	Avg. Flight Time (R0)	Avg. Flight Time (R1)	Avg. Arrival Delay (R0)	Avg. Arrival Delay (R1)	Avg. Ground Delay (R0)	Avg. Ground Delay (R1)	Avg. Fuel (R0)	Avg. Fuel (R1)
1 Oct	Base Case	8793	147.07	146.54	5.73	6.02	2.69	3.50	8863.10	8845.61
1 Oct	Reduced Case	8754	154.90	153.24	18.49	19.62	7.88	10.67	9175.12	9119.00
2 Oct	Base Case	8547	152.74	152.21	4.29	4.44	2.14	2.83	9636.69	9618.34
2 Oct	Reduced Case	8509	158.94	157.54	14.52	15.49	6.43	8.80	9885.90	9838.28
3 Oct	Base Case	8471	153.37	152.88	4.63	4.86	2.21	2.94	9680.97	9663.51
3 Oct	Reduced Case	8438	159.91	158.54	15.15	15.93	6.47	8.62	9945.82	9898.91
4 Oct	Base Case	8644	153.34	152.76	5.86	6.07	2.71	3.50	9633.08	9612.10
4 Oct	Reduced Case	8609	160.82	159.23	18.08	18.94	7.64	10.08	9927.21	9872.31
5 Oct	Base Case	8697	154.75	154.25	5.68	5.93	2.48	3.23	9740.68	9722.71
5 Oct	Reduced Case	8664	162.89	161.18	18.68	19.47	7.62	10.12	10070.25	10010.70
6 Oct	Base Case	8896	152.90	152.30	6.63	6.77	2.93	3.67	9540.95	9519.81
6 Oct	Reduced Case	8860	160.60	159.00	19.12	19.99	8.03	10.50	9850.79	9795.89
7 Oct	Base Case	8715	154.53	154.02	5.18	5.39	2.37	3.10	9747.83	9730.45
7 Oct	Reduced Case	8671	162.20	160.52	17.51	18.32	7.31	9.80	10051.89	9994.18

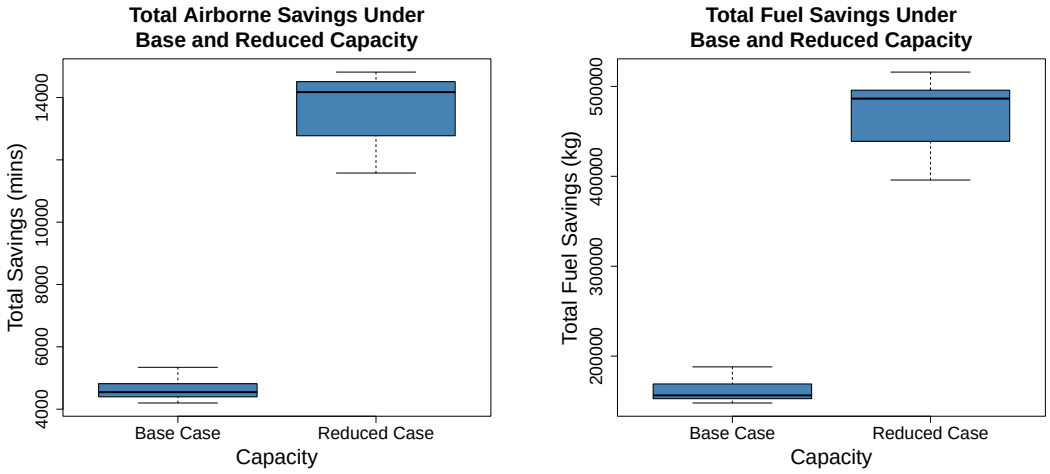

FIGURE 7.5: Total Airborne and Fuel Savings Under R1, Split by Base and Reduced Capacity Scenario

We also provide the histogram of the flight time savings for the seven days of simulated flights in Figure 7.6. For this analysis, we consider only flights that had airborne savings or losses of over 5 minutes. The number of flights under the base and reduced capacity scenarios for the 7 simulated days are 60763 and 60505 respectively. Although the average flight savings under the base and reduced capacity case in Table 7.2 are approximately 0.5 and 1.5 minutes respectively, Figure 7.6 highlights that there are a significant number of flights that are able to save more than 5 minutes of flight time. In particular, the mean preferred flight time for the flights that experienced airborne savings of over 5 minutes under the base and reduced case is 90.34 and 93.87 respectively. This number is significantly lower than the preferred flight time in Table 7.2, given that the long distance flights outside the ASEAN Plus region would not have participated in the FF-ICE R1 regime. Using the formula

$$\text{Mean \% Deviation } = \frac{100}{n}\sum_{i=1}^{n}\frac{|savings_i - preferred\_time_i|}{preferred\_time_i},$$

we find the percentage savings for the flights with more than 5 minutes of airborne savings for the base and reduced capacity to be 12.3% and 17.4% respectively, indicating the substantial benefits that can be achieved through collaboration.

Additionally, we see in Figure 7.6 that the number of flights experiencing savings of more than 5 minutes is significantly larger under the reduced capacity case, supporting our hypothesis that a greater mismatch in demand and capacity would lead to greater savings under the FF-ICE R1 regime. The results also demonstrate that under the base capacity case, the number of flights with significant savings sharply decline after the 10 minute mark. In contrast, under the reduced we see that the savings taper off more

slowly, indicating that there is a lower variation in the distribution of flight savings. We verify this observation by computing the coefficient of variation $CV = \frac{\mu}{\sigma}$, as 4.04 and 3.80 for the base and reduced capacity case respectively.

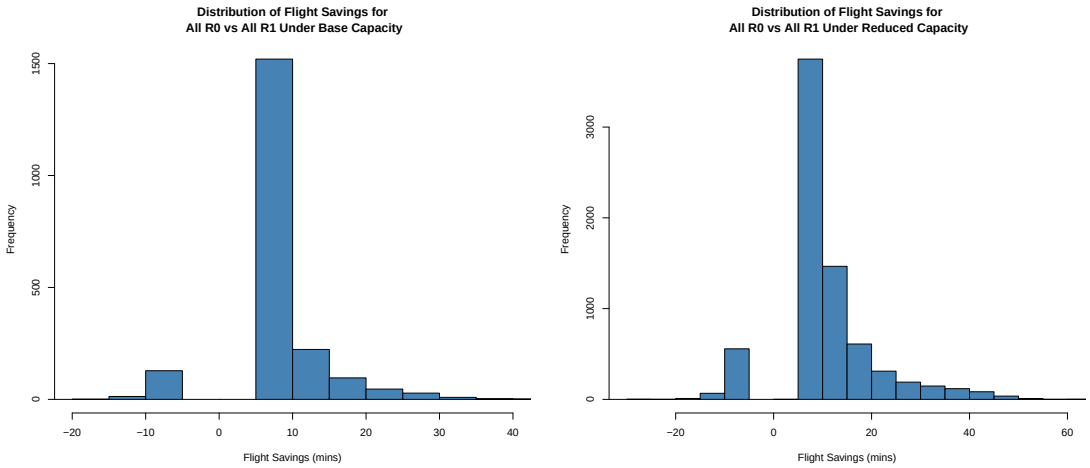

FIGURE 7.6: Distribution of Airborne Savings Under R1, Split by Base and Reduced Capacity Scenario, Only Flights With Deviations of More Than 5 Minutes

Next, in Figure 7.7, only the total increase in ground and arrival delays are considered, giving a high level overview of the increase in frequency and magnitude of the delays. Here, the difference in the ground delay and arrival delay corresponds to airborne savings, which is visually represented in the figure as the difference between the two plots (either between the first and second boxplots, or between the third and fourth boxplots). This observation aligns our earlier analysis, where the height difference in the reduced case is approximately three times greater than in the base case. Similarly, the variance appears proportional to the absolute values, deduced from the proportionality of the interquartile range of the box plots to their value on the y-axis. A significant increase in ground delays under both the base (leftmost box plot) and reduced capacity (third box plot) scenarios suggests that a greater mismatch between capacity and demand results in a proportional increase in GDP requests.

While the results indicate that increased implementation of GDPs contributes to greater airborne and fuel savings, a consequential trade-off that cannot be overlooked is the associated rise in workload for ATC officers. Given that a significant portion of ATC workload is still handled manually, advancing the automation of ATC tasks should be considered a parallel priority. Another side effect of information sharing and collaborative GDPs are the increase in arrival delays. Under both the base and reduced capacity case, we observe a positive increase in arrival delays, represented by the boxplots lying above the x-axis. This underscores the challenge of initiating GDPs,

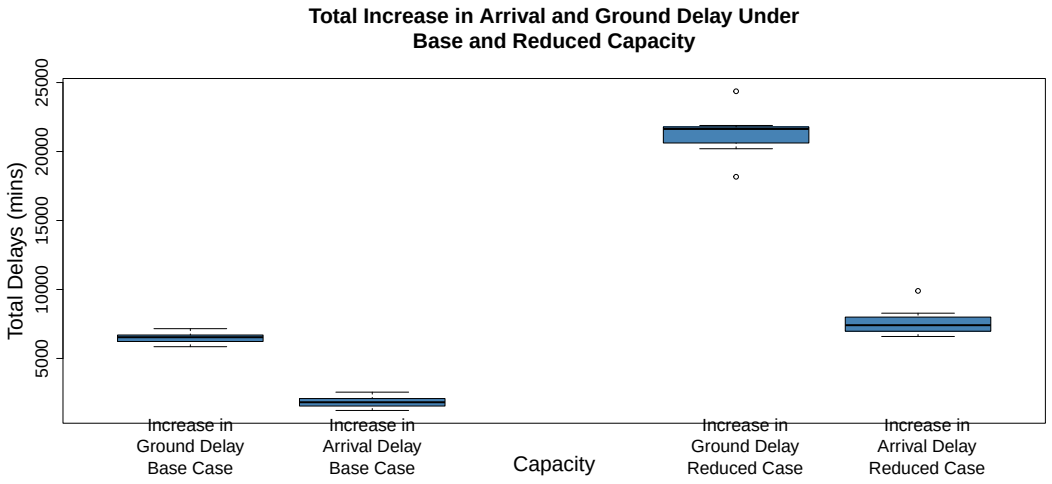

FIGURE 7.7: Total Increase in Ground Delay and Airborne Delay Under R1, Split by Base and Reduced Capacity Scenario

reserving future capacity, and allowing flights to fly at their preferred speed from the origin to destination airport. The problem is made even more complex under a rolling horizon update framework which models the uncertainty and decentralized nature of real-world airspace operations in the ASEAN Plus region. In contrast, we would expect lower arrival delays for algorithms that optimize a full day's schedule in a single run.

We now discuss how the GDPs lead to airborne savings. Figure 7.8 depicts a particular flight, which under R1 experienced a reduction in airborne time. In the figure, the turquoise line represents the TTO for the R1 case, and the red line represents the TTO for the R0 case. The blue dashed line represents the flight trajectory under the perfect case with no delays, where the flight traverses its entire flight path at its preferred speed. We notice that in both the R0 and R1 simulation, the turquoise and red lines are parallel to the blue dashed line for most of the flight, indicating the flight is traveling at its preferred speed for the entire flight path, except at the last node "WIII", where it experienced a delay of approximately two minutes under R0. We also see that on the first leg, labeled "WSSS", the turquoise line lies above the red line, indicating that a ground delay of approximately two minutes was assigned to the flight under the FF-ICE R1 regime, mirroring the arrival delay under R0. Note that the red line above the blue dotted line in the figure indicates a ground delay due to congestion at the departure airport, and not a ground delay due to collaborative decision-making. In this example, the flight under R1, was able to maintain its preferred speed throughout the entire flight path, avoiding a delay at the destination airport "WIII", precisely because it was delayed at the origin airport, and landed within its reserved slot at the destination airport, reducing its airborne delay by approximately two minutes. Note that the vertical jump in the zoomed portion of Figure 7.8 is understood as the holding delay at

the final approach node under R0.

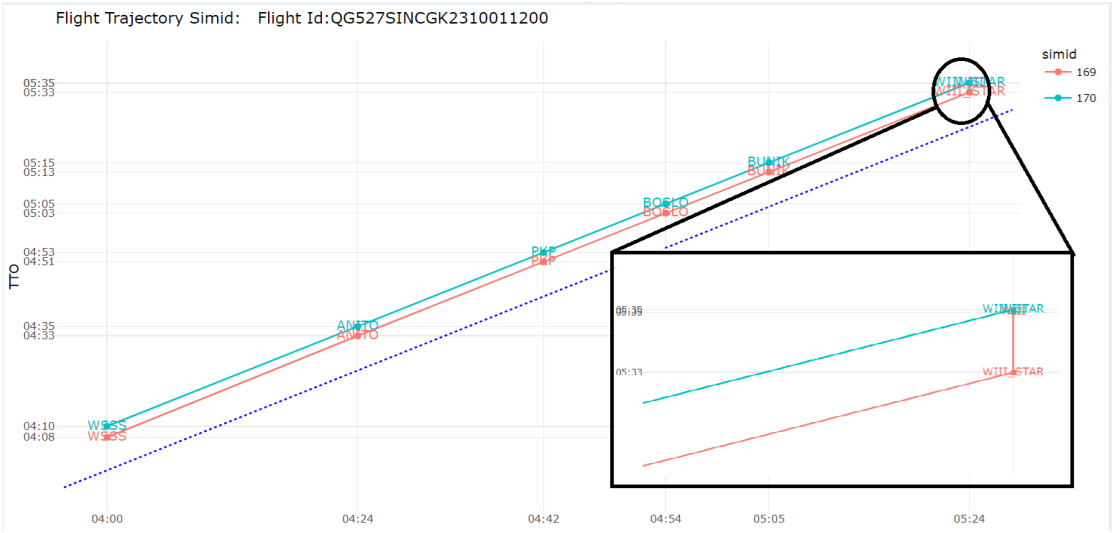

FIGURE 7.8: Example of Ground Delay With No Arrival Delay

7.2.3 Incremental Adoption of the FF-ICE R1 Concept

The purpose of this research project was to develop tools and to investigate the benefits of information sharing and collaboration in the ASEAN region. It is not to be expected that all FIRs will advance to release level R1 in the near future. All Air Navigation Service Providers will need to examine the cost-benefit of infrastructure investments carefully before taking such steps. It is anticipated that the FIRs already engaged in the Distributed Multi-Nodal ATFM Project would be among the first to qualify their systems for R1. Consequently, we consider several different information regimes including the core set of Manila (RPHI), Hong Kong (VHHK), Bangkok (VTBB), and Singapore (WSJC). This set is extended to five FIRs with the addition of Kuala Lumpur (WMFC) and to six with Jakarta (WIIF). It is also assumed that a single country can benefit from implementing these methods without requiring international collaboration. For this reason, and by example, we consider two countries, Vietnam and Indonesia, implementing release level R1 independent of each other and of other countries. Including the base regime of 'All FIR at R0' and widest regime of 'All Internal FIR at R1', we consider a total of seven different information regimes. Table 7.3 lists the six of these regimes with at least one FIR at level R1. In all cases we exclude FIRs that are external to this set of 14 FIRs from operating at level R1. Our model would not be realistic for such an extension. There are many more combinations of collaborating FIRs which could be considered. However, the results of this paper should be sufficient to set expectations as to the level of airborne savings possible in these other combinations.

ICAO FIRs	Region Name	Vietnam Alone at R1	Indonesia Alone at R1	Four FIRs at R1	Five FIRs at R1	Six FIRs at R1	All Internal FIRs at R1
External							
RPHI	Manila			R1	R1	R1	R1
VDPP	Phnom Penh						R1
VHHK	Hong Kong			R1	R1	R1	R1
VLVT	Vientiane						R1
VTBB	Bangkok			R1	R1	R1	R1
VVTS	Ho Chi Minh	R1					R1
VVVV	Hanoi	R1					R1
VYYF	Yangon						R1
WAAF	Ujung Pandang		R1				R1
WBFC	Kota Kinabalu						R1
WIIF	Jakarta		R1			R1	R1
WMFC	Kuala Lumpur				R1	R1	R1
WSJC	Singapore			R1	R1	R1	R1
ZJSA	Sanya						R1

TABLE 7.3: Alternative information regimes considered in addition to base case regime (All FIR at R0)

We reiterate that CTOTs and FSFS scheduling rules apply only to R1-level flights: flights for which both origin and destination airports at at release level R1. For two scenarios, Vietnam Alone at R1 and Indonesia Alone at R1, this restricts R1-level flights to domestic flights within those two countries.

Table 7.4 display the simulated savings in airborne time under the base capacity case, comparing the case where all FIRs are at FF-ICE R0, against each of the six information sharing regimes. The results are sorted by *Day*, and subsequently by *Total Flight Time Difference*. A consistent trend across all days is that having all internal FIRs at R1 provides the largest flight time savings, and Vietnam alone at R1 provides the smallest flight time savings. This can be attributed to the significantly larger percentage of flights participating in information sharing and collaboration when all FIRs are at R1. We also observe that across all simulated days, the overall airspace network benefits are proportional to the number of FIRs participating in the FF-ICE R1 program, with benefits in decreasing order, six FIRs at R1, five FIRs at R1, and four FIRs at R1. Interestingly, Indonesia alone at R1 achieves even greater savings than six FIRs at R1 about half the time. We hypothesize that this is due to that Indonesia possessing a great many small airports, which leads to a large number of collaborating airports, and by extension, collaborating flights. Given that the model assumes that every airport in the R1 regions will participate in collaborative delay programs, it appears to just require

collaboration within a single country on the surface. However, it is likely that the investment to engage all of them to the same level of participation would be substantial in practice.

We consolidate these results for the base capacity case in Figures 7.9 and 7.10. In Figure 7.9, similar to the findings in Section 7.2.2, we observe that the variance is proportional to the absolute value of total savings. We also see that savings for six FIR and Indonesia alone at R1 is almost identical, except that the median for six FIRs at R1 is higher, indicating the average airborne savings is higher for the six FIRs at R1 information regime. Next, in Figure 7.10, we plot the sum of savings for the entire week. While it is clear that having all FIRs at R1 yields the greatest airborne savings,the results also show significant savings under FF-ICE R1 even for a single country, or multiple FIRs. For example, Indonesia alone yields 44% of the savings that is achieved by having all FIRs at R1 and having four FIRs at R1 yields 22% of the savings that is achieved by having all FIRs at R1. This is a significant first step in demonstrating the value of information sharing and collaborative measures, even under the pretext of incremental participation.

