A Systematic Study of Terminal Area Traffic Management

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Heming Chen

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Yiyuan Joseph Zhao, Advisor

August, 2012

Acknowledgements

Completing my Ph.D. degree is the most wonderful experience in the first twenty-eight years of my life. It was my mentors, family and friends that helped me get through the best and worst moments of this journey. I sincerely wish to share this happy moment with all of them.

Firstly, I'm truly grateful to my faculty advisor, Professor Yiyuan Zhao. Without his vision, guidance and encouragement, I wouldn't be able to complete the Ph.D. program. Professor Zhao walked me into the fascinating area of aerospace engineering, and helped me lay a solid foundation in my graduate study, especially in optimization and air traffic control. The learning experience with him transformed me from a student into a real scholar. Secondly, special thanks to my committee, Professor William Garrard, Professor Demoz Gebre-Egziabher and Professor Rajesh Rajamani for their support, guidance and helpful suggestions. I want to thank my friends and office mates for their many helpful discussions, suggestions and encouragement. They brought me the happiest time in my life. I also wish to thank the Department of Aerospace Engineering and Mechanics for the financial support during the past five years. Thank all faculty members in AEM who had inspired me and given me a wonderful learning experience.

I want to thank my parents for their love and support everyday in my life. It was their strong support that enabled me to receive excellent education until receiving my Ph.D. degree. Even though being thousands of miles away, it was their encouragement that always kept me moving forward. In the last, I would like to express my deepest

love to my wife Yao. I thank her for the love, understanding and support. Without her, I couldn't have made the decision to pursue a Ph.D. degree. It was the strong support from all my family members that made me accomplish this work.

This work was supported in part by NASA.

Dedication

This work is dedicated to my parents and Yao.

Abstract

This dissertation conducts a systematic study of terminal area traffic management using optimization methods. The critical role that the airport plays in national airspace system is introduced. The challenges in managing complex terminal traffic are explained. The necessity and importance of the study is addressed. To assist human air traffic controllers in managing complex terminal traffic, our solution strategy is to calculate each flight's optimal arrival/departure schedule, to minimize the overall flight delay and runway congestion in the entire airport. Three solutions are developed in the thesis, including the static solution, the dynamic solution, and the stochastic solution. The static solution uses one computation and attempts to optimize the schedules of many flights arriving/departing the airport within a wide time window. The accuracy of its solution heavily relies on the quality of the predicted traffic situation acquired right before the computation. On the contrary, the dynamic solution attempts to divide the entire traffic flow into a series of small pieces, and optimize flight schedules piece by piece. It collects the latest traffic information before each computation and experiences far less computational load. Both static and dynamic solutions assume the traffic information to be explicitly known. They are inherently deterministic solutions. The third solution proposed in this thesis is a stochastic solution. It assumes that traffic information is not known with certain due to a variety of random factors in actual flight operations. This stochastic solution is mathematically and structurally designed to handle multiple sources of uncertainties in managing terminal traffic. In the last, the conclusion is given based upon the simulation tests of the three proposed runway scheduling solutions. Future work is also suggested.

Contents

List of Tables

List of Figures

Nomenclature

N_{arr}	Total number of arrivals
N_{dep}	Total number of departures
R	Total number of arrival routes
S	Total number of departure routes
P	Total number of arrival scheduling points on an arrival route
Q	Total number of departure scheduling points on a departure route
K	Total number of common sections between two routes
t	Scheduled Time of Arrival (STA)
$\hat{t}$	Estimated Time of Arrival (ETA)
$\mathcal{M}$	A large positive real number
C	Cost of Delay
T_C	Computational time for one scheduling window
T_L	Look-ahead time
T_W	Length of scheduling windows
T_{adv}	Advance time
T_{wait}	Waiting time
i, i_1, i_2	Arrival indices, where $1 \leq i, i_1, i_2 \leq N_{\text{arr}}$
j, j_1, j_2	Departure indices, where $1 \leq j, j_1, j_2 \leq N_{\text{dep}}$
r, r_1, r_2	Arrival route indices, where $1 \leq r, r_1, r_2 \leq R$
s, s_1, s_2	Departure route indices, where $1 \leq s, s_1, s_2 \leq S$
p, p_1, p_2	Scheduling point indices on arrival routes, where $1 \leq p, p_1, p_2 \leq P$
q, q_1, q_2	Scheduling point indices on departure routes, where $1 \leq q, q_1, q_2 \leq Q$
k	Common section index, where $1 \leq k \leq K$

Chapter 1

Introduction

Being one of the most crucial components in the national airspace system, the airport has been directly experiencing the impact of growth in both passenger and cargo throughput in the past decade. However, the current-day air traffic control (ATC) procedures are ad-hoc in nature and less than fully automatic. Like two decades ago, human air traffic controllers still need to read radar screens to monitor the status of each aircraft. ATC instructions are still being issued via voice radio communication between air traffic controllers and pilots. Even in today, air traffic controllers are still relying on their experiences and some predefined rules to coordinate the arrival and departure traffic in airports. Due to the rapid increase in air traffic during the past decade, air traffic controllers in major U.S. airports have long been under heavy workload and high stress. When under pressure, human air traffic controllers can hardly issue ATC instructions that are fully optimized according to the real-time traffic situation. Passengers therefore have also been experiencing more delays and traffic jams in today.

As a result, it is quite desirable to develop automated computer systems that can assist air traffic controllers in managing airport traffic. Typical airport ATC instructions include runway assignment, landing/takeoff flight sequencing, arrival/departure time adjustment, and even runway configuration change according to the varying traffic and

weather conditions. The ultimate goal is to take full advantage of airport resources, such as airspace, runways, taxi routes, ramps, and gates, to reduce flight delays. Meanwhile, flight safety must be always maintained by separating flights by safe distances. The Next Generation Air Transportation System (NextGen)[1] also pursuits this goal.

Historically, researches have formulated this terminal area traffic management problem as the "runway scheduling" problem. This problem studies the optimal aircraft sequencing and routing rules to release runway congestion and reduce flight delays occurred in the terminal area. In fact, a central goal of all these studies is to maximize airport throughput subject to the operational constraints and airport resource limitations.

Optimization theory becomes a perfect fit in pursuing such a goal and played an important role in the past. Many existing studies convert ATC rules into mathematical languages and adopt optimization algorithms to compute optimal runway assignments, landing/takeoff sequences, and arrival/departure schedules. Once optimization algorithms are coded as computer programs, computers can generate a large volume of ATC instructions in a short amount of time that human controllers can hardly compete with. More importantly, ATC instructions generated by optimization algorithms are guaranteed to be optimal and can greatly reduce runway congestion and flight delays. Human controllers can hardly reach the same level of optimality in such a short amount of computational time. Therefore, this dissertation continues the study in runway scheduling problem using optimization theories to handle anticipated heavy traffic demand in the future. This work also attempts to lay a solid foundation for building state-of-the-art automated ATC systems for the future.