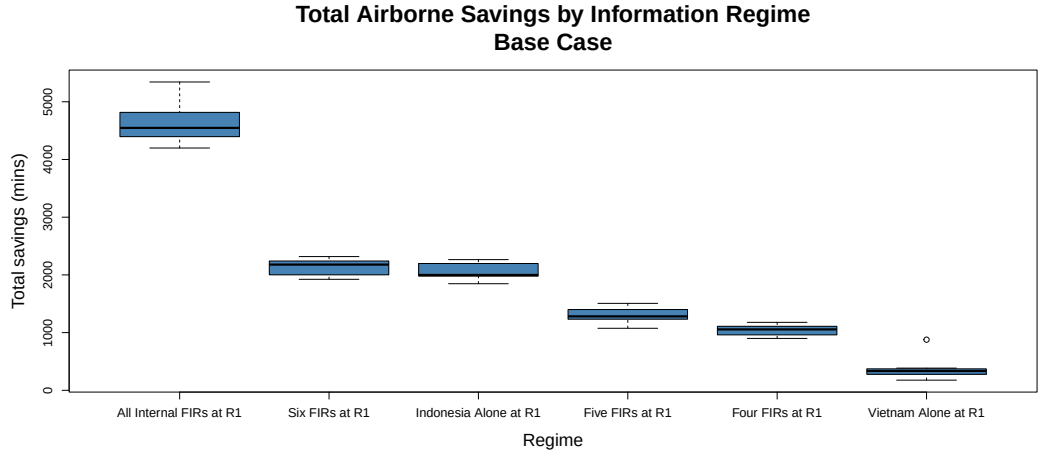

FIGURE 7.9: Airborne Savings By Regime Under the Base Capacity Case

The same analysis is performed for the reduced capacity case, and the results are plotted in Figures 7.11 and 7.12. For brevity and to avoid duplication, the detailed comparison table is omitted. The trends in the results reported for the reduced capacity case are similar to the base capacity case. For example Indonesia alone at R1 and four FIRs at R1 yields 43% and 29% of the savings that is achieved by having all FIRs at R1. This increase in percentage yield of the four FIRs at R1 regime, which includes the RPLL airport in Manila, may be due to the large number of time periods which experience congestion. This is evidenced by the gigantic frequency bubbles pushing against the capacity envelope at RPLL airport in Figure 7.2, suggesting that a larger number of

TABLE 7.4: Comparison Table For Various Information Regimes, For 1 Oct 2023 to 7 Oct 2023, for the Base Capacity Case

Day	Regime	No. of Matching Flights	Avg. Flight Time (R0)	Avg. Flight Time (R1)	Total Flight Time Difference
1 Oct	All Internal FIRs at R1	8793	147.07	146.54	4601.23
1 Oct	Six FIRs at R1	8796	147.04	146.79	2261.03
1 Oct	Indonesia Alone at R1	8794	147.06	146.85	1847.10
1 Oct	Five FIRs at R1	8796	147.04	146.87	1506.60
1 Oct	Four FIRs at R1	8796	147.04	146.92	1068.02
1 Oct	Vietnam Alone at R1	8795	147.05	147.00	384.75
2 Oct	All Internal FIRs at R1	8547	152.74	152.21	4547.02
2 Oct	Indonesia Alone at R1	8549	152.72	152.49	1962.70
2 Oct	Six FIRs at R1	8548	152.73	152.50	1923.60
2 Oct	Five FIRs at R1	8549	152.72	152.59	1074.98
2 Oct	Four FIRs at R1	8549	152.72	152.61	908.70
2 Oct	Vietnam Alone at R1	8550	152.71	152.67	360.67
3 Oct	All Internal FIRs at R1	8471	153.37	152.88	4198.50
3 Oct	Six FIRs at R1	8475	153.34	153.10	1996.75
3 Oct	Indonesia Alone at R1	8472	153.36	153.13	1993.78
3 Oct	Five FIRs at R1	8475	153.34	153.19	1281.57
3 Oct	Four FIRs at R1	8475	153.34	153.21	1055.62
3 Oct	Vietnam Alone at R1	8474	153.35	153.32	228.65
4 Oct	All Internal FIRs at R1	8644	153.34	152.76	5032.03
4 Oct	Indonesia Alone at R1	8646	153.32	153.05	2264.75
4 Oct	Six FIRs at R1	8649	153.29	153.03	2220.17
4 Oct	Five FIRs at R1	8650	153.28	153.14	1224.58
4 Oct	Four FIRs at R1	8650	153.28	153.14	1151.00
4 Oct	Vietnam Alone at R1	8651	153.27	153.23	327.10
5 Oct	All Internal FIRs at R1	8697	154.75	154.25	4311.50
5 Oct	Indonesia Alone at R1	8700	154.72	154.47	2211.05
5 Oct	Six FIRs at R1	8701	154.71	154.48	2005.28
5 Oct	Five FIRs at R1	8702	154.70	154.56	1239.02
5 Oct	Four FIRs at R1	8702	154.70	154.59	1012.63
5 Oct	Vietnam Alone at R1	8701	154.71	154.67	334.52
6 Oct	All Internal FIRs at R1	8896	152.90	152.30	5342.90
6 Oct	Six FIRs at R1	8900	152.86	152.60	2317.63
6 Oct	Indonesia Alone at R1	8899	152.87	152.63	2183.65
6 Oct	Five FIRs at R1	8900	152.86	152.69	1480.28
6 Oct	Four FIRs at R1	8900	152.86	152.73	1177.93
6 Oct	Vietnam Alone at R1	8902	152.84	152.74	877.50
7 Oct	All Internal FIRs at R1	8715	154.53	154.02	4479.33
7 Oct	Six FIRs at R1	8718	154.51	154.26	2178.67
7 Oct	Indonesia Alone at R1	8716	154.53	154.30	1999.25
7 Oct	Five FIRs at R1	8718	154.51	154.36	1320.13
7 Oct	Four FIRs at R1	8718	154.51	154.41	899.23
7 Oct	Vietnam Alone at R1	8718	154.50	154.48	176.10

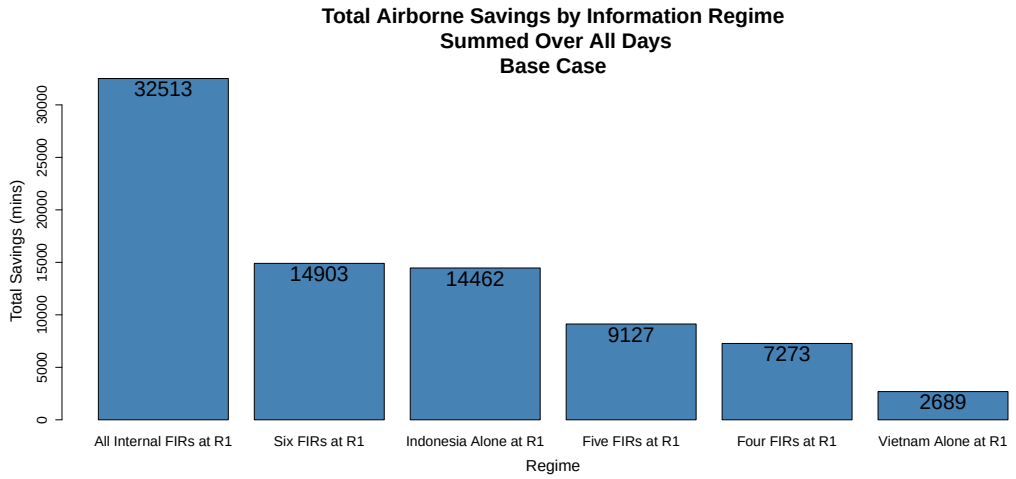

FIGURE 7.10: Sum of Airborne Savings By Regime Under the Base Capacity Case

ground delays would have been initiated at RPLL or its collaborating airport pairs, therefore leading to greater airborne savings.

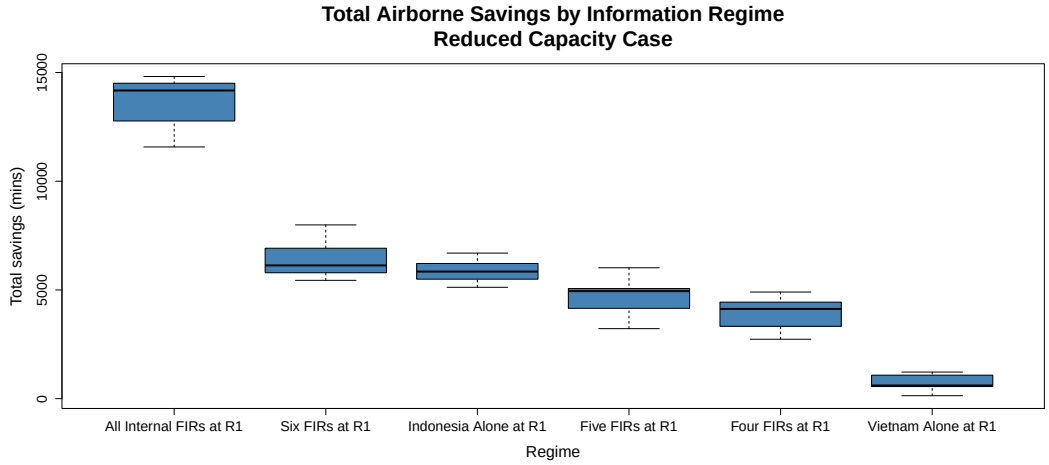

FIGURE 7.11: Airborne Savings By Regime Under the Reduced Capacity Case

We extend the argument that given the large number of airports, it is likely that even within an FIR, only particular airports will be chosen to invest resources and upgrade current systems and practices to adopt the FF-ICE R1 initiative. Our hypothesis is that even with collaboration between only two airports, there are savings to be made. To investigate this conjecture, we further restrict information sharing and collaborative GDPs to occur between flights with a particular origin-destination pair. Table 7.5 presents the results for collaboration between airports in multiple cities, paired with

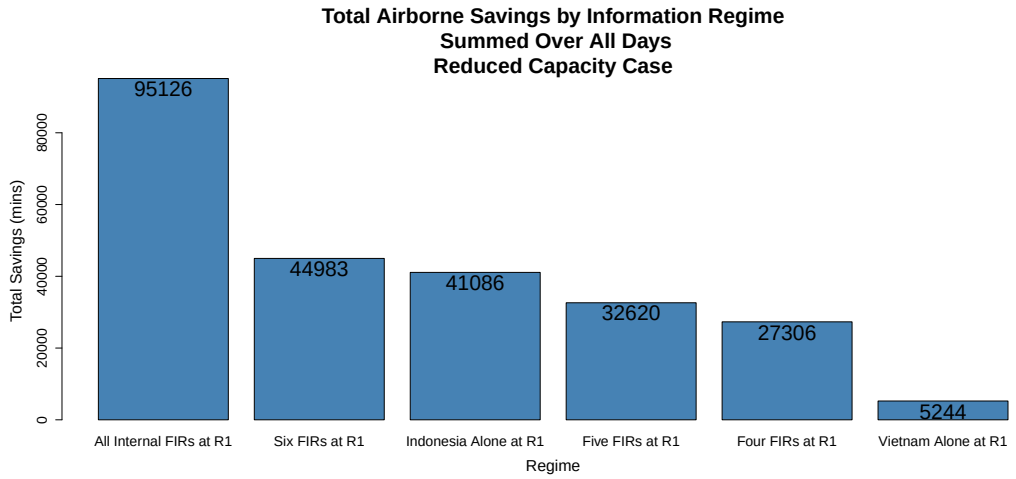

FIGURE 7.12: Sum of Airborne Savings By Regime Under the Reduced Capacity Case

WSSS Singapore Changi Airport. Here, we begin to observe mixed results, with information sharing and collaborative decision-making leading to a net delay at VHHH in the base capacity case, and both VHHH and WIII in the reduced capacity case. This indicates that under very limited collaboration, there exists the possibility that minor GDP assignments could disrupt the flight schedules of non-collaborating flights, rendering the net impact of GDPs to be indeterminate. We plot the total flight time difference against number of GDP flights in Figure 7.13. Apart from the anomalous point at 45 GDP flights, all other points appear to follow a clear trend, with the flight savings increasing in tandem with the number of GDP flights, corroborating our previous findings. There appears to be a threshold number of flights, at approximately 15, where any fewer than this number of flights would lead to a net increase in delay, albeit small. These results suggest that meaningful savings under the FF-ICE R1 initiative depend on achieving a critical level of collaboration.

7.2.4 Identifying Sources of Savings and Delays Under FF-ICE R1

A natural question to ask at this point is where do the savings come from in general. This requires a deeper dive into the simulation results. Figure 7.14 presents the results for the total airborne savings, aggregated over each hour of the day, for the flights occurring on 1 Oct 2023. The stacked bars represent the airborne savings of which the *Airport Group* experienced, computed over all of its arrival flights. We see that in general, the height of the bars are proportional to the total number of flights at each time period. Interestingly, the top six major airports, represented by colored bars except for the purple non-hub airport bar, contributed to only a small percentage of the airborne savings, of about 25% or less, when compared to all other non-hub airports. This

TABLE 7.5: Comparison Table For Various Airport Pair Collaborations With WSSS Singapore Using 1 Oct 2023 Flight Plans

Capacity	Airport Pair	No. of GDP Flights	Total Flight Time Difference (s)
Base Case	VTBS Bangkok	26	99.60
Base Case	WMKK Kuala Lumpur	56	144.65
Base Case	RPLL Manila	15	11.08
Base Case	VHHH Hong Kong	12	-42.93
Base Case	WIII Jakarta	40	64.98
Reduced Case	VTBS Bangkok	28	103.17
Reduced Case	WMKK Kuala Lumpur	71	225.22
Reduced Case	RPLL Manila	21	46.32
Reduced Case	VHHH Hong Kong	12	-21.25
Reduced Case	WIII Jakarta	45	-79.37

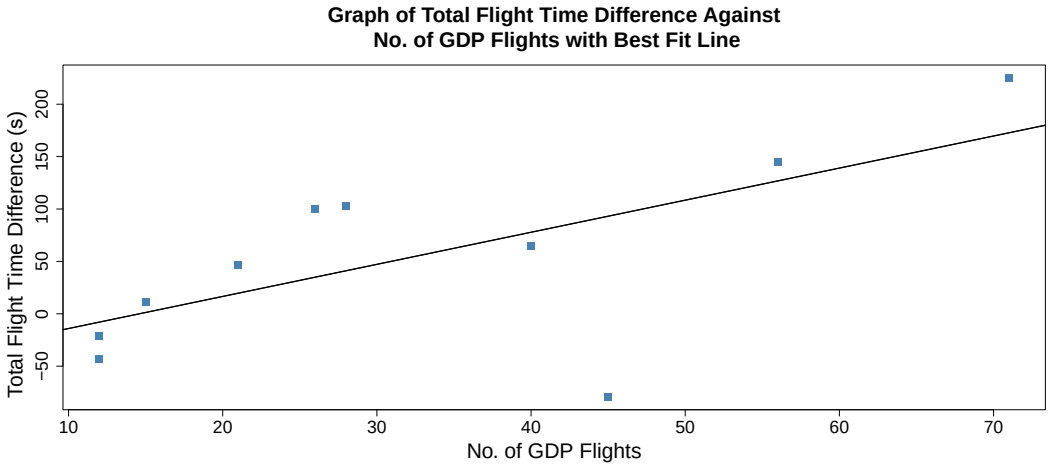

FIGURE 7.13: Total Flight Time Difference Between All R0 and Airport Pairs Collaboration Plotted With Data From Table 7.5

demonstrates that savings are distributed across many smaller airports, highlighting the value of collaboration even at less busy locations. We also notice that some airports as a whole experience small amounts of additional delay, such as RPLL at hours 9 and 10. This is likely due to the scheduling of flights to different channels under R0 and R1, where a number of departure flights utilized the mixed channel capacity, reducing the capacity for arrival flights, ultimately leading to arrival delays. Additionally, having too few GDP flights leading to an overall net increase in delays, was observed at hours 18 and 19, where the number of flights was at its lowest. This phenomenon was similarly observed earlier, in Section 7.2.3.

In Figure 7.15, which presents the same histogram for the reduced capacity case, we see that the problems encountered in the base capacity case have been resolved. RPLL (the second highest stacked bar) no longer experience additional delays at hours 9 and 10, and in contrast, benefited from a significant amount of airborne savings. We recall from the capacity envelopes in Figure 7.2, that RPLL was highly congested under the reduced capacity case, and this result reinforces our premise that airborne savings are proportional to the congestion experienced. We also see that at hours 18 and 19, despite the limited flight activity during the period, successful collaborative measures were carried out in the airspace network and airborne savings were achieved instead. We also notice that the top six major airports contributed to a significantly larger percentage of airborne savings, which again may be attributed to the top six airports operating at close to maximum capacity and hence were more active with the assignment of ground delays.

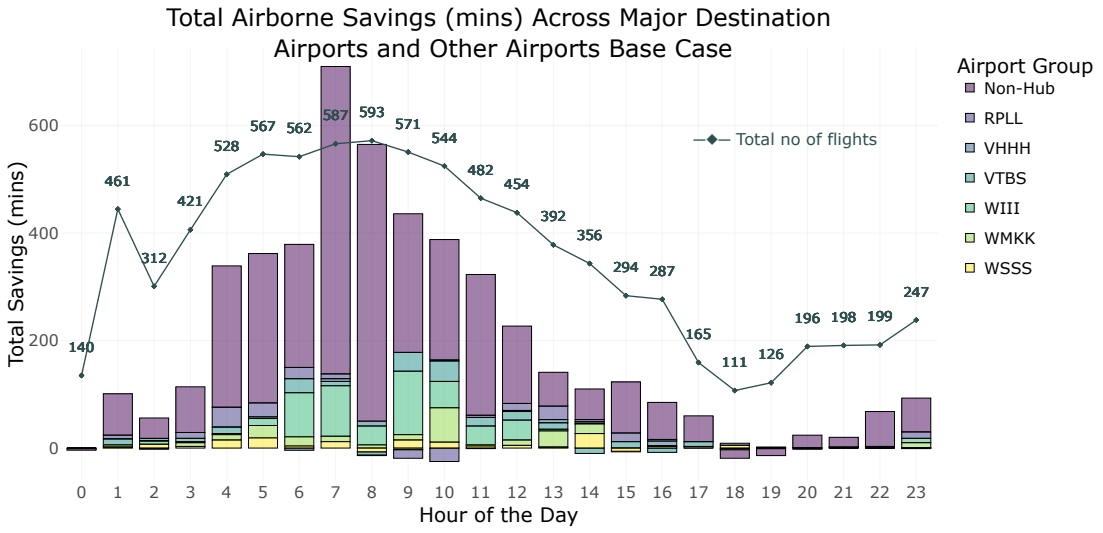

FIGURE 7.14: Airborne Flight Savings Per Hour Under the Base Capacity Case, Highlighting the Savings by the Top Six Airports (Ranked by Number of Inbound Flights)

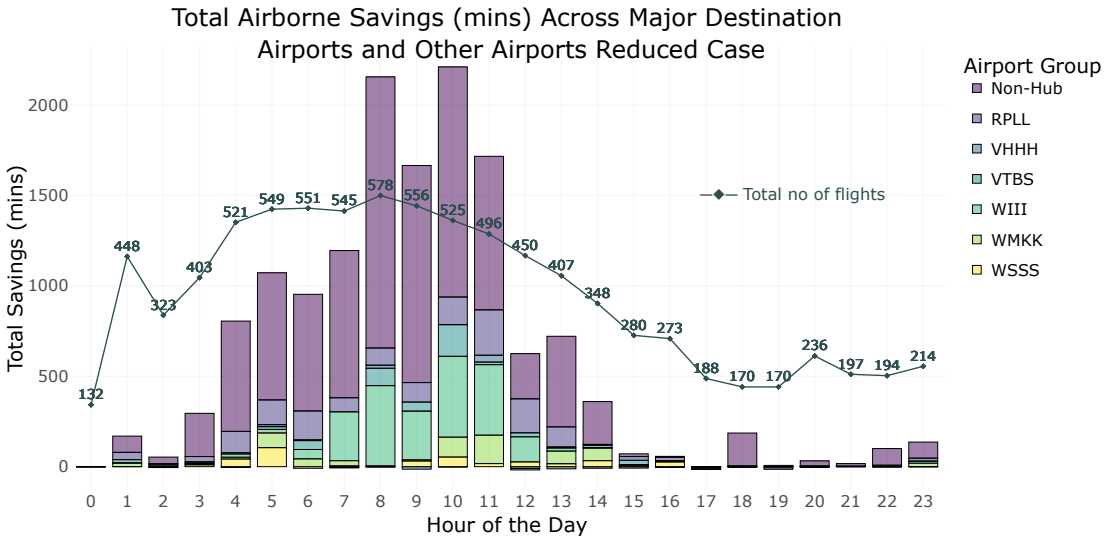

FIGURE 7.15: Airborne Flight Savings Per Hour Under the Reduced Capacity Case, Highlighting the Savings by the Top Six Airports (Ranked by Number of Inbound Flights)

Next, we look at the savings and delays by airport pairs. Figure 7.16 displays screenshots from our analysis tool which summarize the simulated arrival data for each selected airport, overlaid across the world map. The maps display the city pairs with the greatest savings in green or delays in red. We only plot lines for the city pairs whose sum of absolute flight time difference is greater than mean absolute flight time difference for all city pairs. What is interesting in these maps is the revelation that many of the significant savings or delays occur with airports outside the ASEAN plus region, indicated by the numerous plotted lines that leave the region. This raises a discussion point on how long-haul flights are to be treated when implementing regional collaborative measures. By assuming all external FIR are at level R0, we are clearly simulating a system where such long-haul flights could be disadvantaged when approaching a congested airport relative to delayed flights from regional airports. On the other hand, we also observe many green lines spanning outside of the ASEAN Plus Region in Figure 7.16, so such long-haul flights are not universally disadvantaged. By delaying some regional flights it is possible that a long-haul flight serendipitously can have its airborne time reduced, either due to the change in airport channel assignment, or because ground delayed R1 flights result in the long-haul flight being reassigned earlier in sequence.

We also include Figure 7.17 which depicts savings and delays for two main airports in Vietnam under the Vietnam alone at R1 information regime. It is clear that the biggest savings come from domestic flights within Vietnam. This highlights the potential for information sharing and collaboration within a single country alone to be

beneficial for the participants. However this may come at a cost of disruptive delays to flights coming from other regions. In the right map plot for VVNB, flights coming from Bangkok, Hong Kong and Tokyo experienced greater delays when airports in Vietnam participate in the FF-ICE program. This again raises the question on how international flights should be treated, highlighting another point for countries to consider for the implementation of FF-ICE R1.

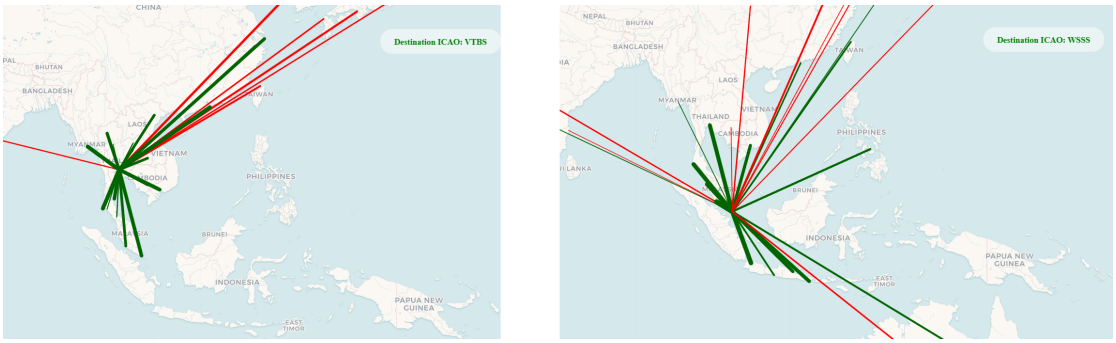

FIGURE 7.16: Map of Airport Pairs with Significant Airborne Savings in Green and Significant Airborne Delays in Red, for VTBS Bangkok and WSSS Singapore Under All R1, Base Capacity Case, 1 Oct 2023

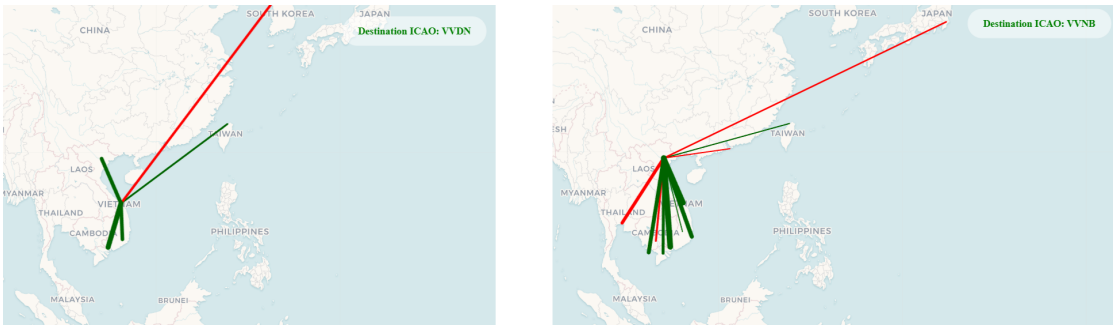

FIGURE 7.17: Map of Airport Pairs with Significant Airborne Savings in Green and Significant Airborne Delays in Red, for VVDN Da Nang and VVNB Hanoi Under All R1, Base Capacity Case, 1 Oct 2023

Next, we zoom into the savings or delays generated by either GDP and non-GDP flights and identify the savings under limited information sharing and collaborative decision-making between a single airport pair. A flight is determined to be a GDP flight if it was assigned a GDP in the partial R1 scheme between airport pairs. Using the flight ids of these GDP flights, any non-GDP flight that departed or arrived within 1800 seconds (30 minutes) of a GDP flight, either in the R0 or partial R1 simulation is considered a *non-GDP flight within 1800 seconds of GDP flights*. All other non-GDP flights are denoted *non-GDP flights outside 1800 seconds of GDP flights*. We again focus on the same airport collaboration pairs as in Section 7.2.3, as these are the airports that are among the most active within the ASEAN Plus region. In both Figures 7.19 and

7.20, the green bar representing only GDP flights, is always positive. This indicates that, under all scenarios, flights subject to ground delay programs almost invariably achieve airborne savings, even when collaboration is limited to just two airports within the airspace network. A surprising observation for the base capacity case, is the minimal impact to non-GDP flights within 1800 seconds of any GDP flights (denoted by the purple bars), but a significant impact on non-GDP flights that are not close to any GDP flight (denoted by turquoise bars). We expect that flights that are distant from GDP flights to not be impacted, and flights close to GDP flights to experience a greater impact, however, the results suggest otherwise under the base capacity case, and at WMKK under the reduced capacity case.

Further inspection of the results reveal an interesting phenomenon — flights that are not close to any GDP flights at either of the collaborating airports, namely WSSS and WMKK, nor have WSSS or WMKK as its origin or destination airport, are severely impacted by the WSSS-WMKK collaboration. The results supporting this statement is presented in Table 7.6. Such a phenomenon occurs because conflicts between GDP flights have the potential to come into conflict with flights between other regions, and cause minor enroute delays. These flights between other regions are unable to conduct their own GDPs to avoid conflict, and hence simply absorb the savings or delays at the arrival airport, leading to significant savings or delays as a whole, despite not participating in the FF-ICE R1 program. A particular case where this occurs in our simulations is given in Figure 7.18. In Figure 7.18, the graph on the left and right represent the flight legs and their corresponding TTOs at waypoint CON, a waypoint close to Hong Kong, under the all R0 regime and WSSS-WMKK regime. We notice, that even at a distant waypoint CON, the arrangement of the bars are different, highlighting the ripple effects of collaborative GDPs between WSSS (Singapore) and WMKK (Kuala Lumpur). Aggregated at the network level, these perturbations yield a noticeable difference in flight schedules, even for flights that are not participating in the FF-ICE R1 program.

From the results in Table 7.7, a similar conclusion can be reached for non-GDP flights within 1800 seconds of GDP flights, except that most savings or delays occur at the collaborating airports WSSS and WIII. Here, most of savings or delays are caused by the rescheduling due to newly generated conflicts at the collaborating airports rather than at the waypoints traversed by GDP flights.

As demonstrated in Section 7.2.3, a threshold number of GDP flights is often required before net positive airborne savings are observed. For instances of severely limited collaboration, such as that between two airports, due to the highly complex and interconnected nature of the global airspace, managing the broader impacts on other flights raises a key concern for stakeholders. The results show the two airports consistently will enjoy airborne savings, with even greater savings under the reduced

FIGURE 7.18: Flight Trajectory Graph, Waypoint CON Only, For All R0 Regime (Left) and WSSS-WMKK Collaboration (Right)

capacity case. However, the overall impact on the wider airspace network remains limited and inconsistent when only two airports participate in the FF-ICE R1 program. This inconsistency is particularly pronounced in the reduced capacity case, where increased congestion leads to a higher likelihood of conflicts at waypoints or airport nodes, driven by the chain effects of the localized GDP measures.

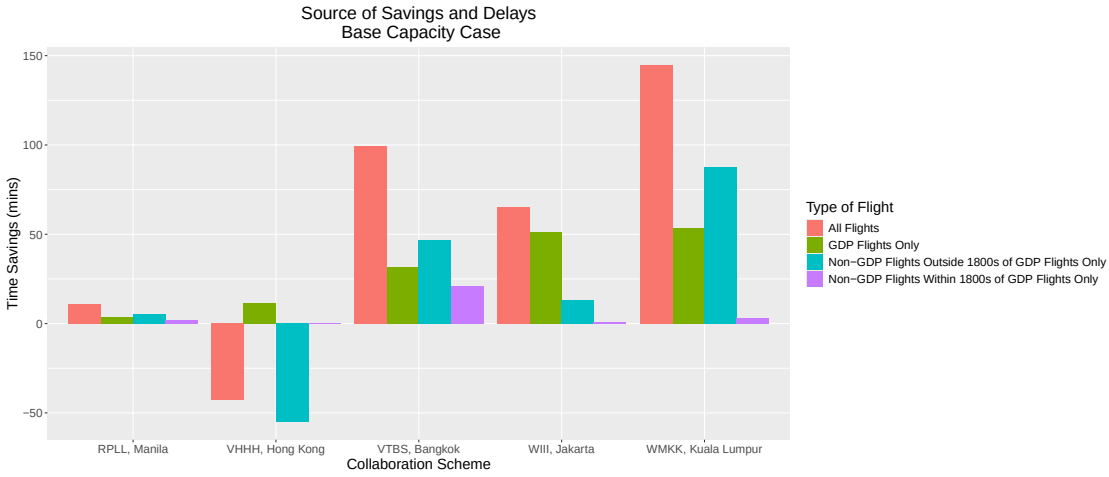

FIGURE 7.19: Comparison of Various Airport Pair Collaborations With WSSS Singapore. Barplot of Flights Grouped by Time From GDP Flights, Base Capacity Case, 1 Oct 2023

The purpose of this chapter was to measure the potential benefits in reduced airborne time that can come from implementing at least FF-ICE R1. Our results demonstrate that benefits exhibit the characteristics of network goods, highlighting the collective value of widespread adoption. Here a network good is defined as one where

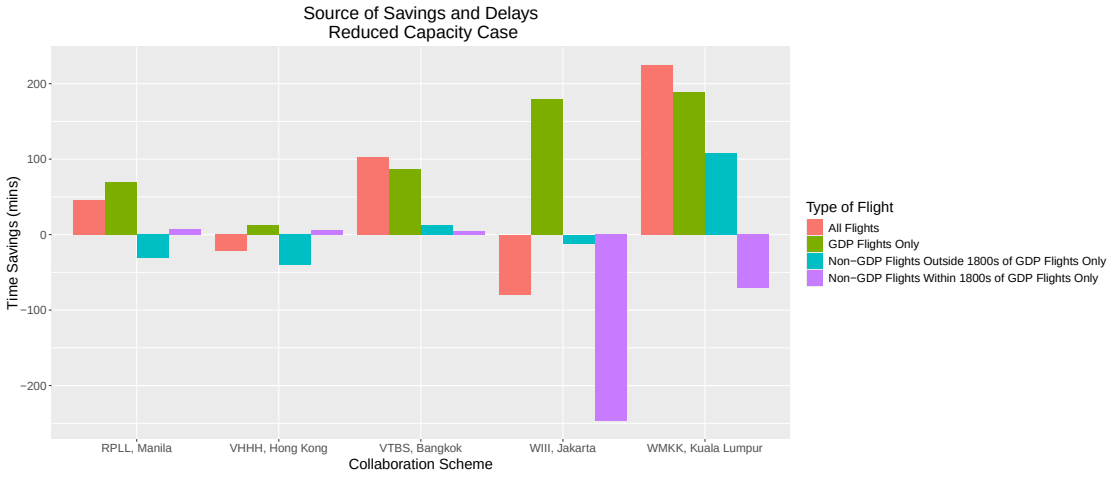

FIGURE 7.20: Comparison of Various Airport Pair Collaborations With WSSS Singapore. Barplot of Flights Grouped by Time From GDP Flights, Reduced Capacity Case, 1 Oct 2023

TABLE 7.6: Top Airport Pairs by Sum of Absolute Flight Time Difference for Flights Outside 1800 seconds of GDP Flights Under the WSSS-WMKK Collaboration Scheme Base Capacity Case

Airport Pair	Sum of Absolute Flight Time Difference
ZGHA-ZJSY	24.43
ZBAA-ZJSY	21.17
ZGSZ-ZJSY	20.68
VTBD-VTSP	17.20
ZGGG-ZJSY	16.27
RCTP-VMMC	15.95
ZJSY-ZSPD	15.07
RKSI-VTBS	14.20
VTBD-VTCC	12.95
VHHH-VTBS	12.80

TABLE 7.7: Top Airport Pairs by Sum of Absolute Flight Time Difference for Flights Within 1800 seconds of GDP Flights Under the WSSS-WIII Collaboration Scheme Reduced Capacity Case

Airport Pair	Sum of Absolute Flight Time Difference
WAAA-WIII	50.02
WADD-WIII	49.93
WIII-WIMM	46.27
WAHI-WIII	29.20
WARR-WIII	28.60
WMKP-WSSS	25.82
WIII-WMKK	22.38
VTBS-WSSS	18.23
WADD-WSSS	17.63
WIEE-WIII	13.12

current users gain when additional users adopt the system, with classic examples such as the telephone and payment systems (Klemperer, 2008).

Accordingly, we have looked at different information regimes and demonstrated that the benefits increase with the number of airports participating and with the congestion levels they face, with GDPs selected as the primary procedure of collaborative decision-making. Our findings reveal a critical threshold for the number of participating flights, of at least 15, before net savings are realized. Crucially, the effectiveness of GDP programs increases more than proportionally under situations of reduced capacity, where a 20% loss of capacity resulted in an increase in savings of over 200%. We have also demonstrated that a single country can benefit from such investment, even in the absence of participation from other countries, primarily because collaboration within a single country, such as Indonesia or Vietnam, involves a sufficient number of flights to reliably achieve net savings across the entire ASEAN Plus airspace. Our results also demonstrate that regional collaborative ground delay programs must include some form of preferential treatment for delayed flights of which the First-Scheduled-First-Served priority rule for R1 level flights was implemented.

Finally, our findings reveal a surprising result — flights not participating in FF-ICE R1, even those with an origin-destination airport pair that is not at R1, are often impacted by the rescheduling of participating flights, particularly when they share a common node, either a waypoint or an airport node. Perhaps this is a limitation of our model, where we do not consider multiple flight levels at waypoints, potentially imposing unnecessary flow restrictions at waypoints that are not present in actual flight operations. Despite this, our findings highlight a key concern of how non-participating flights will be impacted by the FF-ICE R1 initiative, especially for long-haul flights,

which often operate under differing airspace regulations and procedures.

Chapter 8

Conclusion

Amidst factors including the continued strong growth in the aviation sector, the mismatch between airspace capacity and demand, and incremental implementation of the FF-ICE initiative, airspace management retains its position at the forefront of aviation research. Imbalance of capacity and demand is well known to lead to increased flight delays and high costs incurred by airlines, airports and passengers. As we recall, a study done by the Federal Aviation Administration (FAA) estimated delays to cost a total of 33.0 billion USD in 2019, with passengers bearing the brunt of the cost due to time lost from schedule buffers, delayed flights, flight cancellations, and missed connections (FAA, 2019). The cost and time required to upgrade physical infrastructure to increase airspace capacity are substantial. Consequently, improving the efficiency of air traffic management has been regarded as a more practical focus, particularly within the research community. In the literature reviewed, there have been numerous studies on the airspace optimization, however, almost all the studies for complete trajectories assumed the presence of a central controller, such as the Federal Aviation Administration's Air Traffic Organization, that manages decisions. In a decentralized system like Southeast Asia, such control does not currently exist, resulting in tactical measures being implemented only within individual Flight Information Regions (FIRs). Therein lies the dichotomy, that most research is not applicable to decentralized airspace networks such as Southeast Asia, motivating the key question that this dissertation aims to address. Furthermore, while a number of papers and technical reports have been published on the operational concepts and qualitative analyses of FF-ICE (ICAO, 2012; D. Liang, Ngo, et al., 2021; EUROCONTROL, 2024), there has been a lack of quantitative research on this topic. Therefore, the primary focus of this dissertation has been to address the gap in the literature, conducting a quantitative analysis on the benefits of information sharing for various combinations of participating stakeholders, using historical flight schedules and network capacity within the ASEAN region to simulate the ASEAN Plus airspace network. The results are promising, indicating airborne savings for information sharing, both between and within countries, highlighting the benefits of collaboration as the air travel industry continues to grow.

The software deliverable for this dissertation, and associated research project, the FF-ICE simulator, incorporates foundational algorithms and components that enable a systematic analysis of various scenarios within the ASEAN Plus airspace. With further data collection, the simulator could be expanded to model additional regions. Moreover, the FF-ICE simulator may be extended to support analyses such as re-routing strategies, alternative information-sharing regimes, and the evaluation of different concepts of operations through representative simulation scenarios. We believe this tool will assist ANSPs at all planning phases, from the strategic, to pre-tactical and tactical phase, and allow ANSPs to test and better understand the potential benefits of incremental participation in the FF-ICE initiative in a decentralized airspace network. The system definition and user interface design can be found in Appendix C. The data collection, ideation, system architecture and programming of the FF-ICE simulator is a team effort by the researchers at the ASI, consisting of Prof. Peter Jackson, Prof. Nuno Ribeiro, Rakesh Nandi, Dai Gengling, and myself. Additionally, we thank Prof. Daniel Delahaye for providing constructive feedback on the operational standards of air traffic management, and code for the simulated annealing algorithm.

8.1 Summary of Research

In this dissertation, we have proposed several frameworks for building a realistic flight simulator in decentralized airspace networks, such as those in Southeast Asia. First, we provided a brief overview of these methods in Chapter 3. The FF-ICE simulator consists of two modules, the UPDATE and SOLUTION module. The UPDATE module updates the simulator according to the selected information sharing regimes, and prepares for the next time step. The update is done through a Rolling Horizon Concept and Information Sharing between flights or regions participating in the FF-ICE R1 initiative. Next, the SOLUTION module solves the problem of regulating flow at both airports and waypoints, either using the Airport Flow Regulator (AFR) in conjunction with the Waypoint Flow Regulator (WFR) algorithm, or the Discrete Event Simulation (DES), which have demonstrated the ability to generate conflict-free trajectories. The UPDATE and SOLUTION modules operate alternately, exchanging data between each other at regular simulated time intervals to accurately represent operational flight activity, and uncertainty such as changes in the flight trajectories within the airspace network. This is achieved by simulating the transmission of information between aircraft and ANSPs, and updating of flight trajectories once per time step, which represents the sharing of information between participants of the FF-ICE R1 initiative. At the end of Chapter 3, We listed the data sources used to build the FF-ICE simulator.

In Chapter 4, we provide the mathematical formulation of the AFR, whose role is the scheduling of aircraft to channels in the context of ATFM measures and capacity envelopes. The capacity envelope concept, allowed us to model aircraft flow at airports using three channels, the arrival, departure, and common channel, with specified maximum flow rate at each channel. This flow rate was derived from the capacity envelope representing the maximum number of arrivals, departures, and total operations in a given time period of one hour, at all airports. We then described the mathematical formulation of converting these flow rates into minimum required separation times between aircraft, and applied a novel queue pressure algorithm, inspired by the shifting bottle procedure (Adams, Balas, and Zawack, 1988). This algorithm, as the name implies, identifies the queue, either arrival or departure, as a bottleneck and attempts to first schedule flights from that queue. The results of the queue pressure algorithm were then compared against a classic FCFS algorithm. The results indicate no significant benefits, with the solution quality differing by less than 1%. We believe this may be attributed to the ATFM model assumptions, where all flights at any given channel are required to satisfy a unique minimum separation requirement, rather than different separation requirements based on weight class, thereby lacking the benefits of flight sequencing. As such, we have opted to use the FCFS algorithm for the reasons of interpretability and quicker computation times.

In Chapter 5, we discuss the WFR. The goal of the WFR is to issue speed regulation and hold commands to aircraft such that the minimum required separation between any pair of aircraft at all nodes are observed. We presented the mathematical model and necessary assumptions, and proved propositions that guided our design choices, including how we handled frozen flight legs to reduce the state space of the problem. We highlighted that the WFR is a non-trivial combinatorial problem, which motivated our decision in applying heuristic methods. We then continued the chapter by describing the first algorithm applied to the WFR, the Gradient Descent Ascent (GDA) algorithm, including the formulation of the Lagrange dual and partial derivatives necessary for the computations. We also described some of the problems faced by the GDA algorithm, and how we overcame them. In particular, we formulated a pre-sequencing algorithm that provided a good initial solution to the GDA, which resulted in conflict-free trajectories. We conducted a similar study on the Simulated Annealing (SA) algorithm, and benchmarked the results of the GDA and SA against exact methods. Considering the reduced capacity case, GDA offered the best computational performance, resolving all conflicts within a decision window in under 9 seconds. By comparison, SA, EM with a 10-second time limit per region (EM10), and EM with a 600-second time limit per region (EM600), required 75, 102, and 5413 seconds, respectively. While all conflicts were resolved much more rapidly, the deviation from scheduled times under

GDA was 26% worse than the best solution obtained. The SA algorithm generated solutions that were 10% better than GDA, 8% better than EM10, but 13% worse than EM600. We then weighed the benefits and drawbacks for each of the proposed methods. Finally, we concluded that the successful application of the GDA, SA, and exact algorithms to the WFR demonstrated the utility of the methodological framework developed in the chapter for simulating air traffic within the decentralized ASEAN Plus airspace network.