For the sake of convenience, we divide the traffic into different groups by either their directions or physical locations. For instance, by the direction, we define arriving traffic and departing traffic. By the physical location, we define airborne traffic and ground traffic. In reality, although flowing different ATC procedures, these traffic groups need

to share airport resources such as airspace, runways, taxi routes, ramps, and gates. It is critical to realize the existence of all traffic groups while distributing airport resources. However, most existing studies only consider one particular traffic group, such as the arriving traffic[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Ground traffic have also been investigated using optimization algorithms[12, 13, 14, 15, 16]. For instance, Smeltink et al.[12] and Rathinam et al.[15] adopt mixed integer linear program (MILP) to solve the scheduling problem. Keith et al.[16] extends Rathinam et al.'s work by combining routing and scheduling in one model. Other studies design practical real-time scheduling strategies for a particular group of traffic. They incorporate controller preferences and use a combination of optimization techniques, intelligence methods, and operation rules[17, 18, 19, 20, 21, 22, 23, 24]. Airborne traffic scheduling in strategic level has also been investigated. Cheng[25], Bolat[26] and Kim et al.[27] study arriving traffic scheduling using gate assignment rules. Metroplex airborne traffic scheduling has been covered by Capozzi et al.[28] to understand how to optimize traffic flow around several closely spaced airports. Historically, only a few researchers have studied both arriving and departing traffic[29, 30].

Although the runway scheduling problem has been extensively examined, most of the existing work lack an integrated view of airport traffic situation. Most of them study the strategies for managing a particular traffic group while ignoring the existence of others. In this thesis, we want to study the integrated runway scheduling strategies for two reasons. The first reason is for efficiency. Solution approaches that individually optimize each traffic group's ATC operations are essentially providing local optimal results rather than global optimal results. To achieve the latter, all traffic groups must be jointly optimized. The second reason for integrated scheduling is for safety. Airport resources, such as runways, taxi routes, airspace, are always shared among all traffic groups. Neglecting some traffic can result in aircraft conflicts and even collisions. On the contrary, studying an integrated runway scheduling problem can help resolving lots of potential conflicts and prevent aircraft collisions. In all, neglecting a particular group

of traffic in the study not only downgrades the efficiency, but also jeopardizes the safety.

Moreover, most existing studies can only handle traffic arriving/departing the airport within a static time interval, which is normally short due to computational constraints. We call them the static methods. Nevertheless, real-world traffic arrives continually at the airport throughout the entire day. Static methods cannot directly handle continual traffic streams and do not give explicit instructions on how to do it. Intuitively, to accept more traffic, we either need to handle all-day-long traffic in one computation, or divide continual traffic flow into a series of pieces and handle them one after another. However, the former approach is computationally expensive and the latter is not even addressed in most of previous works. In this work, we will look into this issue and provide solutions that can perform "dynamic scheduling."

Uncertainties in flight operations could also affect the efficiency in managing terminal area traffic. Since runway scheduling is always conducted in advance, its solution heavily relies on the accuracy of predicted traffic situation. The predicted traffic situation is often obtained from aircraft trajectory predictions and does not guarantee one hundred percent accuracy. Even worse, as the prediction time increases, the uncertainties in the predicted aircraft trajectories can grow drastically. For these reasons, static methods cannot always provide accurate and reliable solutions. In the past, Solveling et al.[31] has introduced a stochastic runway sequencing solution. However, just like most of the other studies, they only consider one runway rather than the entire airport. The possibility of using runway traffic balancing to reduce flight delay under uncertain traffic is not discussed either. Therefore, we will also look into this issue and develop the solution called integrated stochastic runway scheduling.

Thus, this dissertation proposes three solutions to the three above-mentioned issues in the remainder of the dissertation. An integrated multiple-point static scheduling framework is first developed. Unlike previous investigations, this framework coordinates all groups of traffic in the terminal area, and distributes all airport resources to all

traffic in an efficient way. Furthermore, it monitors and controls flight schedules at multiple locations, such as terminal airspace entrances/exists, runways, taxiways, gates, etc. It incorporates airport operating regulations and resource limitations as numerical constraints for real-world applicability. This static scheduling framework can quickly produce hundreds of flight schedules in a few minutes, which is more efficient than human controllers.

To handle continual traffic flows, several dynamic strategies are developed following the divide-and-conquer philosophy. A long planning horizon is divided into a series of smaller time intervals, called the scheduling windows. A static scheduling method is executed in each of these time intervals sequentially to process continual traffic flows. A dynamic strategy can use most of the existing static scheduling methods in processing continual traffic flows across a long time interval. It is especially useful in real-time ATC operations. Also, repeated computations can use the newest traffic data and thus produce solutions less vulnerable to the uncertainties in actual flight.

To resolve the uncertainty issue, a structural solution and an algorithmic solution are provided. Similar as the dynamic strategies, a structural solution processes the incoming traffic segment by segment, and adopts the newest traffic data in each computation to reduce the impact from uncertainties in predicted traffic situation. This solution is also compatible with most existing deterministic static runway scheduling methods. The algorithmic solution mathematically models the uncertainties and attempts to produce optimal flight schedules weighted against all possible traffic situations. This solution adopts stochastic optimization algorithms making it fundamentally different from the other solutions.

Chapter 2

Modeling of the Airport

To study traffic flows in the airport, we first need to setup the mathematical representation of an airport. This study employs a multiple-point scheme for the integrated routing and scheduling of both arriving and departing traffic, as shown in Fig. 2.1. To solve this problem, the airborne flight routes and taxi routes are modeled as a network that contains all the available taxi routes, runways, ascent routes, descent routes, departure fixes, and arrival fixes. In this network model, it is assumed that each departing flight travels on a route that connects a terminal with a departure fix, and each arriving flight is on a route from an arrival fix to a terminal. Each route consists of an airborne section and a surface section. Its airborne section is typically regarded as the ascent route or decent route, whereas the surface section is equivalent to the taxi-out route or taxi-in route. One that contains one or more different segments from another route is considered a different route. The airborne routes and surface routes of the complete route network used in this work are presented in Figs. 2.2 and 2.3 respectively. This route network model represents an extension of the route structure used in optimal surface aircraft planning and scheduling[16]. It is a directed graph, where the routes and segments may also be called paths and edges[32], respectively.

Scheduling points are locations in the network at which aircraft arrival times are

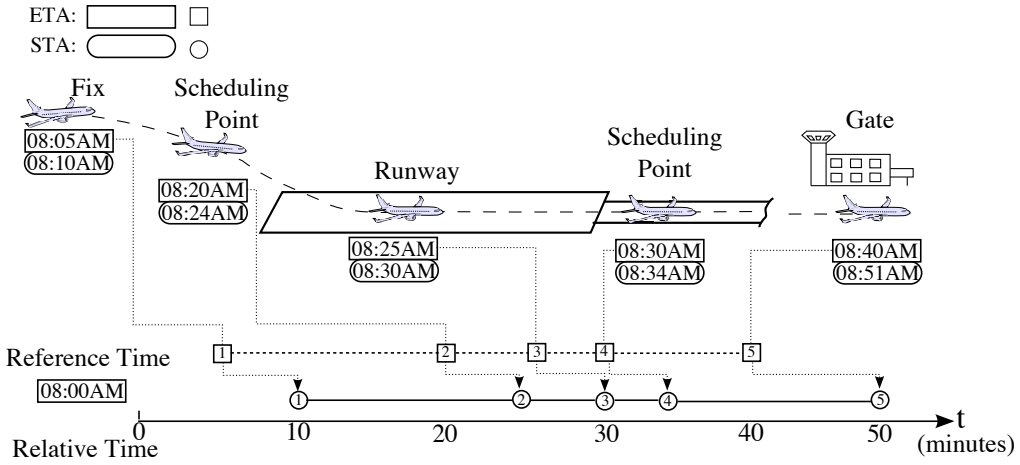


Figure 2.1: An example of multiple-point scheduling.

monitored and scheduled. In the multiple-point scheduling scheme, important locations, such as fixes, runways, and gates, are defined as scheduling points. Additional scheduling points are also added along routes for improved flexibility and trajectory accuracy. By adjusting the arrival times of individual aircraft at the scheduling points, inter-aircraft separations at these scheduling points as well as along routes can be maintained. Once arrival times at scheduling points are determined, aircraft speeds along their assigned routes can be adjusted accordingly.