In Chapter 6, we present a rule-based modeling approach that addresses the problem in a conventional and deterministic manner. The Discrete Event Simulation (DES) framework encompasses the functionalities of both the AFR and the WFR, handling the assignment of airport channels to aircraft and the scheduling of conflict-free trajectories. In essence, the DES sequentially schedules available aircraft legs while considering applicable information-sharing regimes and enforcing minimum separation requirements. Flights participating in information sharing may substitute airborne holding with ground holding and are granted priority at downstream nodes under a FSFS policy. In contrast, flights that do not participate in information sharing are scheduled according to FCFS. The explicit FSFS mechanism for R1-level flights ensures that flights delayed due to CTOT restrictions are prioritized over flights that would not have been ahead without the CTOT delay. We argue that FSFS offers a more realistic representation of collaborative GDPs than FCFS. It is unlikely that airlines and origin airports would be willing to cooperate in a system from which they derive no operational benefit, one in which an aircraft faces departure disruptions and incur arrival delays without queue priority at the network nodes.

In Chapter 7, we argued why the DES is the algorithm of choice for the FF-ICE simulator SOLUTION module. We demonstrated that regional collaborative GDPs must incorporate some form of preferential treatment for delayed flights, which the results show the FSFS principles to be a suitable choice. With the coordination of GDPs and FSFS scheduling, the results demonstrate airborne savings of 12.3% and 17.4%, for flights with more than 5 minutes of airborne savings under the regular and reduced capacity case respectively. We also looked at different information regimes and demonstrated that the benefits increase with the number of airports participating and the congestion levels they face. Our findings show that a 20% reduction in airspace capacity led to an increase of airborne savings of 200% under the FF-ICE R1 regime. Importantly, given that incremental participation is anticipated to occur, we have also demonstrated that a single country, or a group of countries can benefit from such investment, without complete participation with the ASEAN Plus region. In particular, with all 14 FIRs participating as the baseline, the savings range from 8% for a single FIR (Vietnam only), 22% for four FIRs, to 46% for six FIRs. We go further to measure the benefits a single

airport pair collaboration can yield, demonstrating that flights participating in collaborative GDPs consistently enjoy reduced airborne delay. However, these benefits may be outweighed by the unpredictable rescheduling of other non-participating flights, particularly under time periods with tight capacity where any single rescheduled flight may cause downstream flights to be erratically rescheduled.

8.2 Future Research Directions

This dissertation has demonstrated the value of information sharing and collaboration as constant progress is made towards FF-ICE R1, and laid the groundwork for future research related to incremental participation in information sharing systems between regions, airports, and airlines. We believe the core contribution of our research, the FF-ICE simulator, and the insights derived from our analyses, will provide a foundation for future research to be conducted in the ASEAN airspace and beyond, particularly in response to growing capacity pressures. In light of the the constant pressure to address the ever growing demand for air travel and transport, we pose a list of questions and problems we would seek to answer over the next few years.

- **Migration Towards FF-ICE R2:** FF-ICE R1 introduces services that allow for collaboration between flights at the pre-departure phase. Our research has demonstrated promising results for cooperation between FIRs, with the primary ATFM procedure selected as ground delay programs. As the global aviation agencies and regulators prepare for the next phase of FF-ICE, the FF-ICE R2, we plan to extend our research to incorporate and support the FF-ICE R2. To do so, we plan to continuously update our FF-ICE simulator to allow for post-departure negotiation between airspace users and ATM service providers. Possible updates include the possibility of re-routing flights, assigning the FF-ICE release level based on flights rather than region or airport, and an increased focus on the intersection between the fields of ATFM and TBO.

- **Testing the FF-ICE Simulator by Industry Stakeholders:** The FF-ICE simulator developed during our research has been handed over to and is currently under evaluation by the CAAS for further study through the simulation of various collaborative regimes. Their feedback will lead to further refinements of the capabilities of the FF-ICE simulator. Furthermore, if the results of the simulator can be validated through operational tests with other ANSPs in the ASEAN region, this would be the most accurate barometer of the accuracy of our modeling and assumptions of the FF-ICE concept. We would continue to work with CAAS and disseminate our work to other ANSPs in the region, facilitating the discussion

and proposals of the collaborative regimes and ATFM measures that may be considered in future FF-ICE releases.

- **Compare the Results of Decentralized Operations to Centralized Operations:** The core theme of the dissertation has been studying decentralized operations, with results demonstrating that incremental participation in FF-ICE R1 will lead to a reduction in airborne time, as well as fuel consumption. As many studies on a single centralized ATC using exact methods have been published, such as in (Balakrishnan and Chandran, 2014; Tan, 2021; Bolić et al., 2017), future work may include collaborating with these authors to establish the differences in trajectories, airborne and fuel savings on a common dataset. An optimistic estimate of when all countries would collaborate would be decades at earliest due to sensitive data sharing and system upgrading costs. However, it could turn out that it is unnecessary in the first place, and partial collaboration could elicit most of the benefits. Such comparisons would allow us to establish a lower bound to the airspace optimization, gain deeper insight as to where inefficiencies arise from decentralized operations, and quantify the proportion of the benefits that can be gained from partial participation in collaborative decision-making programs in a decentralized airspace.

- **Propose Data-Backed Concept of Operations:** We aim to extend the research on FF-ICE, using our FF-ICE simulator to test and propose other concepts of operations, or ATFM procedures. Preliminary testing has been done on fuel consumption, and delaying transcontinental flights en route if delays are predicted to occur in the TMA. However, the results have been mixed, due in part to the sensitivity of fuel data collection efforts. Other operations that we could run simulations on include platooning and rule-based scheduling with different priority rules. Additionally, examining operations within the TMA could also uncover opportunities for collaboration and benefits could arise from employing different ways to delay flights such as vectoring, holding, and point merge, in a decentralized airspace network.

- **Increased Fidelity of the Model:** Our ATFM model is flow based, and does not take into account aircraft weight classes. In practice, weight classes determine the wake vortex turbulence and by extension, the required separation between aircraft on runways, and therefore sequencing of aircraft on runways by weight class. Hence, one potential avenue to enhance the fidelity of the FF-ICE simulator is to incorporate aircraft weight classes, which may enable more refined operational strategies and unlock new opportunities for optimizing the airspace network. We anticipate this change would yield larger benefits as studies have

demonstrated a reduction in makespan of 5% to 16% (Balakrishnan and Chandran, 2010; Desai et al., 2022), in contrast to our study in Chapter 4, where a fixed separation time across all aircraft weight classes which was shown to be no better under a queue pressure or FCFS algorithm. The second way would be to limit the number of CTOT ground delays permitted per unit time. This reflects the operational procedures taken in Europe, where the number of CTOT slots are limited to prevent a concentration of aircraft within any specified time period (Ma, 2019), for the purpose of more efficient airspace utilization.

Appendix A

Computational Analysis And Considerations For Gradient Descent

The crux of this section is to detail how for-loops were avoided in R, as these require multiple function calls in R, as compared to the vectorization method of programming, which interfaces C and FORTRAN, while requiring only a single function call in R.

We start by enumerating through every waypoint $w \in W_I$, utilizing the `combn()` function to generate all possible combinations of conflicts at each waypoint. The `inner_join()` function then combs through previously initialized data to add more information such as *equipment_id*, *time_id* and *required_separation* for each conflict.

The partial derivatives with respect to λ are then straightforward to construct, and can be done by creating two vectors, storing the time values of all nodes n for flight f and flight f' with length equal to the cardinality of λ. Subsequently, the `pmin` function in R can be used to enumerate over all indices in λ, and simultaneously compute $\min\left\{\frac{\partial\mathcal{L}(\beta^{(n)},\lambda^{(n)})}{\partial\lambda}, \delta_w\left(f, f'\right)\right\}$. Since the partial derivatives of the Lagrangian with respect to β and λ require us to find the values of t_{fn}, we have to do a cumulative sum over all the records for each unique flight. Here, we recommend `ave()` over dplyr's $group\_by()\%>\%mutate()$ as we found it to be roughly 5 times quicker. Here, the pipe operator in R is denoted as $\%>\%$.

The partial derivatives with respect to β require some programming tricks to avoid for-loops. We begin by providing a small-scale example of the implementation. Assume flights 1 and 2, both traveling entering node 1, traveling to node 2, then exiting at node 3. We will initialize and subsequently take the dot product of two matrices,

which possess the datatype `sparse Matrix of class "dgCMatrix"` as such:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 2\left(t_{13} - \tilde{t}_{13}\right) \\ 2\left(t_{13} - \tilde{t}_{13}\right) \\ 0 \\ 2\left(t_{23} - \tilde{t}_{23}\right) \\ 2\left(t_{23} - \tilde{t}_{23}\right) \\ \lambda_{121} k\left(t_{11}, t_{21}\right) * \text{sgn}\left(t_{11} - t_{21}\right) \\ \lambda_{122} k\left(t_{12}, t_{22}\right) * \text{sgn}\left(t_{12} - t_{22}\right) \\ \lambda_{123} k\left(t_{13}, t_{23}\right) * \text{sgn}\left(t_{13} - t_{23}\right) \end{bmatrix} \tag{A.1}$$

The left chunk of the left matrix, is a diagonal square matrix of size equal to the number of legs summed over all flights and is used to maneuver flights towards the preferred exit times. The right chunk of the left matrix has width equal to the number of conflicts, and is used to ensure flights are sufficiently separated, with -1 for flights that occupy the first position of the $k(\cdot, \cdot)$ function, and 1 for flights that occupy the second position. These values of 1 and -1 are matched to the terms containing λ if the current flight leg, or its previous associated legs, are involved in the conflict specified by λ. It is crucial to note that the first three rows, which are associated with flight 1, have the right chunk of the left matrix cumulatively summed upwards such that conflicts on later flight legs would cause changes in earlier legs. This is consistent with the intuition that the TTO of the current flight leg is dependent on all previous legs. For the right chunk, we also set the first row of each flight to contain only zeros, as we cannot change the value of the entry time. This will help us save computational time when we perform a gradient descent step.

With the matrices set up, we are now able to take advantage of the computationally efficient matrix multiplication operator `%*%` in R to obtain the partial derivatives with respect to β. The resulting vector will be of length equal to the number of legs summed over all flights. This vector *partial_b* will be multiplied by the step sizes η_{fj}, where we set

$$\eta_{fj}^{(i)} = \begin{cases} 0 & if\ j = 1 \\ \left(\underline{d}_{fj} - \overline{d}_{fj}\right)^{(i)} * \textit{partial\_b}^{(i)} & otherwise \end{cases}$$

where (i) represents the index of the vector. We have that $j = 1$ being indicative of an entry node with fixed time t_{f1} and that we are unable to modify this value. We also included the expression $\left(\underline{d}_{fj} - \overline{d}_{fj}\right)$ which was factored out from $\frac{\partial \mathcal{L}}{\partial \beta_{fj}}$ as placing more computations in the step size reduces the computational burden in the partial derivatives which have to be computed repeatedly at each iteration. For a design case

of $\sim 10^5$ conflicts, this method of using matrices has been tested to be faster than for-loops by a factor of $\sim 10^3$.

Finally, we provide an analysis of two methods to create the right chunk of the left matrix in Matrix A.1, given the table of conflicts described in the second paragraph of this section. We refer to the right chunk as *RightChunk* The first method is to use dplyr's $group\_by(flight\_id)\% > \%mutate(across(1 : ncol(RightChunk),\ revcumsum(.x)))$. This method is extremely sluggish since as it requires a datatype conversion of the sparse matrix into a dataframe and more critically, `across()` encodes a for-loop over all columns. The second and more efficient method is to first create a long dataframe that links each leg to all its previous legs and use this to create the sparse matrix. For example, assume we have a single flight path visiting 6 nodes, which implies this flight has 5 legs (equivalently betas) associated with it. The last node will be impacted by any changes in the 5 legs, hence has 5 rows, linking it to the t_id of all its previous legs. This linking process is then repeated, associating each t_id with all its previous legs. We can then `left_join()` the associated legs in Table A.1 with the table of conflicts and use this new longer conflicts table to create the sparse matrix.

From our tests, a significant amount of time is saved using the associated legs method. For a subset of ~ 1000 conflicts and 2500 flight legs, 3.5s is required to build the matrices using the dplyr method, as compared to 0.035s using the associated legs method.

Actual t_id	Associated t_id
1	1
2	2
3	2
3	3
4	2
4	3
4	4
5	2
5	3
5	4
5	5
6	2
6	3
6	4
6	5
6	6

TABLE A.1: Long Table for Associating the Current Leg with Previous Legs

Appendix B

Computing Flight Profiles

For the given equipment type, let r_A denote the rate of ascent and r_D the rate of descent, both in feet per second. Let a_C denote the cruising altitude in feet. Let $t_A = a_C/r_A$, the time taken to ascend to cruising altitude and $t_D = a_C/r_D$, the time taken to descend from cruising altitude, both in seconds. Let t_C denote the time spent at cruising altitude, after ascent and before descent and let $t_T = t_A + t_C + t_D$ denote the total flight time. It is possible on short trips that the aircraft does not reach cruising altitude before needing to begin its descent. We assume it continues to ascend until it reaches its peak altitude at which time it must begin its descent. Denote this altitude by a_M, in feet. Let $t_M = a_M/r_A$ denote the time taken to ascend to peak altitude and $t_R = a_M/r_D$ the residual time to descend from this altitude to complete the trip. In this case, $t_T = t_M + t_R$. Figure B.1 illustrates for the typical case in which a_M exceeds a_C.

Let $s(t)$ denote the speed of the aircraft over the course of its journey, in nautical miles per second and s_C the cruising speed in nautical miles per second (all time is measured in seconds). We assume that the rate of acceleration is constant during ascent and the rate of deceleration is constant during descent. In the typical case (that is, when $a_M \geq a_C$), that leads to the formula:

$$s(t) = \begin{cases} \frac{t}{t_A} s_C, & \text{if } 0 \leq t \leq t_A. \\ s_C, & \text{if } t_A < t \leq t_A + t_C, \\ \frac{t_T - t}{t_D} s_C, & \text{if } t_A + t_C < t \leq t_T. \end{cases}$$

In the atypical case ($a_M < a_C$), the aircraft does not reach its cruising speed at its peak altitude. Assuming constant acceleration, it reaches a speed of $s(t_M) = \frac{a_M}{a_C} s_C$. Consequently, the formula for $s(t)$ in the atypical case is given by:

$$s(t) = \begin{cases} \frac{t a_M}{t_M a_C} s_C, & \text{if } 0 \leq t \leq t_M. \\ \frac{(t_T - t) a_M}{t_R a_C} s_C, & \text{if } t_M < t \leq t_T, \end{cases}$$

Let $d(t) = \int_0^t s(u)du$, the distance covered in the time interval $[0, t]$, in nautical miles. Let $d_A = d(t_A)$ denote the distance covered during the ascent portion of flight, $d_C =$

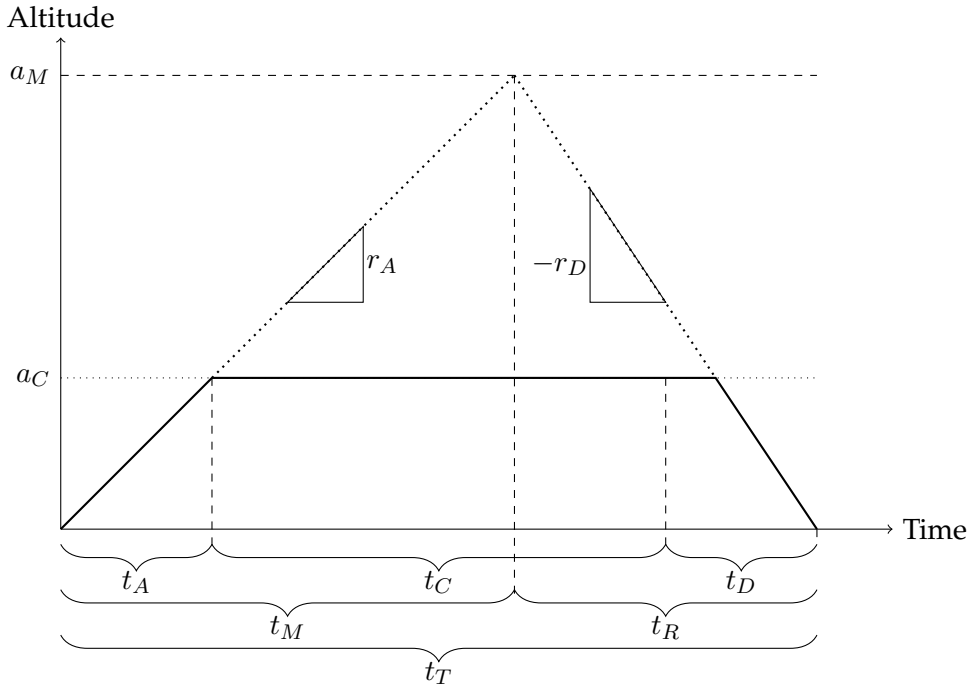

FIGURE B.1: Graph of Idealized Altitude *vs.* Time Profile of Typical Flight

$d(t_A + t_C) - d_A$, the distance covered during cruise, and $d_D = d(t_T) - d_A - d_C$, the distance covered during descent, in the typical case. In the atypical case, let $d_M = d(t_M)$ denote the distance covered until peak altitude a_M is reached and let $d_R = d(t_T) - d_M$ denote the distance covered during descent from a_M. Let $d_T = d(t_T)$ denote the total distance covered under either case ($d_T = d_A + d_C + d_D$ in the typical case and $d_T = d_M + d_R$ in the atypical case). The following results are easily verified:

Lemma B.0.1. *Under the assumption of constant acceleration and deceleration, the critical distances in the trajectory plan are given by:*

$$\begin{aligned} d_A &= \frac{a_C}{2r_A} s_C, \\ d_C &= t_C s_C, \\ d_D &= \frac{a_C}{2r_D} s_C, \\ d_M &= \frac{a_M^2}{2a_C r_A} s_C, \text{ and} \\ d_R &= \frac{a_M^2}{2a_C r_D} s_C. \end{aligned}$$

Proof. In the typical case, for $t \leq t_A$, we have:

$$d(t) = \int_0^t \frac{u}{t_A} s_C du = \frac{t^2}{2t_A} s_C.$$

Hence,

$$d_A = d(t_A) = \frac{t_A^2}{2t_A} s_C = \frac{t_A}{2} s_C = \frac{a_C}{2r_A} s_C.$$

Similarly,

$$d_D = \int_{t_T - t_D}^{t_T} \frac{t_T - u}{t_D} s_C du = \int_0^{t_D} \frac{x}{t_D} s_C dx = \frac{t_D^2}{2t_D} s_C = \frac{t_D}{2} s_C = \frac{a_C}{2r_D} s_C.$$

In the atypical case, for $t \leq t_M$:

$$d(t) = \int_0^t \frac{u a_M}{t_M a_C} s_C du = \frac{t^2 a_M}{2t_M a_C} s_C$$

Hence,

$$d_M = d(t_M) = \frac{t_M^2 a_M}{2t_M a_C} s_C = \frac{t_M a_M}{2a_C} s_C = \frac{a_M^2}{2a_C r_A} s_C.$$

Similarly,

$$d_R = \int_{t_T - t_R}^{t_T} \frac{(t_T - u) a_M}{t_R a_C} s_C du = \int_0^{t_R} \frac{x a_M}{t_R a_C} s_C dx = \frac{t_R^2 a_M}{2t_R a_C} s_C = \frac{t_R a_M}{2a_C} s_C = \frac{a_M^2}{2a_C r_D} s_C.$$

■

□

Let $\bar{D}$ denote the distance between an arbitrary origin-destination pair of airports. In the typical case, that would imply that $d_C = \bar{D} - d_A - d_D$. Hence,

$$\begin{aligned} d_C &= \bar{D} - \frac{a_C s_C}{2} \left(\frac{1}{r_A} + \frac{1}{r_D}\right) \\ &= \bar{D} - \frac{a_C s_C}{2} \left(\frac{r_D + r_A}{r_A r_D}\right) \end{aligned}$$

Provided $d_C \geq 0$, this leads to the following:

$$\begin{aligned} t_T &= t_A + t_D + t_C \\ &= \frac{a_C}{r_A} + \frac{a_C}{r_D} + \frac{d_C}{s_C} \\ &= a_C\big(\frac{r_D + r_A}{r_A r_D}\big) + \frac{\bar{D}}{s_C} - \frac{a_C}{2}\Big(\frac{r_D + r_A}{r_A r_D}\Big) \\ &= \frac{\bar{D}}{s_C} + \frac{a_C}{2}\Big(\frac{r_D + r_A}{r_A r_D}\Big) \end{aligned}$$

In the atypical case, a_M would need to satisfy the equation $d_M + d_R = \bar{D}$. Hence,

$$\begin{aligned} \frac{a_M^2}{2a_C r_A} s_C + \frac{a_M^2}{2a_C r_D} s_C &= \bar{D} \\ \frac{s_C}{2a_C}\big(\frac{1}{r_A} + \frac{1}{r_D}\big) a_M^2 &= \bar{D} \\ a_M^2 &= \frac{2\bar{D}a_C}{s_C}\Big(\frac{r_A r_D}{r_D + r_A}\Big) \end{aligned}$$

For a given origin-pair distance, $\bar{D}$, the critical points of the atypical flight trajectory are thus given by:

$$\begin{aligned} d_M &= \frac{\bar{D}}{r_A}\Big(\frac{r_A r_D}{r_D + r_A}\Big) \\ d_R &= \frac{\bar{D}}{r_D}\Big(\frac{r_A r_D}{r_D + r_A}\Big) \end{aligned}$$

and the total time is given by:

$$\begin{aligned} t_T &= t_M + t_R \\ &= \frac{a_M}{r_A} + \frac{a_M}{r_D} \\ &= a_M\Big(\frac{r_D + r_A}{r_A r_D}\Big) \\ &= \sqrt{\frac{2\bar{D}a_C}{s_C}\Big(\frac{r_D + r_A}{r_A r_D}\Big)} \end{aligned}$$

The condition of atypicality, $a_M \leq a_C$, is equivalent to the condition $d_C \leq 0$.