A single airport with multiple runways is assumed in all numerical examples of this study. As shown in Fig. 2.2, the example has a total of seven routes and two runways. The three arrival routes start from arrival fix ROBER, LENDY and CAMRN respectively. The arrival route from ROBER lands on Runway 31R and the other two land on 31L. In Fig. 2.2, all three arrival routes converge on the ground and eventually reach the same terminal ramp, as shown in Fig. 2.3. There are also four departure routes. They all start from the same terminal ramp and pass through Runway 31L. They then diverge and end at the four departure fixes: OCEAN GATE, EAST GATE, NORTH GATE and WEST GATE.

A given route can pass through only one runway, and runway assignments can be achieved through route selections. In addition, a route network can represent a runway configuration through the set of selected routes. Therefore, the multiple-point route network model can be used to study different runway configurations.

As mentioned before, the multiple-point scheduling scheme has another advantage. There are many sources of uncertainty in the National Airspace System. These uncertainties can cause scheduling inefficiencies or flight deviations. A solution strategy may provide robust solutions through algorithmic design and/or the solution structure. By calculating schedules at multiple points, the proposed scheme offers the structural mechanisms to compensate for deviations.

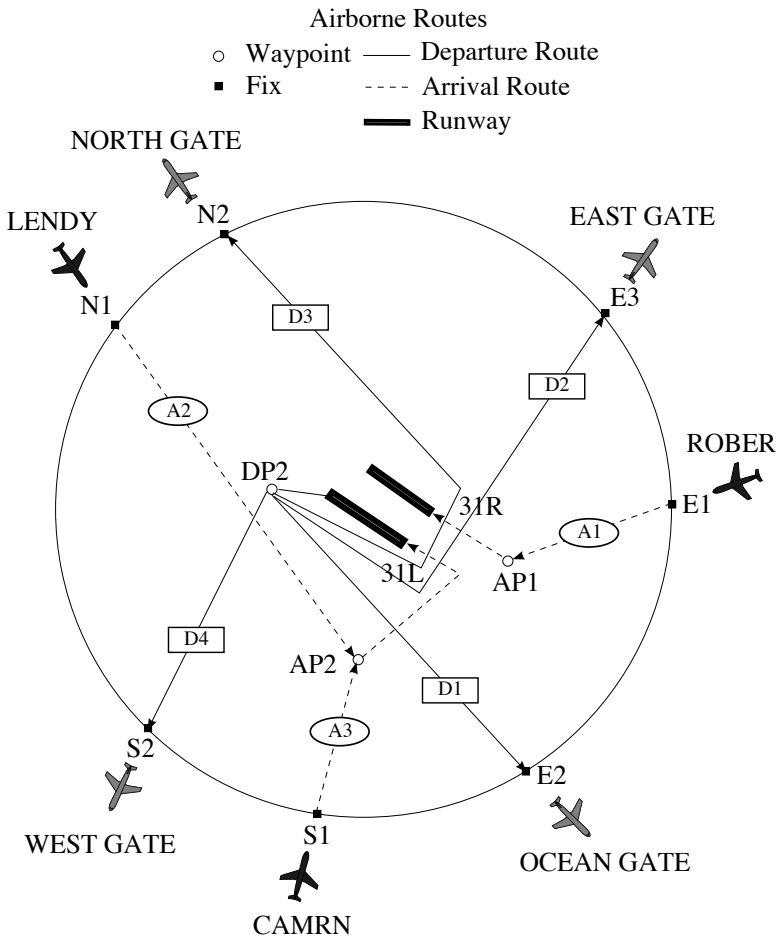


Figure 2.2: Airborne routes (not to scale).

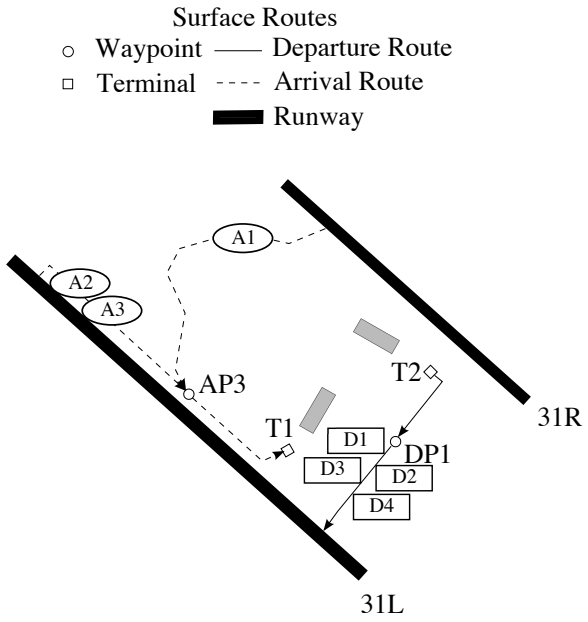


Figure 2.3: Surface routes (not to scale).

Chapter 3

Static Runway Scheduling

3.1 Problem Formulation

The following mixed integer linear programming (MILP) formulation is adopted to study static runway scheduling. Table 3.1 gives its dimensions.

Table 3.1: Problem dimensions

Symbol	Meaning
N_{arr}	Total number of arrivals
N_{dep}	Total number of departures
R	Total number of arrival routes
S	Total number of departure routes
P	Total number of arrival scheduling points
Q	Total number of departure scheduling points
K	Total number of common scheduling points between two routes
i, i_1, i_2	Arriving flight indices, where $1 \leq i, i_1, i_2 \leq N_{\text{arr}}$
j, j_1, j_2	Departing flight indices, where $1 \leq j, j_1, j_2 \leq N_{\text{dep}}$
r, r_1, r_2	Arrival route indices, where $1 \leq r, r_1, r_2 \leq R$
s, s_1, s_2	Departure route indices, where $1 \leq s, s_1, s_2 \leq S$
p, p_1, p_2	Scheduling point indices on arrival routes, where $1 \leq p, p_1, p_2 \leq P$
q, q_1, q_2	Scheduling point indices on departure routes, where $1 \leq q, q_1, q_2 \leq Q$
k	Common section index, where $1 \leq k \leq K$

3.1.1 Decision Variables

Three groups of decision variables are introduced. The first group decides on route assignments.

$$x_i^r = \begin{cases} 1 & \text{if arrival route } r \text{ is assigned to arrival } i, \\ 0 & \text{otherwise.} \end{cases}$$

$$x_j^s = \begin{cases} 1 & \text{if departure route } s \text{ is assigned to departure } j, \\ 0 & \text{otherwise.} \end{cases}$$

where $1 \leq i \leq N_{\text{arr}}$, $1 \leq j \leq N_{\text{dep}}$, $1 \leq r \leq R$ and $1 \leq s \leq S$. If the computed solution shows $x_i^r = 1$, then arrival i must be directed to route r. Otherwise, arrival i cannot choose route r. Proper constraints must be included if the route assignment for an aircraft is known before scheduling. For example, if departure j cannot use route s, then $x_j^s = 0$ is added as a constraint.