Proposition B.0.2. *Under the assumption of constant acceleration and deceleration, the distance-time profile $d(t)$ when the given origin-destination pair distance, $\bar{D}$ satisfies $\bar{D} \geq \frac{a_C s_C}{2}\Big(\frac{r_D + r_A}{r_A r_D}\Big)$ is given by:*

$$d(t) = \begin{cases} \frac{t^2 r_A}{2a_C} s_C, & \text{if } 0 \le t \le t_A. \\ d_A + (t - t_A)s_C, & \text{if } t_A \le t \le t_A + t_C. \\ \bar{D} - \frac{(t_T - t)^2 r_D}{2a_C} s_C, & \text{if } t_A + t_C < t \le t_T. \end{cases}$$

When the condition is not satisfied, the distance-time profile is given by:

$$d(t) = \begin{cases} \frac{t^2 r_A}{2a_C} s_C, & \text{if } 0 \le t \le t_M. \\ \bar{D} - \frac{(t_T - t)^2 r_D}{2a_C} s_C, & \text{if } t_M < t \le t_T. \end{cases}$$

Proof. As an example of the calculations, when $t > t_M$, we have:

$$\begin{aligned} d(t) &= d_M + \int_{t_M}^{t} \frac{(t_T - u)a_M}{t_R a_C} s_C du \\ &= d_M + \int_{t_T - t}^{t_T - t_M} \frac{x a_M}{t_R a_C} s_C dx \\ &= d_M + \frac{x^2 a_M}{2t_R a_C} s_C \Big|_{t_T - t}^{t_T - t_M} \\ &= d_M + \frac{(t_T - t_M)^2 a_M}{2t_R a_C} s_C - \frac{(t_T - t)^2 a_M}{2t_R a_C} s_C \\ &= d_M + \frac{t_R a_M}{2a_C} s_C - \frac{(t_T - t)^2 a_M}{2t_R a_C} s_C \\ &= d_M + \frac{a_M^2}{2r_D a_C} s_C - \frac{(t_T - t)^2 r_D}{2a_C} s_C \\ &= d_M + \frac{\bar{D}}{r_D}\Big(\frac{r_A r_D}{r_D + r_A}\Big) - \frac{(t_T - t)^2 r_D}{2a_C} s_C \\ &= d_M + d_R - \frac{(t_T - t)^2 r_D}{2a_C} s_C \\ &= \bar{D} - \frac{(t_T - t)^2 r_D}{2a_C} s_C. \end{aligned}$$

□

■

Let $t(d) = d^{-1}(d)$, the time in seconds required to traverse d nautical miles using the trajectory for the given origin-destination pair distance, $\bar{D}$.

Corollary B.0.3. *Under the assumption of constant acceleration and deceleration, the time-distance profile $t(d)$ when the given origin-destination pair distance, $\bar{D}$ satisfies $\bar{D} \ge \frac{a_C s_C}{2}\left(\frac{r_D + r_A}{r_A r_D}\right)$ is given by:*

$$t(d) = \begin{cases} \sqrt{\frac{2a_C d}{r_A s_C}}, & \text{if } 0 \leq d \leq d_A. \\ \frac{a_C}{r_A} + \frac{(d-d_A)}{s_C}, & \text{if } d_A \leq d \leq d_A + d_C. \\ \frac{a_C}{r_A} + \frac{d_C}{s_C} + \frac{a_C}{r_D} - \sqrt{\frac{2a_C(\bar{D}-d)}{r_D s_C}}, & \text{if } d_A + d_C < d \leq \bar{D}. \end{cases}$$

When the condition is not satisfied, the time-distance profile is given by:

$$t(d) = \begin{cases} \sqrt{\frac{2a_C d}{r_A s_C}}, & \text{if } 0 \leq d \leq d_M. \\ \sqrt{\frac{2\bar{D}a_C}{s_C}\left(\frac{r_D + r_A}{r_A r_D}\right)} - \sqrt{\frac{2a_C(\bar{D}-d)}{r_D s_C}}, & \text{if } d_M < d \leq \bar{D}. \end{cases}$$

Proof. For example, when $d_A + d_C \leq d \leq \bar{D}$, or when $d_M < d \leq \bar{D}$:

$$\begin{aligned} d &= \bar{D} - \frac{(t_T - t)^2 r_D}{2a_C} s_C \\ \frac{(t_T - t)^2 r_D}{2a_C} s_C &= \bar{D} - d \\ t &= t_T - \sqrt{\frac{2a_C(\bar{D}-d)}{r_D s_C}}. \end{aligned}$$

Substituting for the two possible values of t_T yields the result.

□

■

Figure B.2 illustrates the time-distance profile in the typical case ($a_M > a_C$) and Figure B.3 illustrates the time-distance profile in the atypical case when the aircraft does not reach cruising altitude before beginning its descent. Figure B.4 demonstrates how we compute minimum, maximum and preferred leg times (t_{min}, t_{max} and t_{pref}, respectively) for a flight leg connecting waypoints at nautical mile 120 and nautical mile 180. The aircraft starts at 0m, gradually climbs to cruise height and attains the cruising altitude at 120nm, cruises from 120nm to 180nm, and continuously descends from 180nm to 200nm until landing. We assume a maximum speed-up of 5% (scale=1.05) and a maximum slow-down of 10% (scale=0.90).

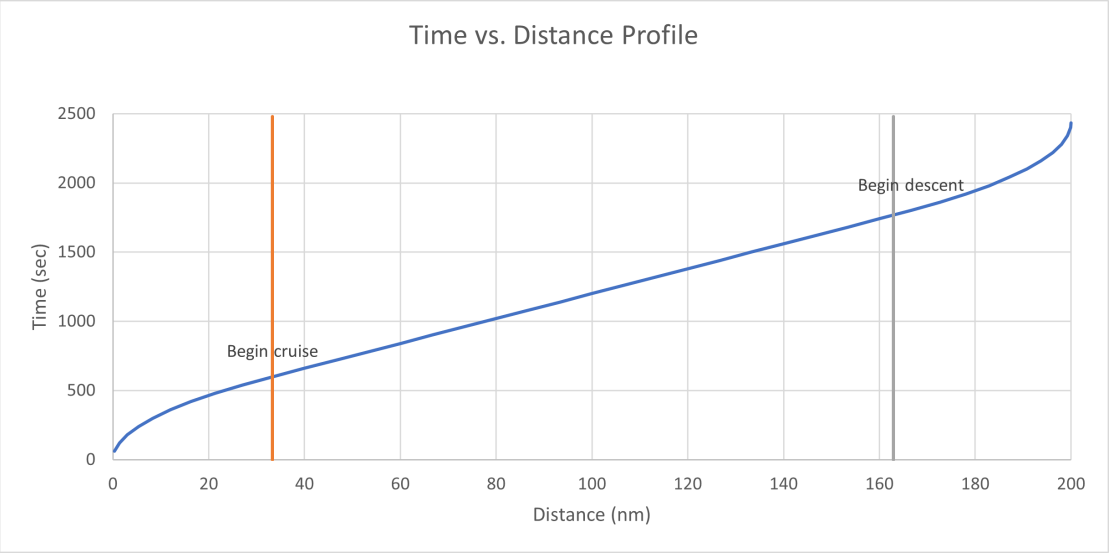

FIGURE B.2: Graph of Idealized Time vs. Distance Profile of Typical Flight

FIGURE B.3: Graph of Idealized Time vs. Distance Profile for Flight with No Cruise Portion

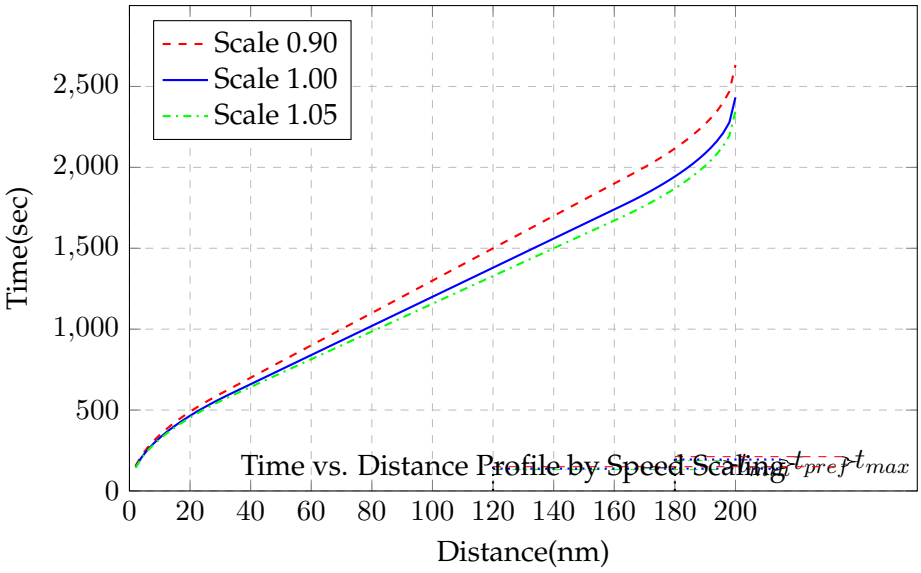

FIGURE B.4: Graph of Time vs. Distance Profile with Travel Time Estimates (t_{min}, t_{max},and t_{pref}). Cruising Occurs Over Distance Interval [120,180] Corresponding to Speed Scalings (Maximum: 1.05, Minimum: 0.90, and Preferred: 1.00), Respectively

Appendix C

System and User Interface Design for the FF-ICE Simulator

The Integrated Computer Aided Manufacturing Definition for Function Modeling (IDEF) techniques are a systematic method of describing and assessing the functions of a system. Technique 0 (IDEF0) provides a representation of the functionality of a system, the inputs, resources, and constraints associated with each function and the outputs produced.

This appendix chapter provides a textual description to further explain each IDEF0 diagram showing the high level user functionality of the software tool, "FF-ICE Regional Traffic Flow Simulator", along with screenshots from the FF-ICE simulator as an accompanying illustration of the UI design and tool functionality.

The purpose of the software tool, "FF-ICE Regional Traffic Flow Simulator", is to measure the impact of employing differing levels and extent of FF-ICE concepts in the South-east Asia region, specifically in the Association of South East Asia Nations (Brunei Darussalam, Myanmar, Cambodia, Indonesia, Laos, Malaysia, Philippines, Singapore, Thailand, and Vietnam) together with Flight Information Regions for Sanya and Hong Kong, collectively referred to in our documentation as "ASEAN Plus".

The focus of the impact measurement is on the comparison between operating under Release Level R0 and Release Level R1 of the FF-ICE concept of operations. It is assumed that all regions in ASEAN Plus are operating now at level R0. This is the minimal level of information sharing. Pre-flight flight plans are shared with all participating airports and flight information regions and scheduled departure times are known for all flights. For a flight information region to be at Release Level R1, the key additional information which will be shared is the actual departure time of flights originating in that region. The benefit of this information is that if a flight has a delayed departure time, the destination airport can update its place in the arrival queue and potentially make better use of airport capacity. The second feature of Release Level R1 is that it can facilitate collaborative ground delay programs through the sharing of Calculated

Take-off Time (CTOT) for each flight from a destination airport to collaborating departure airports. Thus, if an airport is experiencing high delays, with approaching aircraft likely to spend time in holding areas, that information can be shared with participating airports through CTOT's. The departure airports can then delay flights accordingly and thereby substitute ground holding time for airborne holding time.

The software tool is constructed to allow the user to identify which flight information regions are operating at each Release Level (R0 or R1) and to simulate air traffic under the different information sharing rules. The analyst can then measure the benefit in reduced airborne delays of employing more advanced information and collaborative tools. It will be interesting to measure these benefits at the region level, at the individual flight information level, and at the level of important city pairs.

NETSIM Context

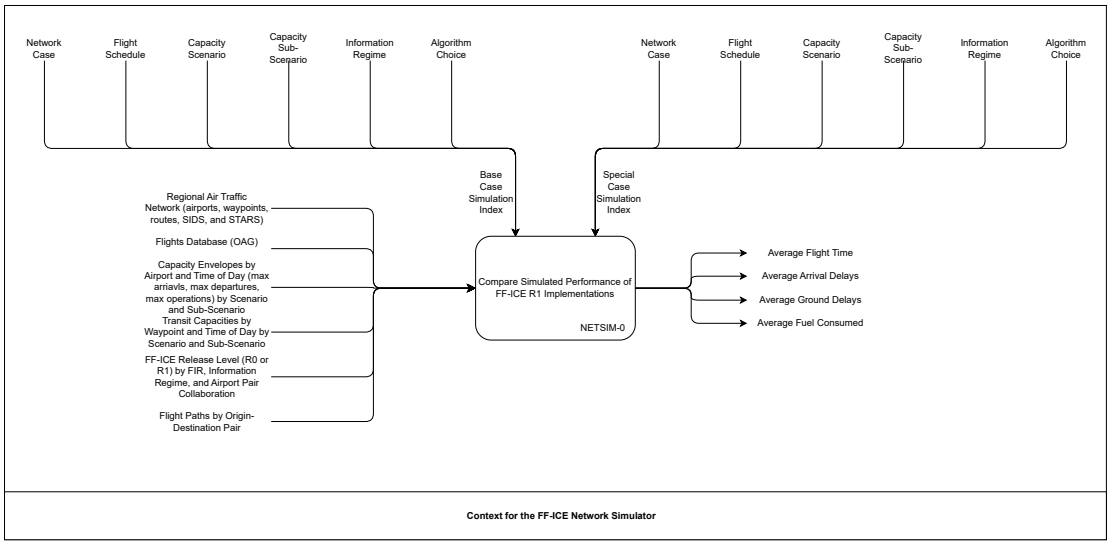

FIGURE C.1: Context for the FF-ICE Network Simulator

The context of the FF-ICE Regional Network Simulator is the use case in which an analyst seeks to "Compare Simulated Performance of FF-ICE R1 Implementations", shown in the Figure C.1 as function numbered NETSIM-0. The two user inputs for such a comparison are the simulation index for a Base Case and a simulation index for a Special Case. A simulation index uniquely identifies a detailed collection of choices to describe the actual simulation to be run. We assume the focus will be on comparing information regimes. Consequently, most choices, except for Information Regime, will be the same for both Base Case and Special Case. These choices are listed as:

- *Network Case*: If there are alternative network designs available, the network case ID identifies which network design to use in the simulation. This will likely be the same for both Base Case and Special Case.

- *Flight Schedule Selection*: The user can specify the starting date and time to select flights from the database of historical flights. Flights will be loaded into the simulator in batches starting from this date and time. As the simulation progresses, it loads in consecutive batches so that all simulated decision makers (airports and en route controllers) have an adequate amount of look-ahead information. This will likely be the same for both Base Case and Special Case.

- *Capacity Scenario*: The tool comes pre-loaded with different capacity scenarios for airports and waypoints. The user indicates which scenario to use through the Scenario ID. This will likely be the same for both Base Case and Special Case.

- *Capacity Sub-Scenario*: The tool editor allows the user to define sub-scenarios in which he or she specifies specific events such as waypoint closure or airport restrictions which further restrict capacity beyond the chosen Capacity Scenario. The user must then specify which sub-scenario to use in the simulation. This sub-scenario will likely be the same for both Base Case and Special Case.

- *Information Regime*: The tool editor allows the user to define information regimes in which each Flight Information Region (FIR) is set separately to operate at either Release Level R0 or Release Level R1. For example, one information regime may be to set all regions at R0, another information regime may be to set all regions at R1, and a third regime may be to set just a subset of the regions at R1. Additionally, the analyst may further refine the information regime by only setting particular airport pairs at R1 within the set of regions at R1. It is very likely that the analyst will then select different information regimes for the Base Case and Special Case. Typically, the Base Case would be an information regime in which all regions are at R0.

- *Algorithm Choice and Parameters*: The developer uses this tool to conduct experiments on different optimization algorithms or collaborative ground delay programs. Certain parameters may also need to be tuned for optimum performance of the algorithms. The different possibilities are captured by means of an Algorithm Choice ID. The typical user will use the choice recommended by the developer.

The tool comes with pre-built databases of air traffic simulation inputs including the following:

- *Regional Air Traffic Network*: The air traffic network consists of nodes and arcs representing airports, waypoints, routes, Standard Instrument Departure Routes (SIDs) and Standard Arrival Routes (STARS). This database has been developed by researchers at ASI through reference to the Aeronautical Information Publication (AIP) documents for each FIR.

- *Flights Database (OAG)*: The flights database is derived from access under license to historical flight schedules maintained by OAG. It covers a multi-day block of time in late 2023. It includes flights originating and/or terminating in the ASEAN Plus region as well as some overflights. Multi-stop flights have been transformed into sequences of single-leg flights to simplify the simulation. A small number of historical flights were deleted from the database because they were found to be incompatible with the Regional Air Traffic Network (for example, if they referenced a newly opened airport).

- *Capacity Envelopes and Waypoint Capacities*: This database includes different scenarios of network capacity constraints. An airport capacity constraint is captured as a so-called Capacity Envelope (maximum arrivals per hour, maximum departures per hour, and maximum total operations per hour). A Waypoint Capacity is simply the maximum number of aircraft transits permitted per hour at that waypoint. Each scenario provides a Capacity Envelope for every airport and a Waypoint Capacity for every waypoint. Some scenarios are designed to be more restrictive than others in order to test the conditions under which FF-ICE information sharing becomes valuable (it is not likely to be of value when capacity is high, that is, when demand is much less than supply).

- *FF-ICE Release Levels*: Researchers at ASI have created a number of pre-built information regimes. For each regime, the Release Level (R0 or R1) is specified for each Flight Information Region. The user may use any of these regimes in a simulation by specifying its Information Regime ID. The user may also define new regimes for investigation by simulation.

- *Flight Paths by Origin Destination Pair*: For every origin-destination pair found in the Flights Database, researchers at ASI have constructed flight plans (waypoint sequences) through the Regional Air Traffic Network from origin to destination. We used the online rFinder tool to construct these flight plans, resorting to SkyVector in problematic cases. In cases where the origin or destination lies outside the ASEAN Plus region, we connected the external location directly to the first or last waypoint in the flight plan which belongs in the Regional Air Traffic Network.

With the exception of FF-ICE Release Levels, the software tool does not provide any facility to modify these pre-built databases. These data are difficult to acquire, assemble and validate. Alternatively, the database formats are easily accessible through tools such as dBeaver and the advanced user can modify the data directly.

The output of the comparison is a paired matching of completed simulated flights between the Base Case and the Special Case. Any two simulations can be compared but the Use Case assumes the analyst is using a Base Case in which the information regime has all regions at Release Level R0. By focusing on completed flights that are matched between the two simulations, we get statistics that can be broken down to any level of detail desired: system-wide, regional, or city-pairs. The comparison ignores flights that are completed under one simulation but not completed under another. It is important therefore to ensure that simulations are run long enough so that these mismatched flights do not affect the long-term averages by much. The analyst can extend simulation runs until the averages stabilize.

For each pair of completed flights we know the scheduled departure time, the preferred flight time (based on equipment type and distance), the simulated departure time, and the simulated flight time. The simulated flight time may include aircraft speed-up, slow-down, and/or airborne holding. For each simulation run and for each aggregation level (system, flight information region, or city-pair), the software reports the following key statistical measures:

- *Average flight time*: This is the total flight time across all flights in the aggregation group divided by the number of flights.
- *Average arrival delays*: this is the difference between the simulated arrival time and the scheduled arrival time (taken to be the scheduled departure time plus the preferred flight time) averaged over all flights in the aggregation group.
- *Average ground delay time*: This is the difference between the simulated departure time and the scheduled departure time averaged over all flights in the aggregation group.
- *Average fuel consumed*: This is the difference between the simulated fuel consumed over all flights in the aggregation group.

If Release Level R1 provides a benefit, we would expect to see lower average flight times and fuel consumed under the Base Case (R0) and a Special Case (some FIRs using R1) but greater ground delays under the Special Case, and possibly larger arrival delays.

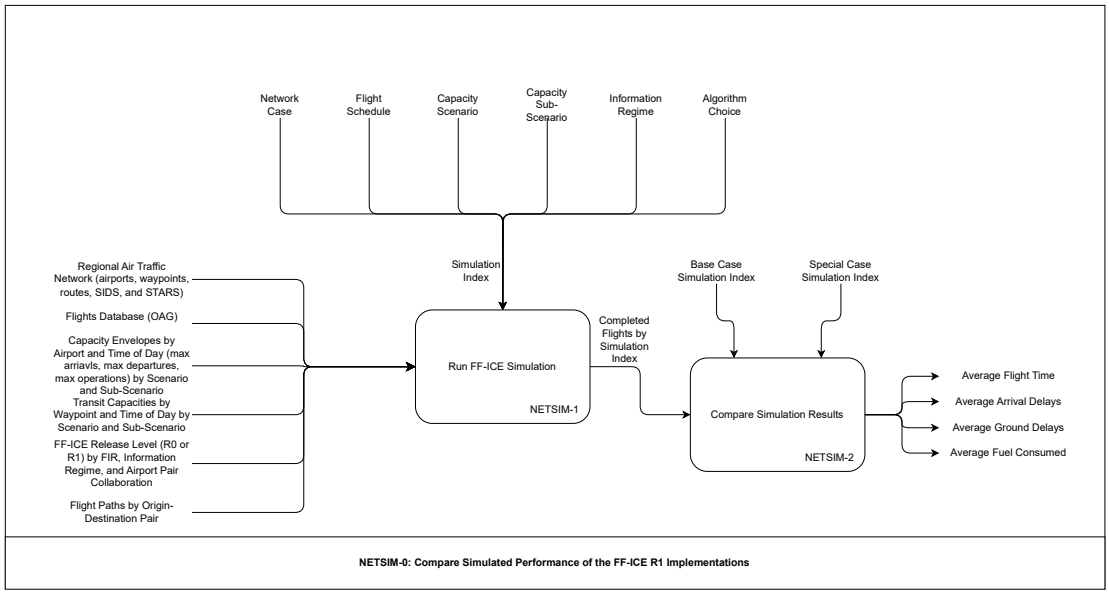

FIGURE C.2: NETSIM-0: Compare Simulated Performance of the FF-ICE R1 Implementations.

NETSIM-0

The diagram NETSIM-0 expands the function "*Compare Simulated Performance of Regional FF-ICE R1 Implementations*" to identify two distinct functions which make up the use case. The function "Run FF-ICE Simulation" labeled as NETSIM-1 captures the essential role of building and running simulations. We think of this as a "Designer" role because it requires specifying the combination of Network Case, Flight Schedule Selection, Capacity Scenario, Capacity Sub-Scenario, Information Regime, and Algorithm Choice and Parameters which make up the unique simulation index as explained in the previous section. The inputs to running the simulation are the Regional Air Traffic Network, Flights Database (OAG), Capacity Envelopes Scenarios and Sub-Scenarios, Transit Capacities Scenarios and Sub-Scenarios, FF-ICE Release Levels, and Flight Paths are as described in the previous section. The Designer also runs the simulation and chooses how long to run the simulation. After each simulation run, the state of the simulation is saved. The Designer can choose to extend the simulation run for as many time steps as desired. The simulator software will read in the previously saved state of simulation and continue from that point. The Designer can also delete simulation indices and all recorded results for those indices. This can be useful if a mistake was made in setting up, say, the capacity sub-scenarios.

The second function of the use case is "Compare Simulation Results" labeled as

NETSIM-2. We think of this as an "Analyst" role because it enables studying the statistical comparisons of paired simulation runs. The outputs of this function are as described in the previous section. The only new input captured in this diagram is:

- *Completed flights by simulation index*: Each simulation archives completed flights into the database by simulation index, and is retrieved when comparing the results of two simulations.

NETSIM-1

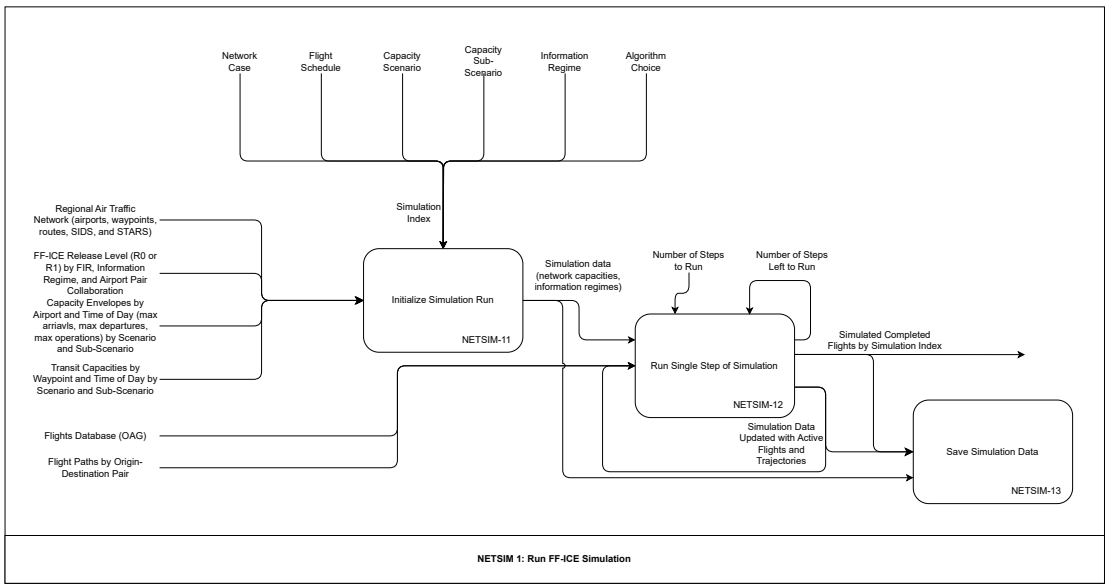

FIGURE C.3: NETSIM 1: Run FF-ICE Simulation

The diagram NETSIM-1 expands the function "Run FF-ICE Simulation" to capture three major functions of the software simulator:

- **NETSIM-11: Initialize Simulation Run**: This function deletes any prior record of the simulation run for the given simulation index, resets the simulation clock to the start of the flight schedule, and loads all data relevant to the simulation as determined by the simulation index and the network, capacities, and information regime choices which the index represents.

- **NETSIM-12: Run Single Step of Simulation**: This function advances all simulated flights along their flight paths making such decisions as sequencing and releasing aircraft for departure or holding them for ground delay, adjusting the speed of aircraft en route and perhaps putting them into holding patterns, sequencing aircraft for arrival, and marking them as completed when landed. The

step size of this advance is currently set at 5 minutes. This function is described in more detail in diagram NETSIM-12 below, in Section C.

- **NETSIM-13: Save Simulation Data**: The simulation data captures all information about the current state of the simulation (what flights are active, what stage of preparation or flight they are in, and what their current locations and speeds are). It is saved after each simulation step so that the simulation can be interrupted, and then reloaded and continued at a later time.

The major outputs of NETSIM-12 are twofold:

- *Simulation Data Updated with Active Flights and Trajectories*: As described above, the simulation data captures all information about the current state of the simulation. These data are saved so that the simulation can be interrupted and continued later.
- *Simulated Completed Flights by Simulation Index*: When a simulated flight lands, it is marked as completed and removed from the simulation data. It is archived into a separate database table for use in subsequent analysis, as in NETSIM-2.

A minor output of NETSIM-2 is the Number of Steps Left to Run which counts down from Number of Steps to Run to allow the user to let the simulation run many steps at once. The software UI displays this countdown number so that the user can monitor progress of the simulation.

NETSIM-12

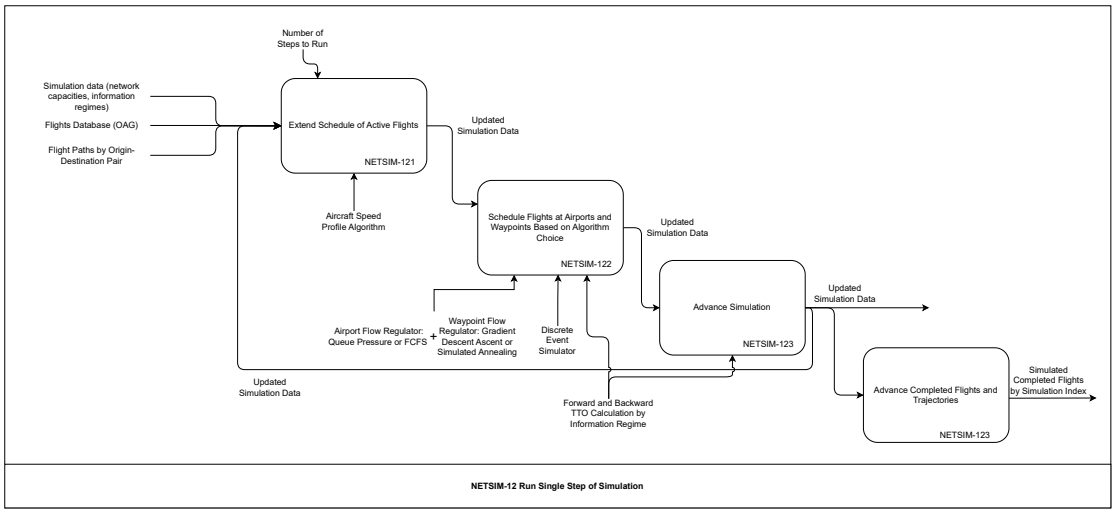

FIGURE C.4: NETSIM-12 Run Single Step of Simulation

Diagram NETSIM-12 expands the function "Run Single Step of Simulation" to reveal the basic sequence of functions used to implement the simulation. The inputs and outputs of these functions are the same as documented in the earlier diagrams. What this diagram adds is the identification of specialized algorithms used by the different functions. These algorithms are briefly described as follows:

- **Aircraft Speed Profile Algorithm**: Based on the equipment table, with rate of ascent, rate of descent and cruising altitude, this algorithm computes a speed and altitude profile of an aircraft for any given trip distance. If the trip distance is short, the aircraft may not reach its cruising altitude before beginning its descent. The speed-distance profile is adjusted up and down in percentage fashion to determine, for each leg of the flight plan, the minimum, maximum and preferred time to traverse the leg. We refer the reader to Appendix B for the detailed calculation procedures.

- **Airport Flow Regulator (AFR)**: This algorithm sequences aircraft into/out of an airport based on arrival and departure queues of flights for that airport and its current Capacity Envelope (maximum arrivals per hour, maximum departures per hour, and maximum operations per hour). It does this by converting the Capacity Envelope into a three-channel representation of the airport (an arrivals-only channel, a departures-only channel, and a common channel). Each channel can be treated like a waypoint with a fixed minimum separation time. The algorithm schedules each flight into the earliest available channel time for that flight. It chooses the next flight to schedule from the queue (Arrival or Departure) with the highest so-called queue pressure. Queue pressure is simply the sum of incremental delays flights from the queue will experience if the next flight is chosen from that queue. If the FCFS algorithm is chosen, the next flight to schedule will be the earliest available flight that is permitted to use the selected channel.

- **Waypoint Flow Regulator (WFR)**: This algorithm modifies the Target Time Over (TTO) of each future leg of the trajectory of each aircraft in a FIR to satisfy separation constraints at the FIR waypoints. The TTO of past legs are fixed and irrelevant. The TTO of the current leg is also fixed under the assumption that it is too late to adjust the speed of an aircraft on its current leg. The algorithm uses speed changes and vectoring/holding to adjust these future TTOs. Separation times are derived from the Waypoint Capacity Scenario and Sub-Scenario. The objective is to minimize the deviation of exit TTOs (the TTO of the last leg of each flight in the FIR) from the backTTO. The user may select between the Gradient Descent Ascent (GDA) or Simulated Annealing (SA) algorithm. The optimization problem is solved using the selected algorithm. Neither optimality nor feasibility

are guaranteed by this algorithm, but the results are promising, often generating conflict-free trajectories.

- **Forward and Backward TTO Calculation by Information Regime**: This algorithm computes a TTO for the endpoint of each leg in the trajectory for each active flight. If the origin FIR for the flight is at Release Level R0, then the calculation uses the scheduled departure time from the origin airport plus the cumulative sum of preferred leg times (time to traverse the leg at its preferred speed). If the origin FIR for the flight is at Release Level R1 and the leg is in a FIR at Release Level R1 (or if the flight is in the FIR), then the TTO is based on the actual (simulated) departure time from the origin airport plus the cumulative sum of actual leg times. In this way, the airport scheduling algorithm will be using information appropriate to the information regime in place. Similarly, the algorithm will compute backwards from an arrival time using the reverse cumulative sum of preferred leg times to arrive at a backTTO for each flight leg. This backTTO is used by the Waypoint Scheduler to keep flights on schedule. It is also used in the Airport Scheduler as a CTOT. If all regions are at Release Level R0 then the backTTO is calculated from the scheduled arrival time (which we set equal to the scheduled departure time plus preferred trip time). So under R0, there is no difference between the scheduled departure time and the CTOT. No collaborative ground delays are possible. However, if both origin and destination FIRs are at level R1, then the CTOT at the origin airport will reflect the current arrival plan for the destination airport. In this way, the airports can collaborate, where the origin airport can delay a departure until the CTOT.

The functions needed to execute a single step of the simulation are as follows:

- **NETSIM-121: Extend Schedule of Active Flights**: In this function, the software simulator checks to see if there is an adequate sequence of flights in the future available for look-ahead decisions (such as arrival and departure sequencing). If needed, it fetches the next batch for scheduled flights from the flights database and matches them with pre-calculated flight paths for origin-destination pairs to create a flight plan for each flight. The Aircraft Speed Profile Algorithm is then used to compute a preferred speed for each leg of each flight together with minimum and maximum speeds. These are added to the simulation data in the form of Active Flights and Trajectories.

- **NETSIM-122: Schedule Flights at Airports and Waypoints Based on Algorithm Choice — WFR+AFR**: In this function, the software simulator considers each airport in the region, one by one, and updates a dynamic schedule of arrival and departure times for each active flight arriving or departing at that airport. The

information available about each flight depends on the information regimes of origin and destination FIR. For example, if both FIRs are at level R1 then the departure airport will know the CTOT as calculated by the arrival airport and the arrival airport will know the updated departure time from the departure airport. If either FIR is at level R0, then both airports will know only the original schedule information and no updates in either departure time or CTOT. This function makes use of either the Arrival and Departure Queue Pressure or FCFS Algorithm. It also uses the Forward and Backward TTO Calculation by Information Regime. For the WFR component, the software simulator considers each FIR in the region, one by one, and updates the portion of the trajectory of each active flight which intersects that FIR. The chief concern of the WFR is to ensure that no two flights arrive at any node in the FIR too close in time to each other. The required minimum separation times are derived from the Waypoint Capacity Scenario and Sub-Scenario for this simulation. It accomplishes this separation by adjusting planned flight speeds along each future leg of the flight plan (we assume that the speed on the current leg and previous legs are fixed). If changes in speed are not sufficient to resolve separation conflicts, the scheduler can resort to vectoring or holding activities. The WFR also attempts to keep the flight on schedule by scheduling the TTO of the last leg endpoint as close as possible to the target TTO for that waypoint. This optimization problem is solved using the GDA or SA Algorithm. The WFR is also used to fine-tune the arrival and departure times to make them consistent with en route trajectories. Like the Airport Scheduler, it also uses the Forward and Backward TTO Calculation by Information Regime.

- **NETSIM-122: Schedule Flights at Airports and Waypoints Based on Algorithm Choice — DES**: In this function, the software considers the same constraints as the AFR+WFR, and schedules aircraft trajectories, with a minimum separation observed between all flight pairs using the same resource. This algorithm also decides on the choice of channels at airports. In the main loop of the DES, we process all events in non-decreasing order of time. For flights subject to collaborative ground delays, we employ a First Scheduled, First Served (FSFS) algorithm, where flight events are processed in order of their scheduled time, rather than the time of which they arrive at a given node. Events may refer to either flight events or resource events, and a resource refers to a unique node-channel pair. Flight events first assigns a TTO to the current flight leg that is being processed, and subsequently adds a resource event to mark when the node will next be available to schedule any pending flight legs. Finally an event is added to mark the next leg in its flight path as pending, if any. Resource events signify times at which a resource is available. An event indicating a specific resource is available, will

trigger a check to assign the next available flight to this resource, if any. At the end of processing all events, the DES will have generated conflict free TTOs for all flights within the current simulation step. These TTOs uniquely determine the time between any two nodes in a flight path, and the difference from the preferred leg time is first assigned to a decrease in speed, and any remaining difference to an extraordinary hold time. The output format of the DES matches that of the AFR+WFR, and either algorithm can be chosen without any modifications to the data format or subsequent analysis.

- **NETSIM-124: Advance Simulation**: This function advances the simulation clock by one time step (currently set at 5 minutes). Each flight is checked and updated as to whether its stage of flight will change by the end of that time step. The stages of flight are:
  - Not Ready: The aircraft is not available for departure until its scheduled departure time or CTOT, whichever is later;
  - Ready: The aircraft is available for departure;
  - Pushback: The aircraft has a committed departure time in the future;
  - En Route: The committed departure time has passed so the aircraft is in the air en route to its destination but without a committed arrival time;
  - Approach: The aircraft is near the destination airport but has not been given a committed arrival time;
  - Final: the aircraft has a committed arrival time in the future; and
  - Land: The committed arrival time has passed so the aircraft has landed. It becomes a completed flight at this point.
- **NETSIM-125: Archive Completed Flights and Trajectories**: This function removes completed flights from the simulation data (to conserve computer memory) once they are sufficiently far in the past to not violate separation constraints between any active and future flights. These removed flights will also be archived for subsequent analysis. Two tables are maintained: the flights table (one row per completed flight) and the trajectories table (one row per leg of each completed flight).

User Interface Design

We envision three key user categories for the FF-ICE simulator — Administrator, Analyst, and Designer.

The administrator would be the user that sets up the FF-ICE simulator. The administrator will first install the necessary software, including R and MySQL. For our partnering associations, a user manual is also provided to guide the user through the setup process. After setting up the required software and user permissions, the administrator may then utilize the administrator app, as illustrated in Figure C.5, which contains three main tools. The first tool *Check Database Connections*, verifies if the installation and app and database credentials are correctly set up. The next is a tool that initializes all databases based on a default empty database file, which is provided together with the FF-ICE simulator and should not be modified. Finally, the administrator may also choose to restore the database from a database file, if prior runs had been conducted and a database file was shipped together with the software or saved by a previous user. In most cases, the latest or most significant results would have been showcased to our partners, and the runs of which the results are based on, will be included as a supplementary database file that can be used with the *restore database* tool to copy the data onto the current computer.