The second group of decision variables represents the continuous STAs at various scheduling points. The collection of these scheduling points uniquely specify a complete route. Specifically, t_{ip} represents the STA of arrival i at the pth scheduling point on its assigned route, and t_{jq} represents the STA of departure j at the qth scheduling point on its assigned route, where $1 \leq p \leq P$ and $1 \leq q \leq Q$. In comparison, $\hat{t}_{ip}$ and $\hat{t}_{jq}$ represent the corresponding ETAs. Unlike t_{ip} and t_{jq}, $\hat{t}_{ip}$ and $\hat{t}_{jq}$ are constant. They are the initial inputs to the model.

Finally, the sequence decision variables are contained in the third group. A sequence decision variable determines the order with which two flights pass through the same scheduling point on the common section between their assigned routes. This common

scheduling point can be either a runway, gate, fix, or an intermediate scheduling point.

$$z^k_{i_1 i_2} = \begin{cases} 1 & \text{if arrival } i_1 \text{ is ahead of arrival } i_2 \text{ on their } k\text{th common section,} \\ 0 & \text{otherwise.} \end{cases}$$

$$z^k_{j_1 j_2} = \begin{cases} 1 & \text{if departure } j_1 \text{ is ahead of departure } j_2, \text{ on their } k\text{th common section} \\ 0 & \text{otherwise.} \end{cases}$$

$$z^k_{ij} = \begin{cases} 1 & \text{if arrival } i \text{ is ahead of departure } j, \text{ on their } k\text{th common section} \\ 0 & \text{otherwise.} \end{cases}$$

where $1 \leq i_1, i_2, i \leq N_{\text{arr}}$, $1 \leq j_1, j_2, j \leq N_{\text{dep}}$, and $1 \leq k \leq K$.

Two routes may have multiple common sections, on which two flights may maintain different sequences. To capture this phenomenon in mathematical expressions, it is assumed that there are K sequence decisions for two flights that have a total of K common sections. For instance, $z^k_{i_1 i_2}$, the sequence of arrivals i_1 and i_2 on their kth common section, can be different from $z^{k+1}_{i_1 i_2}$. In particular, K is equal to 1 if two flights have only one common section. Although it can hold different values for different aircraft pairs, K is generally small to make sure that the route network is not overly complicated for computation.

While the above notations may overlap as mathematical expressions, they make it convenient to incorporate different requirements on arrival and departure traffic in the scheduling process, such as the different separation requirements for departure-departure, arrival-arrival, and arrival-departure pairs.

3.1.2 Optimization Criteria

General optimization criteria can include the maximization of capacity, throughput, aircraft efficiencies, and/or environmental friendliness, as well as their proper combinations. These criteria may be represented by their total values, average values, peak

values, or weighted sums. The solution processes are similar when different criteria are used. In this study, the overall flight delay is minimized.

$$\underset{T,X,Z}{\text{Min}} \quad I = \sum_{i=1}^{N_{\text{arr}}} C_i(t_i^{\text{gate}} - \hat{t}_i^{\text{gate}}) + \sum_{j=1}^{N_{\text{dep}}} C_j(t_j^{\text{fix}} - \hat{t}_j^{\text{fix}}) \tag{3.1}$$

where T, X and Z represent the collections of continuous time decision variables, route decision variables, and sequence decision variables, respectively. In this expression, t_i^{gate} and t_j^{fix} are the gate STA and fix STA for arrival i and departure j respectively, C_i and C_j are the weighting factors for the delays, and $\hat{t}$ represents a constant ETA. The difference between a STA and the corresponding ETA is defined as the delay. It is assumed that no time advance is used.

3.2 Expressions of Constraints

Proper constraints are essential in the optimal runway scheduling formulation to ensure feasibility of the solutions. These constraints may come from sources such as aircraft performance limitations, runway configurations, ambient conditions (weather), and operational procedures. For example, an aircraft is not able to change its speed, heading or altitude instantaneously, and its speed must stay within a meaningful range. In order to maintain sufficient separations, any two aircraft cannot pass through the same location at the same time. Therefore, their STAs at each common scheduling point must be sufficiently separated.

3.2.1 Single Route Constraints

In reality, each aircraft lands on or takes off from one runway. Since the terminal airspace is represented by a number of routes connecting fixes and terminals with runways, it is assumed that each flight travels on one route only. Mathematically, the route assignment variable of a flight is one with respect to its assigned route, and is zero with respect to the unselected routes. The single route constraints state that the sum of all route

assignment variables for each flight is equal to one.

$$\sum_{r=1}^{R} x_i^r = 1 \tag{3.2}$$

$$\sum_{s=1}^{S} x_j^s = 1 \tag{3.3}$$

3.2.2 Sequencing Constraints

Sequencing constraints specify the orders with which two flights share common route sections.

$$z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq 0 \tag{3.4a}$$

$$z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) - \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \leq 1 \tag{3.4b}$$

$$z_{j_1 j_2}^k - \frac{1}{\mathcal{M}}(t_{j_2 q_2} - t_{j_1 q_1}) + \mathcal{M}(2 - x_{j_1}^{s_1} - x_{j_2}^{s_2}) \geq 0 \tag{3.5a}$$

$$z_{j_1 j_2}^k - \frac{1}{\mathcal{M}}(t_{j_2 q_2} - t_{j_1 q_1}) - \mathcal{M}(2 - x_{j_1}^{s_1} - x_{j_2}^{s_2}) \leq 1 \tag{3.5b}$$

$$z_{ij}^k - \frac{1}{\mathcal{M}}(t_{jq} - t_{ip}) + \mathcal{M}(2 - x_i^r - x_j^s) \geq 0 \tag{3.6a}$$

$$z_{ij}^k - \frac{1}{\mathcal{M}}(t_{jq} - t_{ip}) - \mathcal{M}(2 - x_i^r - x_j^s) \leq 1 \tag{3.6b}$$

In this study, aircraft are not allowed to pass one another once they are on a common route segment. As a result, the order of two flights at their transition onto a common segment is maintained. Consequently, the arrival times of two flights at all scheduling points on a common section are consistent with their sequence.

Specifically, Eqs. (3.4a) and (3.4b) together constrain the order of two arriving flights. Both equations become active when flight i_1 and flight i_2 are respectively directed to route r_1 and r_2 ($x_{i_1}^{r_1} = 1$ and $x_{i_2}^{r_2} = 1$), and r_1 has at least one common section

with r_2. p_1 and p_2 are physically the same scheduling point, but have different indices on route r_1 and r_2. In this case, the term $\mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2})$ becomes zero in Eqs. (3.4a) and (3.4b), and the remaining two terms on the left hand side become dominant. If arrival i_1 is ahead of arrival i_2 on their kth common section, flight i_1's STAs should always be earlier (smaller) than that of flight i_2 at the same locations. Therefore, in both equations, we have

$$t_{i_2 p_2} - t_{i_1 p_1} > 0 \tag{3.7}$$

Under this condition, Eqs. (3.4a) and (3.4b) together enforce

$$z_{i_1 i_2}^k \geq \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) \tag{3.8a}$$

$$z_{i_1 i_2}^k \leq \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) + 1 \tag{3.8b}$$

Since $\frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1})$ is a small positive number, the binary variable $z_{i_1 i_2}^k$ can only be one. This is the correct value, since $t_{i_2 p_2} - t_{i_1 p_1} > 0$ is valid. On the contrary, if i_2 is ahead of i_1, Eqs. (3.4a) and (3.4b) together can ensure that $z_{i_1 i_2}^k = 0$.