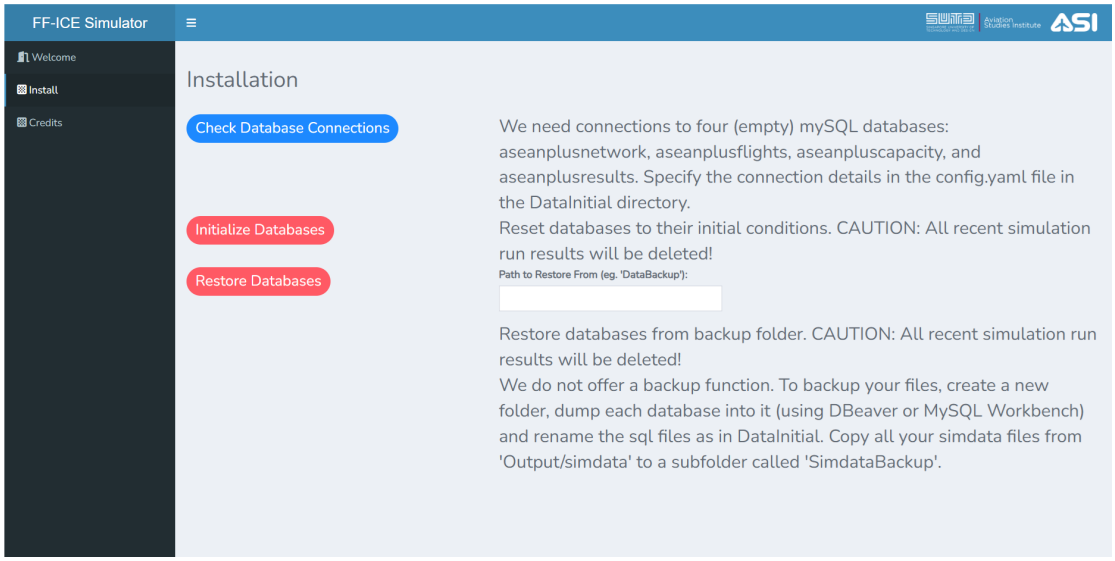

FIGURE C.5: Administrator App: Database Set Up Screen

Next, the designer would be the user that creates and runs simulations. The system definition of this user is captured in Figure C.3, where the designer will specify the combination of Network Case, Flight Schedule Selection, Capacity Scenario, Capacity Sub-Scenario, Information Regime, and Algorithm Choice to create a new simulation index with constants dictating the options selected. The UI for selecting the choices are given in Figure C.6, where the designer can use the drop down lists to select an option for each choice. We follow the order of arrows in Figure C.3, and go from left to right, top to bottom on Figure C.7, explaining the possible processes to be run on the designer

app — *Run Simulation*. The designer first selects the simulation index to perform runs on, and from here, the designer may click on *Initialize/Reset the Simulation*, which deletes all prior runs corresponding to the simulation index, if any. Thereafter, the designer can input the number of steps to run, each step corresponding to 5 minutes of simulated time, and click *Run Simulation* to start the runs for the selected simulation index. The *Run Simulation* button initiates a procedure that is described in the *NETSIM-12* process. The designer also has the option to *Select history detail level*, where a .rds file will be saved for each simulation step if *Complete history* is selected, and only for the final simulation step otherwise. The .rds files are intended primarily for debugging purposes by the ASI team in the event of application malfunctions, and should be disregarded by all other users. Finally, the designer has the liberty of terminating the runs prematurely by clicking *Stop Simulation*, which will stop the runs once the current simulation step is completed. The runs may be stopped and continued at any time. No data loss will occur, nor will there be any difference in results regardless if the runs are completed in single session or across multiple batches.

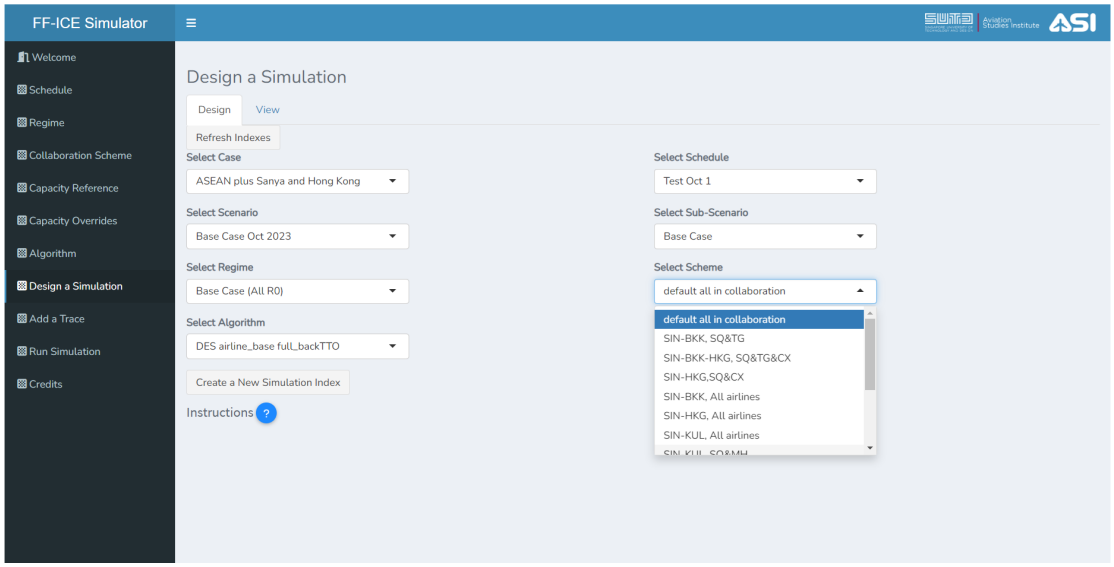

FIGURE C.6: Designer App: Choice of Design Settings To Initialize a Simulation

Once multiple runs have been completed, the analyst will be able to compare two simulations, with the main inputs and outputs depicted in the system definition diagram in Figure C.1, and draw analyses and insights from the reported results. The FF-ICE simulator also allows the analyst to generate various plots and diagrams to further understand the results. First, the analyst will select two simulation indices, of which is most often aligned on every setting except for the information sharing scheme, labeled *regime* and *scheme* in Figure C.8. Also in Figure C.8, clicking the *compare* button

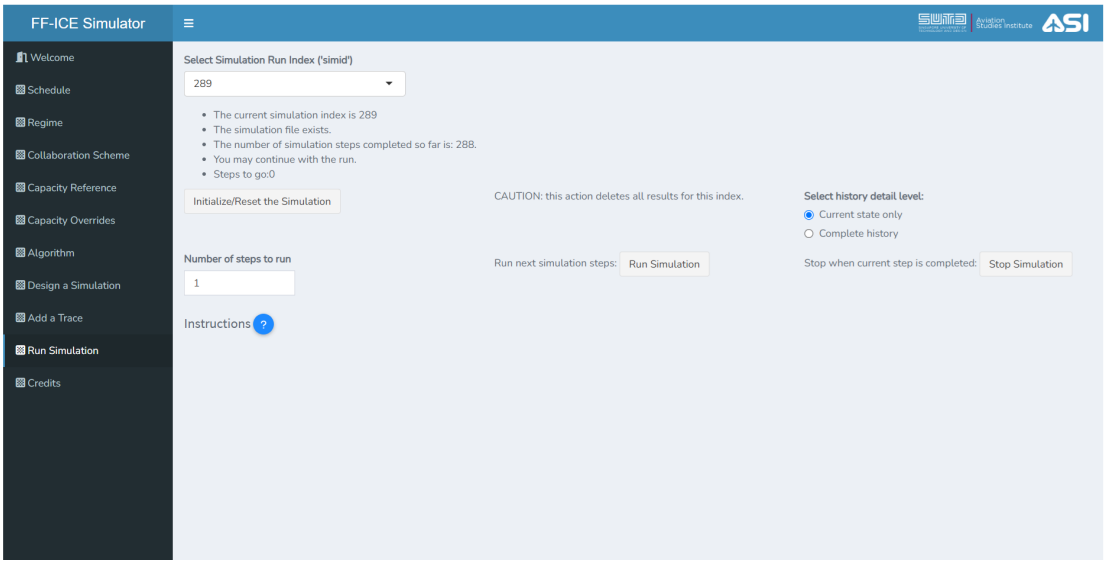

FIGURE C.7: Designer App: Tool For Running and Saving Simulations

will identify flights that are present in both simulations, and compute the comparison statistics for these flights, as shown in the bottom of the figure. Here, there are more columns in the results comparisons than in Figure C.1. However, the extra columns are not used in the results presented earlier in this paper, nor in any of our analyses, hence have not been included in the the NETSIM diagram. Upon scrolling further down, the analyst will be presented with more diagrams and analytic tools to take a more detailed look at the results. The next tool in the dashboard is the histogram in Figure C.9, which provides a visualization of how fuel savings, and number of flights vary throughout the simulation period (24h in this example). The analyst may use the drop down list from *Select Number of Hub Airports* to select how many of the top airports, by number of flights, that should be plotted with a different color. Further down in the analyst app, the analyst would find an option to compute the comparison statistics similar to Figure C.8, but computed only for the selected airport. Also, a bird's-eye view of which airports benefit from or are disadvantaged by the FF-ICE R1 initiative, as illustrated in Figure C.10, is provided. In this map plot, the green lines represent time savings under the Special Case, while red lines represent negative time savings, i.e. increased delays. The thickness of the lines reflects the magnitude of savings, with thicker lines corresponding to increased savings. Only airports pairs which have significant savings or delays, whose sum of absolute values is greater than the mean absolute savings, have lines plotted between them.

The next tool, the TTO vs Scheduled R0TTO Graph, in Figures C.11 and C.12, starts with the analyst selecting a flight, which will generate the TTO vs Scheduled R0TTO Graph, with the TTO on the y-axis and R0TTO on the x-axis. The dotted line is the

baseline result, which represents the expected plot if the flight has zero delays, where the TTO on every flight leg matches its scheduled time exactly. When a line lies above the dotted line, it indicates that the flight has been delayed, and the TTO is later than the scheduled R0TTO. In Figure C.11, we observe that the turquoise line, representing the Special Case (FF-ICE R1 in this example), is higher than the red line, and this delay started from the first node on the left, implying that it was assigned a ground delay. Figure C.12 provides a close up view of the last node in Figure C.11. Here, we see a perfect overlap of the turquoise and red line at WADD, which reveals a successful ground delay, where the ground delay on the turquoise flight led to a later departure, with no change in arrival time, from which the analyst can conclude that a decrease in flight time was observed.

The last analyst app in Figure C.13 provides a visual representation of the entire flight trajectory of a selected flight. The horizontal length of the bars in this plot represent the required separation time, where a longer bar indicates a larger required separation time, and the block of time that has been reserved for a particular aircraft leg. The colors indicate whether a flight has been completed, i.e., archived and no longer under consideration by the optimization algorithms, or active. Also, the additional image layer of *GDP Flights* can be toggled on or off by clicking anywhere within the bounding box of the red bar and *GDP Flights* on the legend. To give an example of how to interpret this graph: the analyst could follow the trajectory of the selected flight by focusing on the black bars, and perhaps observe changes in flight patterns of the selected flight itself, or any adjacent flights to the selected flight under the Base Case and Special case.

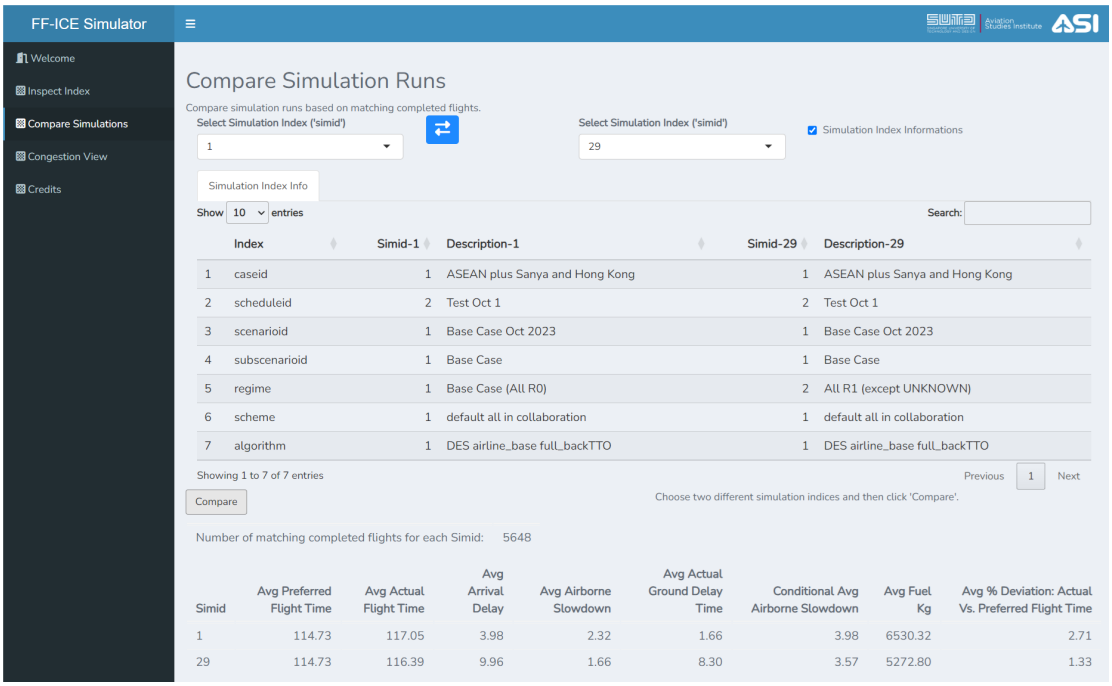

FIGURE C.8: Analyst App: Simulation Index Selection Screen

FIGURE C.9: Analyst App: Histogram Displaying Savings at Individual Airports

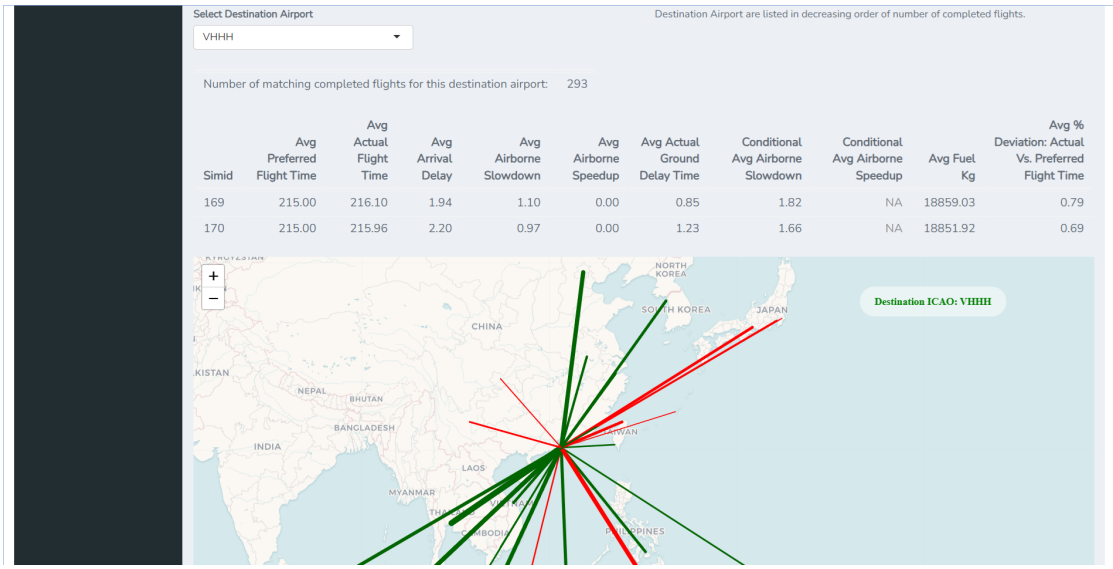

FIGURE C.10: Analyst App: Map of Airborne Savings of Flights Arriving at VHHH (Hong Kong International Airport)

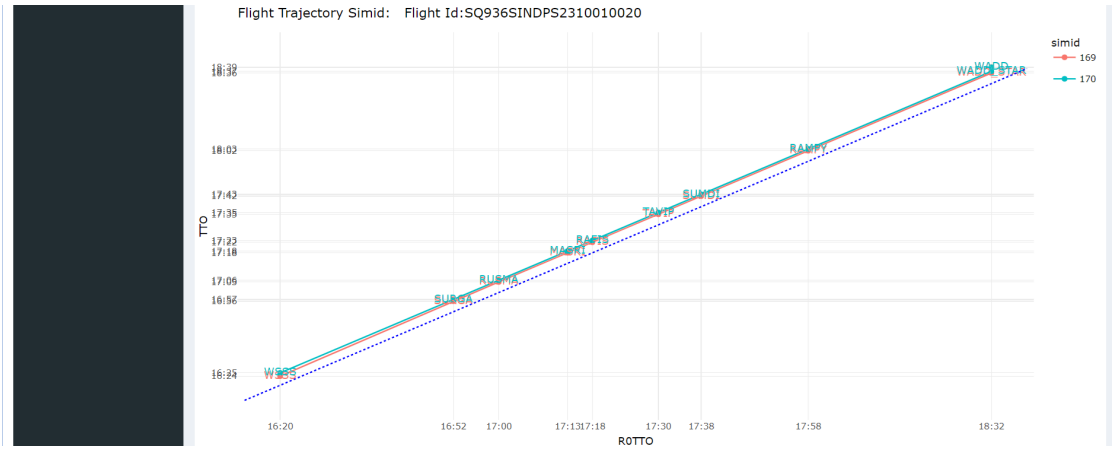

FIGURE C.11: Analyst App: TTO vs Scheduled R0TTO Graph

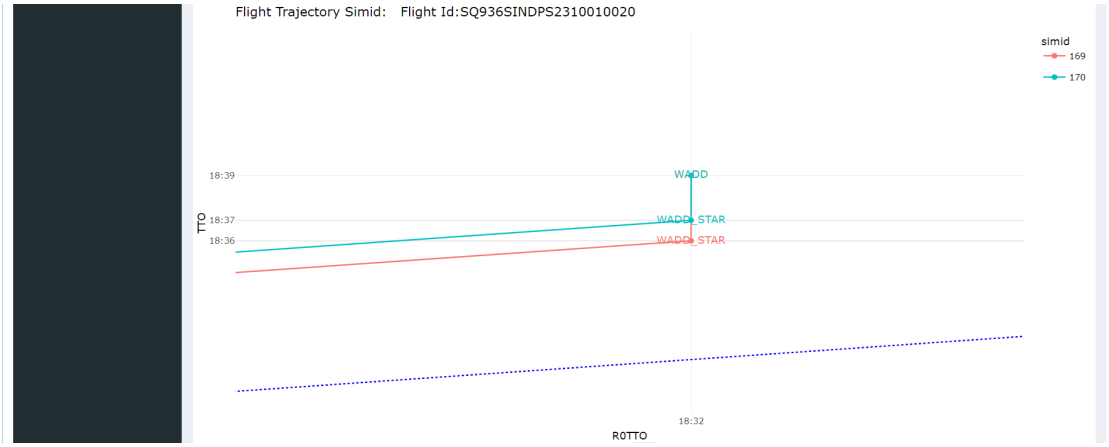

FIGURE C.12: Analyst App: A Zoomed-in View of TTO vs Scheduled R0TTO Graph

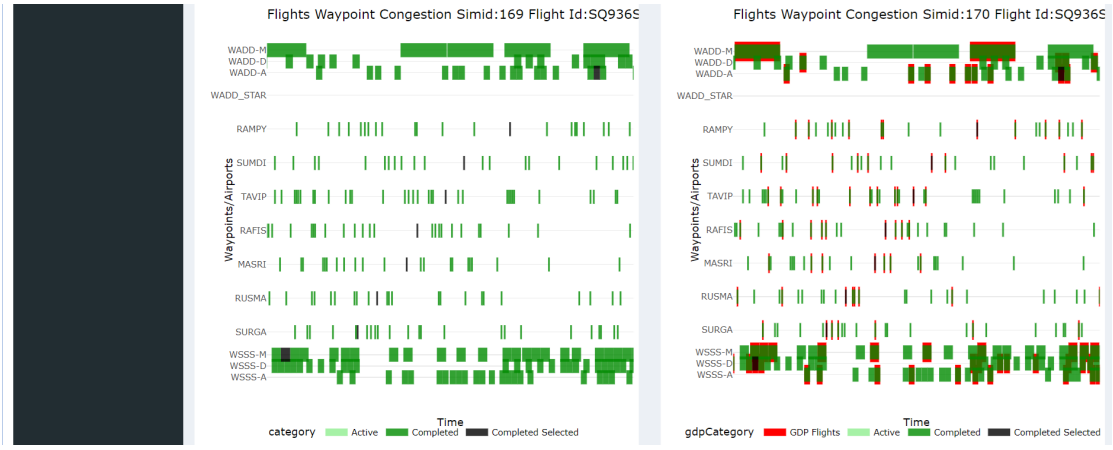

FIGURE C.13: Analyst App: Flight Trajectory Graph

Bibliography

Aarts, Emile H. L. and Jan Korst (1997). *Simulated annealing and Boltzmann machines. A stochastic approach to combinatorial optimization and neural computing*. Reprinted. Wiley-Interscience series in discrete mathematics and optimization. Chichester [u.a.]: Wiley. 272 pp. ISBN: 978-0-471-92146-2.