Similarly, departure sequences are constrained by Eqs. (3.5a) and (3.5b). Arrival-departure sequences can be determined using Eqs. (3.6a) and (3.6b).

3.2.3 Separation Requirements

In this work, separation requirements are enforced at all common scheduling points between two flights. Mathematically.

$$(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(3 - z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{3.9a}$$

$$-(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 + z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{3.9b}$$

$$(t_{j_2 q_2} - t_{j_1 q_1}) + \mathcal{M}(3 - z_{j_1 j_2}^k - x_{j_1}^{s_1} - x_{j_2}^{s_2}) \geq \delta(z_{j_1 j_2}^k, w_{j_1}, w_{j_2}, q_1, q_2) \tag{3.10a}$$

$$-(t_{j_2 q_2} - t_{j_1 q_1}) + \mathcal{M}(2 + z_{j_1 j_2}^k - x_{j_1}^{s_1} - x_{j_2}^{s_2}) \geq \delta(z_{j_1 j_2}^k, w_{j_1}, w_{j_2}, q_1, q_2) \tag{3.10b}$$

$$(t_{jq} - t_{ip}) + \mathcal{M}(3 - z_{ij}^k - x_i^r - x_j^s) \geq \delta(z_{ij}^k, w_i, w_j, p, q) \tag{3.11a}$$

$$-(t_{jq} - t_{ip}) + \mathcal{M}(2 + z_{ij}^k - x_i^r - x_j^s) \geq \delta(z_{ij}^k, w_i, w_j, p, q) \tag{3.11b}$$

where $\delta(\ )$ is a subroutine that accepts as inputs the sequence and their weight classes of two flights as well as the physical location of the scheduling point, and calculates the required separation time between the two flights at their common scheduling point. The same scheduling point may be denoted with different indices on different routes, such as p_1 and p_2 on route r_1 and r_2 respectively in Eqs. (3.9a) and (3.9b).

If arrival i_1 and i_2 are respectively directed to route r_1 and r_2 that have a few common scheduling points, then $x_{i_1}^{r_1} = 1$ and $x_{i_2}^{r_2} = 1$ in Eqs. (3.9a) and (3.9b). Meanwhile, if flight i_1 is ahead of i_2, $z_{i_1 i_2}^k = 1$. In this case, Eqs. (3.9a) and (3.9b) can be simplified to

$$(t_{i_2 p_2} - t_{i_1 p_1}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{3.12a}$$

$$-(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M} \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{3.12b}$$

where Eq. (3.12a) enforces the appropriate separation between $t_{i_1 p_1}$ and $t_{i_2 p_2}$. Since $\mathcal{M}$ is a large number, Eq. (3.12b) no longer affects the solution.

On the contrary, if arrival i_2 is ahead of i_1, then $z_{i_1 i_2}^k = 0$. The appropriate separation between $t_{i_1 p_1}$ and $t_{i_2 p_2}$ is now enforced by the simplified version of Eq. (3.9b). Similarly, Eqs. (3.10a) and (3.10b) are introduced to enforce departure-departure separations, whereas Eqs. (3.11a) and (3.11b) enforce arrival-departure and departure-arrival separations.

In practice, airborne separations are often measured by distances, which are affected by the wake vortices of leading aircraft, weather, visibility, as well as the weight classes of both aircraft involved. For the convenience of numerical solutions, these separation requirements are expressed in times. In comparison, the taxi speed of an aircraft is much

smaller than its airspeed. This low speed reduces the wake vortex effect of a leading aircraft on its trailing aircraft. As a result, the required separations on taxi routes are much smaller than runway and airborne separations.

3.2.4 Aircraft Performance Limitations

Due to aircraft performance limitations and operational regulations over the terminal airspace, the transit time of an aircraft between two neighboring scheduling points is constrained by the following equations.

$$t_{i(p+1)} - t_{ip} + \mathcal{M}(1 - x_i^r) \geq \tau_{\min}(w_i, p, p+1) \tag{3.13a}$$

$$t_{i(p+1)} - t_{ip} - \mathcal{M}(1 - x_i^r) \leq \tau_{\max}(w_i, p, p+1) \tag{3.13b}$$

$$t_{j(q+1)} - t_{jq} + \mathcal{M}(1 - x_j^s) \geq \tau_{\min}(w_j, q, q+1) \tag{3.14a}$$

$$t_{j(q+1)} - t_{jq} - \mathcal{M}(1 - x_j^s) \leq \tau_{\max}(w_j, q, q+1) \tag{3.14b}$$

In Eqs. (3.13a) through (3.14b), $(\tau_{\min}, \tau_{\max})$ represent the feasible range of aircraft transit times between two neighboring scheduling points. In this study, bounds on transit times are calculated from the distance between the two neighboring scheduling points and the speed range of the aircraft. The speed range of an aircraft depends on its weigh class (w_i or w_j) and its current location (p or q) in the terminal airspace.

The calculation of the performance index in Eq. (3.1) requires gate arrival times and fix arrival times. Below, Eqs. (3.15) through (3.18) are introduced to recognize that some times are equivalent.

$$t_i^{\text{gate}} = t_{iP}^r \tag{3.15}$$

$$t_j^{\text{fix}} = t_{jQ}^s \tag{3.16}$$

$$\hat{t}_i^{\text{gate}} = \hat{t}_{iP}^r \tag{3.17}$$

$$\hat{t}_j^{\text{fix}} = \hat{t}_{jQ}^s \tag{3.18}$$

Specifically, the gate is the last scheduling point on an arrival route. This is expressed by Eqs. (3.15) and (3.17). Similarly, Eqs. (3.16) and (3.18) indicates that the departure fix is the last scheduling point on a departure route. In these equations, P and Q are the last scheduling point of an arrival route and a departure route, respectively.

Finally, no time advance is allowed in the current study. Mathematically,

$$t_{ip} - \hat{t}_{ip} \geq 0 \tag{3.19}$$

$$t_{jq} - \hat{t}_{jq} \geq 0 \tag{3.20}$$

3.3 Solution Algorithms

While the above MILP formulation can be solved in its full version, the computational time under heavy traffic can be long. In order to systematically evaluate the tradeoffs between computational times and scheduling efficiencies, three different algorithms are introduced and compared in this study.

3.3.1 MILP Solutions with Optimal Route Selections

In this case, both the STAs and motion routes of arriving and departing traffic are determined via the MILP optimization process. Theoretically, the solution given by this algorithm is the optimal solution for a given runway scheduling problem. However, this full optimization approach typically requires the longest time to solve, and thus may not always be feasible for real time uses.

3.3.2 MILP Solutions with Fixed Routes

In this algorithm, motion routes of individual aircraft are pre-specified and are thus fixed in the optimization process. The MILP optimization only produces optimal STAs at the multiple scheduling points.

3.3.3 A First-Come-First-Served Comparison Algorithm

A First-Come-First-Served (FCFS) comparison algorithm is also developed to evaluate the above two algorithms. In the FCFS algorithm, it is assumed that each flight's route assignment is pre-specified. It then determines the flight sequences at merge points based on the FCFS principle. Finally, the STAs are determined based on minimal deviations from the ETAs subject to the constraints. Specifically, logics of the FCFS algorithm are as follows.

1. Combine and sort all flights in ascending order based on their ETAs at the first scheduling point and schedule each flight on a FCFS basis;
2. The first flight is permitted to travel on its assigned route without any delay;
3. Each remaining flight is scheduled to fly with the minimum deviation from its ETA while satisfying separation requirements and all other constraints.

Clearly, the FCFS algorithm may not perform as well as the other two algorithms in terms of delay reductions. On the other hand, it is easy to understand, and its logics strongly resemble what human controllers would do in practice. Thus, it is a good candidate algorithm to provide baseline solutions in evaluating the MILP algorithms.

3.4 Evaluation Criteria

Two criteria are used to evaluate the performances of the MILP-based optimization solutions and the FCFS solutions. The first criterion is the computational speed. Everything else being equal, a fast solution is preferred in practice. However, there are many factors that can affect the computational speed. These factors make a strict comparison of computational times different. For example, there are different ways to implement an algorithm using a computer language. In addition, the MILP formulation needs a "solver", which is a special software to solve the general MILP formulations. The

computational time of the MILP algorithms also depends on the efficiency of the solver. In comparison, the FCFS algorithm does not require a solver. Still, trends of computational times of different algorithms as the amount of traffic increases should reflect the inherent characteristics of the required computational times of these algorithms.

The second criterion is the performance of solutions in terms of the optimization index. In the examples below, a feasible solution with a smaller overall delay is regarded as a better solution. An optimal solution produces the minimum value of the performance index. It is important to recognize that different optimization indices may introduce different notions of being optimal. In addition, many practical considerations may not be easily expressed mathematically through optimization indices or constraints. While optimal solutions provide useful references for practical applications, they may not necessarily represent best solutions in the real world.

3.5 Simulation Studies

Coded in Java language, the three proposed algorithms are tested under different traffic demands. The solver for the MILP based algorithms is "Gurobi Optimizer", provided by Gurobi Optimization under an academic license. All codes are compiled and executed on a laptop with a 2.4GHz duo-core CPU and 4GB of memory.

3.5.1 Airport Model and Traffic Data

To solve a specific runway scheduling problem, the airport model, aircraft weight classes and the original schedules of all flights over a specified time window are needed. An flight's original schedule contains its ETAs at all scheduling points. Test scenarios are created using real traffic data from the JFK airport. These data has recorded individual flights schedules, which are converted into the original ETAs used in simulations. Runway 31L and 31R were active when traffic data was recorded.

A simplified airborne routes used in the traffic data are shown in Fig. 2.2. Three

arrival routes start from LENDY, CAMRN and ROBER. Four departure routes end at NORTH GATE, EAST GATE, WEST GATE and OCEAN GATE. It is assumed that flights from ROBER would land on 31L by default. In comparison, Runway 31L serves a mixed traffic of both arriving and departing flights.

Taxi routes on the surface are constructed in Fig. 2.3. They are also simplified somewhat for faster computations. Here, it is assumed that all arrival routes end at terminal T1. All departing traffic begins from terminal T2 and takes off from Runway 31L. Subsequently, they split at DP2 and each uses one of the four routes that end at the four departure gates.

An equal number of five scheduling points are defined on each route. These scheduling points include fixes, runways, and merge/diverge points, as well as a few additional scheduling points.

3.5.2 Test Scenarios

To test the performances of the different algorithms, a series of traffic scenarios ranging from 10 to 60 flights over the scheduling window of one hour are solved. In addition, the scenarios with 20, 40, and 60 flights are selected for discussions. They represent light, medium, and heavy traffic respectively. The initial traffic distribution in each scenario is shown in Table 3.2. The weight classes used in each scenario are given in Table 3.3. In this study, aircraft types include large, heavy and Boeing-757.

Table 3.2: Traffic demands and distributions

# of Flights \ Routes	A1	A2	A3	D1	D2	D3	D4
20 (Light Traffic)	4	4	2	2	3	3	2
40 (Medium Traffic)	5	9	6	5	6	7	2
60 (Heavy Traffic)	7	14	8	7	11	8	5

Table 3.3: Aircraft weight classes in simulations

# of Flights \ Types	Large	B757	Heavy
20 (Light Traffic)	11	0	9
40 (Medium Traffic)	24	0	16
60 (Heavy Traffic)	30	4	26

3.5.3 Separation Standards

As some previous studies,[15, 14, 28] this work expresses inter-aircraft separation in time. The separation time at a particular airborne location is computed by dividing the required separation distance by a nominal speed. For example, the required separation distance between two large aircraft is 3 nm near the fixes. The airspeed normally varies from 230 to 250 knots at these locations. Using an average speed of 240 knots as the nominal speed, the separation time is about 45 seconds. Adding a small margin, the separation requirement between two large aircraft is assumed to be 50 seconds near the fixes. Over the terminal airspace, aircraft speeds can vary between 100 to 250 knots. Typically, aircraft speeds are smaller when they are closer to the runways. The largest speeds are often recorded on the boundary of a terminal airspace, whereas smallest airspeeds occur during landing and taking off.

For the convenience of computations, it is assumed that the same separation time applies to the same aircraft pair at all airborne locations and on the runway. In comparison, the low taxi speed reduces the wake vortex effect of a leading aircraft on its trailing aircraft. As a result, the required separations on taxiways are much smaller than runway and airborne separations. Literature review produces few available standards for the minimum separation requirement between taxiing aircraft. Roling et al.[14] suggests 200m as a safe separation distance on taxiways. It also mentions that the average taxi speed ranges from 8m/s to 16m/s. The corresponding separation time varies from 12.5 seconds to 25 seconds. For simplicity, a 30-second separation requirement is used in the

current simulation studies. Airborne and runway separation requirements are listed in Table 5.4, whereas surface separation requirements are shown in Table 5.5.

Table 3.4: Airborne and runway separations (seconds)

Leader \ Follower	Large	B757	Heavy
Large	50	50	50
757	85	65	65
Heavy	85	85	65

Table 3.5: Surface separations (seconds)

Leader \ Follower	Large	B757	Heavy
Large	30	30	30
757	30	30	30
Heavy	30	30	30

3.5.4 Performance Index

The performance index shown in Eq. (3.1) is a fairly general expression that minimizes the weighted overall delay. In practical operations, airborne delays may cost more than ground delays when aircraft flights are considered on an individual basis. In this study, equal weighting is used for arrivals and departures for simplicity.

3.6 Results

Figs. 3.1 and 3.2 compare the overall delays and computational times of the three algorithms as the traffic volume increases. The trends are clear. At the traffic volume increases, both the overall delays and computational times by the three algorithms increase. Among them, the open-route MILP algorithm produces the least overall delays

but requires the largest computational times. In contrast, the FCFS algorithm produces the largest overall delays but requires the least computational times.

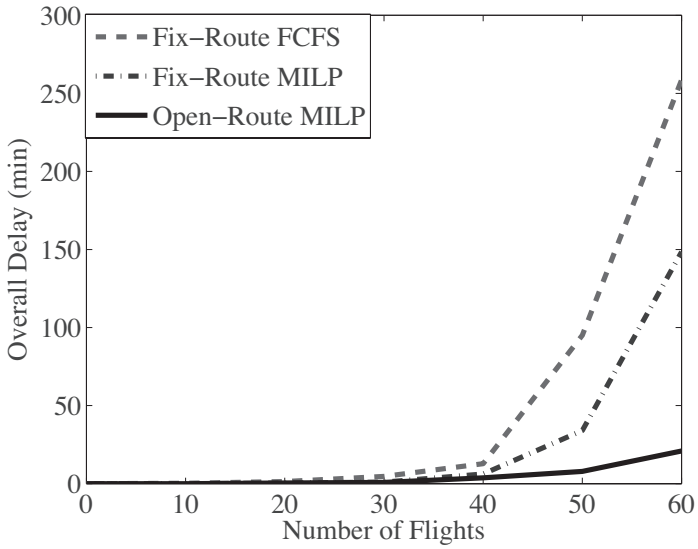


Figure 3.1: Overall delay vs. traffic volume.