Adams, Joseph, Egon Balas, and Daniel Zawack (1988). "The Shifting Bottleneck Procedure for Job Shop Scheduling". In: *Management Science* 34.3, pp. 391–401. ISSN: 0025-1909. URL: https://www.jstor.org/stable/2632051 (visited on 09/07/2022).

Alam, Sameer et al. (2017). "A Distributed Air Traffic Flow Management Model for European Functional Airspace Blocks". en. In: URL: https://enac.hal.science/hal-01511340v1/preview/Paper_55.pdf.

Anily, S. and A. Federgruen (Sept. 1987). "Simulated annealing methods with general acceptance probabilities". In: *Journal of Applied Probability* 24.3, pp. 657–667. ISSN: 1475-6072. DOI: https://doi.org/10.2307/3214097.

ASN, Aviation Safety Network (2024). *Boeing 707-321B*. URL: https://aviation-safety.net/database/record.php?id=19780420-1.

ATO, FAA; SWIM; (May 2016). *Connect with System Wide Information Management (SWIM) 2016*. SWIM Program Office, ATO International Office. URL: https://www.icao.int/APAC/Meetings/2016%20SWIM/2.3%20-%20ICAO%20Bangkok%20SWIM%20Connect%20May%202016_FINAL(5-2).pdf.

Bae, Sangjun et al. (Sept. 2018). "A New Multiple Flights Routing and Scheduling Algorithm in Terminal Manoeuvring Area". In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE. DOI: 10.1109/dasc.2018.8569239.

Bai, Yuhan (2022). "RELU-Function and Derived Function Review". In: *SHS Web of Conferences* 144. Ed. by A. Luqman, Q. Zhang, and W. Liu, p. 02006. ISSN: 2261-2424. DOI: 10.1051/shsconf/202214402006.

Balakrishnan, Hamsa and Bala G. Chandran (Dec. 2010). "Algorithms for Scheduling Runway Operations Under Constrained Position Shifting". In: *Operations Research* 58.6, pp. 1650–1665. ISSN: 1526-5463. DOI: doi:10.1287/opre.1100.0869.

— (2014). "Optimal Large-Scale Air Traffic Flow Management". en. In: *Massachusetts Institute of Technology, Tech. Rep.* URL: https://www.mit.edu/~hamsa/pubs/BalakrishnanChandran_ATFM.pdf.

Ball, Michael O. et al. (2001). "Collaborative Decision Making in Air Traffic Management: Current and Future Research Directions". In: pp. 17–30. ISSN: 1431-9373. DOI: 10.1007/978-3-662-04632-6_2.

Bertsimas, Dimitris and Shubham Gupta (May 2011). "A proposal for network air traffic flow management incorporating fairness and airline collaboration". In.

Blom, Henk A. P. and G. J. Bakker (June 2015). "Safety Evaluation of Advanced Self-Separation Under Very High En Route Traffic Demand". In: *Journal of Aerospace Information Systems* 12.6, pp. 413–427. ISSN: 2327-3097. DOI: 10.2514/1.I010243.

Bolić, Tatjana et al. (Feb. 2017). "Reducing ATFM delays through strategic flight planning". In: *Transportation Research Part E: Logistics and Transportation Review* 98, pp. 42–59. ISSN: 1366-5545. DOI: doi:10.1016/j.tre.2016.12.001.

Bosson, Christabelle S. and Dengfeng Sun (July 2016). "Optimization of Airport Surface Operations Under Uncertainty". In: *Journal of Air Transportation* 24.3, pp. 84–92. DOI: 10.2514/1.d0013.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex optimization*. Cambridge University Press, p. 730. ISBN: 9780521833783.

Butler, James Franklin (1987). *An air traffic control simulator for the evaluation of flow management strategies*. Tech. rep. Massachusetts Institute of Technology. Flight Transportation Laboratory.

CAAS (2017). *Leveraging on ATFM and A-CDM to optimise Changi Airport operations*. URL: https://www.icao.int/Meetings/ATFM2017/Documents/3-Gan%20Heng%20A-CDM%20at%20Global%20ATFM%20symposium_gh.pdf.

Chandra, Aitichya, Nipun Choubey, and Ashish Verma (2025). "Some Comments on the Aircraft Landing Problem: How Optimal is the First Come First Serve Policy?" In: *Transportation Research Procedia* 82, pp. 923–937. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2024.12.244.

Dear, Roger George (Aug. 1978). "The dynamic scheduling of aircraft in the near terminal area." In: *Transportation Research* 12.4, pp. 297–298. ISSN: 0041-1647. DOI: https://doi.org/10.1016/0026-2714(89)90171-6.

Deb, K. et al. (Apr. 2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2, pp. 182–197. DOI: 10.1109/4235.996017.

DeepSeek-AI et al. (2025). "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning". In: DOI: https://doi.org/10.48550/arXiv.2501.12948.

Delahaye, Daniel, Supatcha Chaimatanan, and Marcel Mongeau (Sept. 2018). "Simulated Annealing: From Basics to Applications". In: pp. 1–35. ISSN: 2214-7934. DOI: https://doi.org/10.1007/978-3-319-91086-4_1.

Desai, Jitamitra et al. (Aug. 2022). "A 0–1 mixed-integer program-based group-and-release strategy for solving the integrated runway scheduling and taxiway routing problem". In: *Naval Research Logistics (NRL)* 69.7, pp. 939–957. DOI: 10.1002/nav.22072.

Egami, Shusaku et al. (Oct. 2019). "Enriching Geospatial Representation for Ontology-based Aviation Information Exchange". In: DOI: 10.1109/GCCE46687.2019.9015574.

Erkan, Hale, Nesim K. Erkip, and Özge Şafak (Nov. 2019). "Collaborative decision making for air traffic management: A generic mathematical program for the rescheduling problem". In: *Computers and Industrial Engineering* 137, p. 106016. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106016.

EUROCONTROL (2018). *European aviation in 2040, challenges of growth.* Tech. rep. URL: https://www.eurocontrol.int/sites/default/files/content/documents/official-documents/reports/challenges-of-growth-2018.pdf.

— (2019). *User Manual for the Base of Aircraft Data (BADA) Revision 3.15, EEC Technical/Scientific Report No. 19/03/18-45*. Tech. rep. EUROCONTROL Experimental Centre (EEC).

— (Sept. 2024). *Aircraft Performance Database*. URL: https://contentzone.eurocontrol.int/aircraftperformance/default.aspx?.

FAA (2019). *Cost of Delay Estimates*. URL: https://www.faa.gov/sites/faa.gov/files/data_research/aviation_data_statistics/cost_delay_estimates.pdf.

Festa, P. (July 2014). "A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems". In: DOI: 10.1109/icton.2014.6876285.

Finnsson, Hilmar and Yngvi Björnsson (Jan. 2008). "Simulation-Based Approach to General Game Playing". In: vol. 1, pp. 259–264.

Forbus, John J. and Daniel Berleant (Oct. 2022). "Discrete-Event Simulation in Healthcare Settings: A Review". In: *Modelling* 3.4, pp. 417–433. ISSN: 2673-3951. DOI: https://doi.org/10.3390/modelling3040027.

Frankovich, Michael Joseph (2012). "Air traffic flow management at airports : a unified optimization approach". eng. Accepted: 2013-03-13T15:51:50Z. Thesis. Massachusetts Institute of Technology. URL: https://dspace.mit.edu/handle/1721.1/77826 (visited on 02/21/2023).

Gardner, Everette S. (Jan. 1985). "Exponential smoothing: The state of the art". In: *Journal of Forecasting* 4.1, pp. 1–28. ISSN: 1099-131X. DOI: https://doi.org/10.1002/for.3980040103.

Gendreau, Michel and Jean-Yves Potvin (2010). "Handbook of Metaheuristics". In: *International Series in Operations Research; Management Science*. ISSN: 0884-8289. DOI: https://doi.org/10.1007/978-1-4419-1665-5.

Gupta, Uma G. (Apr. 1997). "Using Citation Analysis to Explore the Intellectual Base, Knowledge Dissemination, and Research Impact of Interfaces (1970–1992)". In: *Interfaces* 27.2, pp. 85–101. ISSN: 1526-551X. DOI: https://doi.org/10.1287/inte.27.2.85.

Gurobi (2025). *Gurobi optimizer delivers unmatched performance*.

Guruge, Gammana (2020). "Real-time flight scheduling for optimal air traffic flow management". In: DOI: 10.32657/10356/140363.

Henderson, Darrall, Sheldon H. Jacobson, and Alan W. Johnson (2006). "The Theory and Practice of Simulated Annealing". In: pp. 287–319. DOI: 10.1007/0-306-48056-5_10.

Henry, Antoine, Daniel Delahaye, and Alfonso Valenzuela (June 2022). "CONFLICT RESOLUTION WITH TIME CONSTRAINTS IN THE TERMINAL MANEUVERING AREA USING A DISTRIBUTED Q-LEARNING ALGORITHM". In: *International Conference on Research in Air Transportation (ICRAT 2022)*. Tampa, United States. URL: https://hal-enac.archives-ouvertes.fr/hal-03701660.

Hu, Xiao-Bing and Wen-Hua Chen (2005). "Receding horizon control for aircraft arrival sequencing and scheduling". In: *IEEE Transactions on Intelligent Transportation Systems* 6.2, pp. 189–197. DOI: 10.1109/TITS.2005.848365.

Huo, Ying, Daniel Delahaye, and Mohammed Sbihi (Nov. 2021). "A probabilistic model based optimization for aircraft scheduling in terminal area under uncertainty". In: *Transportation Research Part C: Emerging Technologies* 132, p. 103374. DOI: 10.1016/j.trc.2021.103374.

— (Apr. 2023). "A dynamic control method for extended arrival management using enroute speed adjustment and route change strategy". In: *Transportation Research Part C: Emerging Technologies* 149, p. 104064. DOI: 10.1016/j.trc.2023.104064.

IATA (2020). *Value of Aviation - Country Reports*. Tech. rep. URL: https://www.iata.org/en/publications/economics/economics-library/?Search=&EconomicsL2=175&Ordering=DateDesc.

— (2023a). *Air Passenger Market Analysis*. Tech. rep. URL: https://www.iata.org/en/iata-repository/publications/economic-reports/air-passenger-market-analysis-december-2023/.

— (Dec. 2023b). *Airline revenue to surpass pre-pandemic levels in 2023*. Tech. rep. URL: https://www.iata.org/en/iata-repository/publications/economic-reports/airline-revenue-to-surpass-pre-pandemic-levels-in-2023/.

ICAO (2005). *Global Air Traffic Management Operational Concept*. Tech. rep.

— (2012). *Doc. 9965. Manual on Flight andFlow Information for a Collaborative Environment (FF-ICE)*. Tech. rep.

— (2015). *ICAO Doc 10039: Manual on System Wide Information Management Concept*. Tech. rep. URL: https://www.icao.int/safety/acp/ACPWGF/CP%20WG-I%2019/10039_SWIM%20Manual.pdf.

— (2016). *Doc 4444. PROCEDURES FOR AIR NAVIGATION SERVICES. Air Traffic Management*. Tech. rep. International Civil Aviation Organization. URL: http://sar.mot.go.th/document/ICAO/ICAO%20Doc%204444-Pans-Air%20Traffic%20Management%2016th%20edition%202016.pdf.

— (2018). *Doc. 9971. Manual on Collaborative Air Traffic Flow Management (ATFM)*. Tech. rep.

— (Jan. 2019). *Future of Aviation*. URL: https://www.icao.int/Meetings/FutureOfAviation/Pages/default.aspx.

— (2023a). *Safety Report 2023*. Tech. rep. URL: https://www.icao.int/safety/Documents/ICAO_SR_2023_20230823.pdf.

— (Feb. 2023b). *World Airlines Traffic and Capacity*. URL: https://www.airlines.org/dataset/world-airlines-traffic-and-capacity/.

Kalashnikov, Vladimir V. (1994). "Mathematical Methods in Queuing Theory". In: DOI: https://doi.org/10.1007/978-94-017-2197-4.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: DOI: https://doi.org/10.48550/arXiv.1412.6980.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (May 1983). "Optimization by Simulated Annealing". In: *Science* 220.4598, pp. 671–680. ISSN: 1095-9203. DOI: 10.1126/science.220.4598.671.

Klemperer, Paul (2008). "Network Goods (Theory)". In: pp. 1–4. DOI: https://doi.org/10.1057/978-1-349-95121-5_2178-1.

Kok, Yufeng (June 2023). *S'pore, Japan, Thailand and US testing new way to manage flights more efficiently across regions*. URL: https://www.straitstimes.com/singapore/transport/s-pore-japan-thailand-and-us-testing-new-way-to-manage-flights-more-efficiently-across-regions.

Lavandier, Julien et al. (Oct. 2021). "Selective Simulated Annealing for Large Scale Airspace Congestion Mitigation". In: *Aerospace* 8.10, p. 288. DOI: 10.3390/aerospace8100288.

Law, Averill M. (2015). *Simulation Modeling and Analysis*. Fifth Edition. Description based on publisher supplied metadata and other sources. New York: McGraw-Hill US Higher Ed USE Legacy. 1776 pp. ISBN: 0073401323.

Liang, Diana, Kristin Cropf, et al. (Apr. 2019). "Operational Evaluation of FF-ICE/R2". In: DOI: 10.1109/ICNSURV.2019.8735320.

Liang, Diana, Thien Ngo, et al. (Apr. 2021). "Operational and Technical Evaluation of The FF-ICE/Execution Strategic Phase Across Boundaries". In: DOI: 10.1109/ICNS52807.2021.9441642.

Liang, Diana, Nabil Sandhu, et al. (Apr. 2021). "Evaluation of the Four-Dimensional Trajectory Live Flight Demonstration (4DT LFD) Project". In: DOI: 10.1109/ICNS52807.2021.9441598.

Liang, Tengyuan and James Stokes (2018). "Interaction Matters: A Note on Non-asymptotic Local Convergence of Generative Adversarial Networks". In: DOI: https://doi.org/10.48550/arXiv.1802.06132.

Ma, Ji (2019). "Optimisation du trafic aérien dans de grands aéroports". PhD thesis. Université Toulouse 3 - Paul Sabatier. URL: https://theses.hal.science/tel-03010033/document.

Ma, Ji, Daniel Delahaye, and Man Liang (July 2024). "Arrival and Departure Sequencing, Considering Runway Assignment Preferences and Crossings". In: *Aerospace* 11.8, p. 604. ISSN: 2226-4310. DOI: https://doi.org/10.3390/aerospace11080604.

Ma, Ji, Mohammed Sbihi, and Daniel Delahaye (Mar. 2019). "Optimization of departure runway scheduling incorporating arrival crossings". In: *International Transactions in Operational Research* 28.2, pp. 615–637. DOI: 10.1111/itor.12657.

Metropolis, Nicholas et al. (June 1953). "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6, pp. 1087–1092. ISSN: 1089-7690. DOI: 10.1063/1.1699114.

Mondoloni, Stephane (2013). "Improved Trajectory Information for the Future Flight Planning Environment". In: MITRE. URL: https://www.researchgate.net/profile/Stephane-Mondoloni/publication/344380598_Improved_Trajectory_Information_for_the_Future_Flight_Planning_Environment/links/5f6e4e44458515b7cf507124/Improved-Trajectory-Information-for-the-Future-Flight-Planning-Environment.pdf.

N X.D Lu N Wickramasinghe, M Brown (2022). *SWIM Based Trajectory Coordination to Achieve Strategic Planning and Collaborative Decision Making*. Tech. rep. Electronic Navigation Research Institute. URL: https://www.jstage.jst.go.jp/article/iwac/1/0/1_220/_pdf/-char/ja.

NATCA (Dec. 2019). *A History of Air Traffic Control*. URL: https://www.natca.org/wp-content/uploads/2019/12/NATCA_ATC_History.pdf.

Ng, Wayne and Nuno Antunes Ribeiro (2023). "Strategic Model To Optimize Terminal Airspace Operations". In: *Air Transport Research Society (ATRS)*.

Ngo, Thien et al. (Apr. 2019). "Technical Evaluation of FF-ICE/R2". In: *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE. DOI: 10.1109/ICNSURV.2019.8735347.

OAG (Feb. 2024a). *COVID-19 AIR TRAVEL RECOVERY*. URL: https://www.oag.com/coronavirus-airline-schedules-data.

— (Mar. 2024b). *HISTORICAL DATA*. URL: https://www.oag.com/historical-flight-data.

Odoni, Amedeo R. (1987). "The Flow Management Problem in Air Traffic Control". In: pp. 269–288. DOI: 10.1007/978-3-642-86726-2_17.

Ozgur, Metin and Aydan Cavcar (Feb. 2014). "0–1 integer programming model for procedural separation of aircraft by ground holding in ATFM". In: *Aerospace Science and Technology* 33.1, pp. 1–8. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2013.12.009.

Prakash, Rakesh, Jitamitra Desai, and Rajesh Piplani (Nov. 2021). "An optimal data-splitting algorithm for aircraft sequencing on a single runway". In: *Annals of Operations Research* 309.2, pp. 587–610. DOI: 10.1007/s10479-021-04351-2.

Prakash, Rakesh, Rajesh Piplani, and Jitamitra Desai (Oct. 2018). "An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput". In: *Transportation Research Part C: Emerging Technologies* 95, pp. 570–581. DOI: 10.1016/j.trc.2018.07.031.

Rapaya, Abba, P. Notry, and Daniel Delahaye (2021). "Coordinated Sequencing of Traffic on Multiple En-Route Constraint Points". In: pp. 41–57. ISSN: 1876-1119. DOI: https://doi.org/10.1007/978-981-33-4669-7_3.

Ribeiro, Nuno Antunes et al. (June 2018). "An optimization approach for airport slot allocation under IATA guidelines". In: *Transportation Research Part B: Methodological* 112, pp. 132–156. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2018.04.005.

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In: DOI: https://doi.org/10.48550/arXiv.1609.04747.

Rutenbar, R.A. (Jan. 1989). "Simulated annealing algorithms: an overview". In: *IEEE Circuits and Devices Magazine* 5.1, pp. 19–26. ISSN: 8755-3996. DOI: 10.1109/101.17235.

SCMP (Oct. 2018). *Why the world's flight paths are such a mess*. URL: https://multimedia.scmp.com/news/world/article/2165980/flight-paths/index.html.

Sekar, John A. P. and James R. Faeder (2012). "Rule-Based Modeling of Signal Transduction: A Primer". In: pp. 139–218. ISSN: 1940-6029. DOI: https://doi.org/10.1007/978-1-61779-833-7_9.

SESAR (2024). *TBO roadmap*. URL: https://www.sesarju.eu/node/4843.

Stoen, Hal (n.d.). *HOLD IT!* Accessed: 2025-08-17. No publication year specified. URL: https://www.aavirtual.com/pages/Tutorial/Hold_it.htm.

Tan, Benjamin W. J. (2021). "Interconnectivity of Airline Schedules and Delay Propagation in Air Traffic Flow Management". PhD thesis. Singapore University of Technology and Design.

travelperk (Feb. 2025). *Delayed and canceled flight statistics from 2024*. URL: https://www.travelperk.com/blog/delayed-canceled-travel-statistics/.

Wambsganss, Michael C. (2001). "Collaborative Decision Making in Air Traffic Management". In: pp. 1–15. ISSN: 1431-9373. DOI: https://doi.org/10.1007/978-3-662-04632-6_1.

Xiangwei, Meng, Zhang Ping, and Li Chunjin (2011). "Sliding window algorithm for aircraft landing problem". In: *2011 Chinese Control and Decision Conference (CCDC)*, pp. 874–879. DOI: 10.1109/CCDC.2011.5968306.

Ye, Bojia, Minghua Hu, and John Friedrich Shortle (Feb. 2014). "Collision risk-capacity tradeoff analysis of an en-route corridor model". In: *Chinese Journal of Aeronautics* 27.1, pp. 124–135. ISSN: 1000-9361. DOI: https://doi.org/10.1016/j.cja.2013.12.007.

Zhou, Jun et al. (2017). "Air Traffic Management and Systems II". In: Springer Japan. Chap. Optimizing the Design of a Route in Terminal Maneuvering Area Using Branch and Bound. ISBN: 9784431564232. DOI: 10.1007/978-4-431-56423-2.