Because runway thresholds are special scheduling points that can serve both arriving and departing traffic, Figs. 3.3-3.8 are created to display runway sequences and delays to gain an understanding of the performances of different algorithms. In Figs. 3.3-3.8, the two columns in the middle indicate the original runway assignments. ETAs of different flights are marked using relative times. Each mark indicates a specific weight class. Solid lines and dash-lines represent departures and arrivals respectively. Each solid line or dash-line connects a flight's runway ETA with its STA. In each of these figures, there are two runway STAs for each flight, which are produced by two different algorithms.

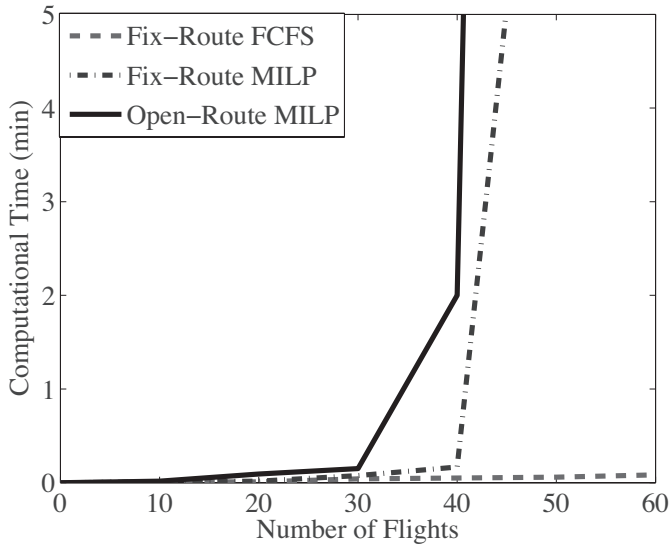


Figure 3.2: Computational time vs. traffic volume.

3.6.1 Light Traffic Scenario

In the light traffic scenario, 20 flights are scheduled over an hour. As shown in Fig. 3.1, the overall delay as well as individual delays introduced by either the FCFS algorithm, the fix-route MILP, or the open-route MILP are quite small. In Fig. 3.1, both MILP algorithms produce an overall delay about 0.4 minutes. In comparison, the FCFS algorithm produces an overall delay of 1.3 minutes. Most flights can arrive or depart on time using any of the three algorithms.

In this scenario, due to light traffic demand, all three algorithms maintain the original runway assignments. Figs. 3.3 and 3.4 indicate that the two MILP algorithms adjust some runway sequences to achieve smaller overall delays. For example, a heavy-large aircraft pair has been switched to a large-heavy pair, because the large-heavy pair requires a smaller separation time. In comparison, the FCFS algorithm maintains the original sequences.

The computational times of all three algorithms are equally small as shown in Fig. 3.2. A solution can be obtained in a few seconds. In this scenario, all algorithms can provide real time solutions.

3.6.2 Medium Traffic Scenario

In the medium traffic scenario, 40 flights are scheduled. Fig. 3.1 indicates that while the overall delay and individual delays are still small, the three algorithms start to exhibit differences in their performances. In Fig. 3.1, the open-route MILP algorithm provides an overall delay of 3.7 minutes, which is the smallest delay in this scenario. In comparison, the overall delay by the fix-route MILP algorithm is over 6 minutes. Finally, the FCFS algorithm introduces an overall delay of 12.5 minutes, which is the largest in this scenario.

The runway sequence plots in Figs. 3.5 and 3.6 show changes in runway sequences and runway assignments. Compared with the FCFS algorithm, the fix-route MILP algorithm swaps the order of a few flights on the runways to reduce the overall delay. In comparison, the open-route MILP algorithm not only adjusts the sequences, but also changes the runway assignments of two flights to balance the traffic loads on the runways; achieving the smallest overall delay.

Fig. 3.2 shows the computational times of the different algorithms. The FCFS algorithm is still the fastest, whereas both MILP algorithms require longer computational times. As expected, the open-route MILP algorithm is the slowest due to the large number of integer decision variables in its model. These integer variables can slow down the convergence speed of a MILP formulation.

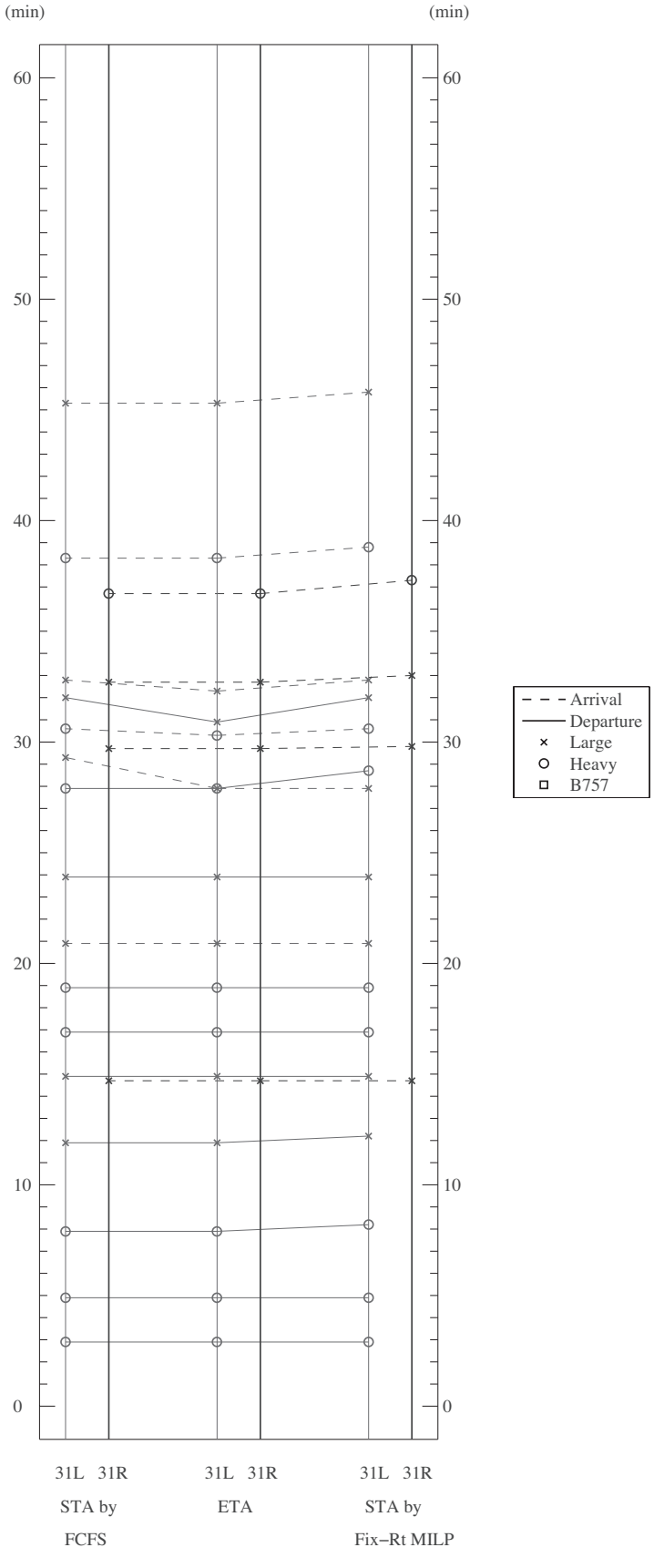
3.6.3 Heavy Traffic Scenario

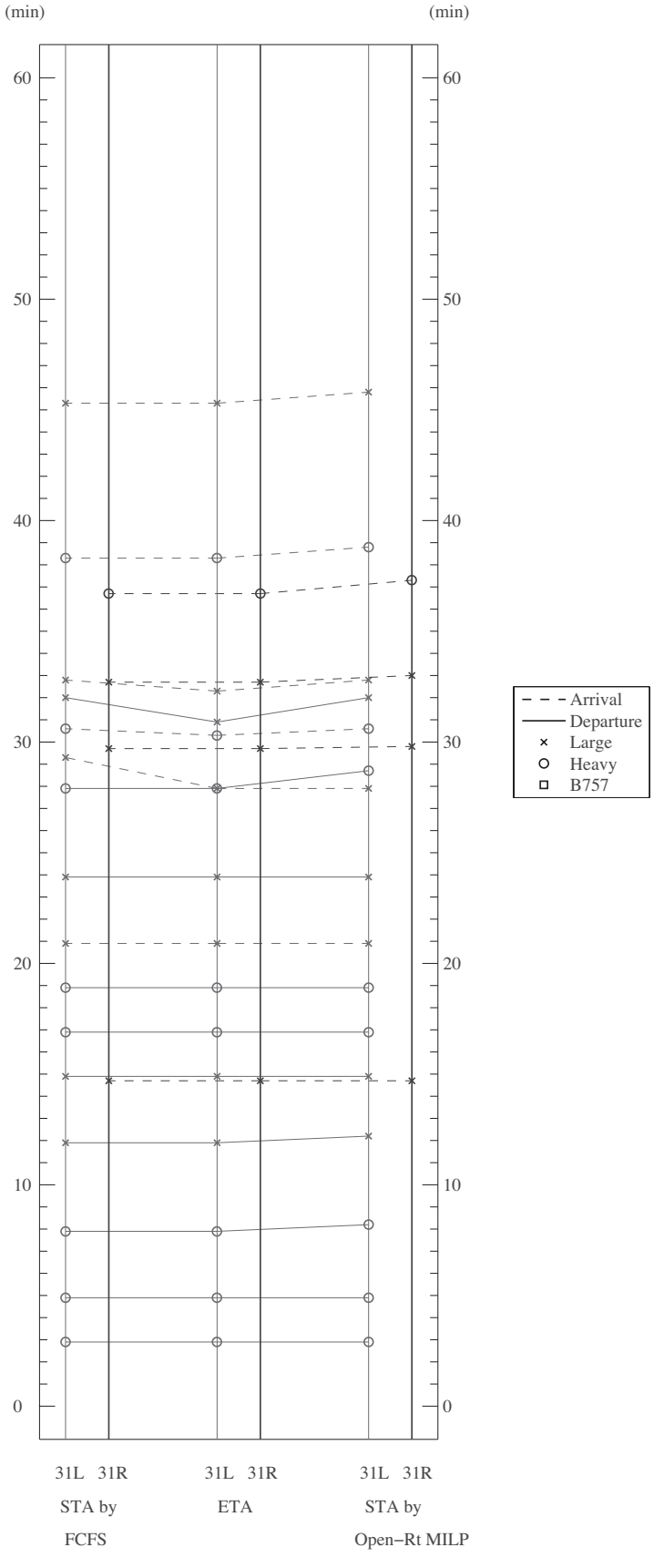
The heavy traffic scenario contains 60 flights over the one hour scheduling window. Fig. 3.1 shows that the overall delay introduced by the FCFS algorithm is around

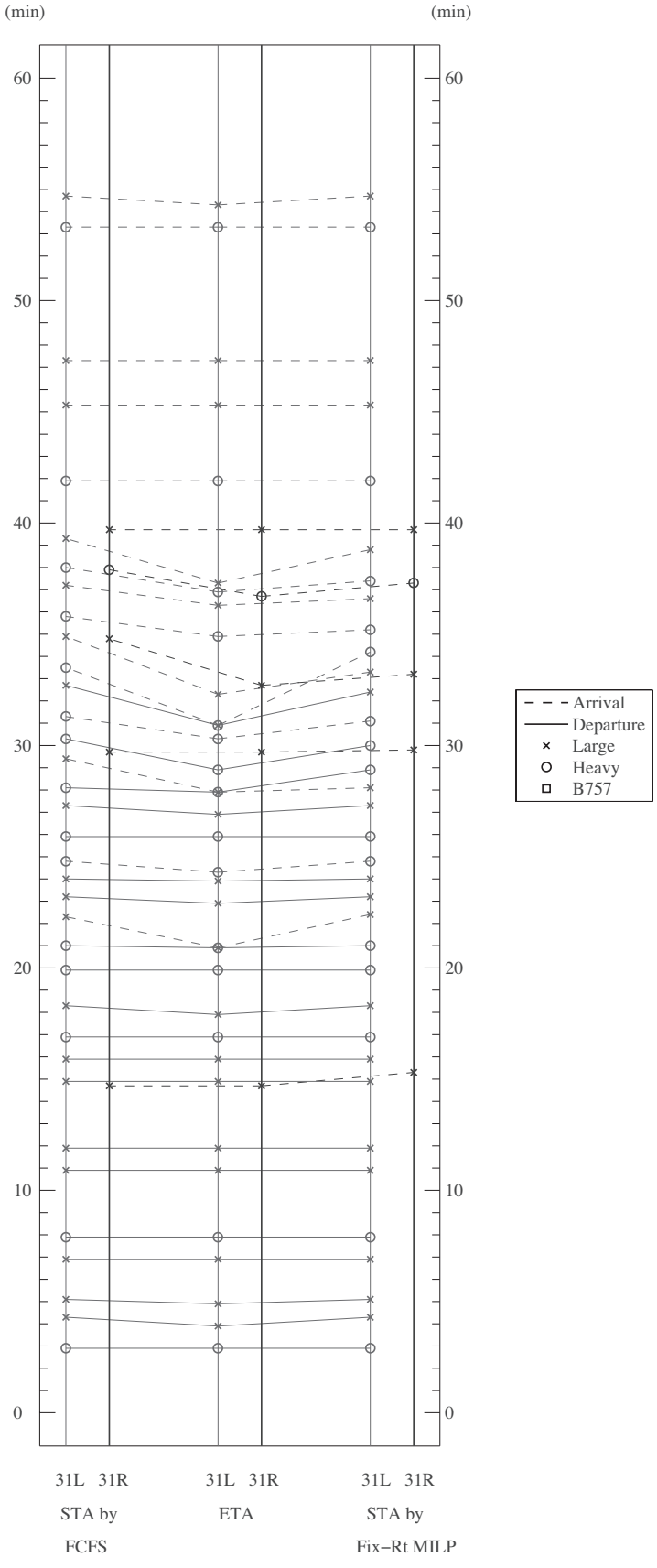
258 minutes, which is significantly larger than those by the MILP algorithms. The open-route MILP algorithm provides the best performance. Its overall delay is only 21 minutes, which is less than 10% of the FCFS solution. The fix-route MILP algorithm fares in between. In all cases, delays in the heavy traffic scenario are larger than those for the light and medium traffic scenarios.

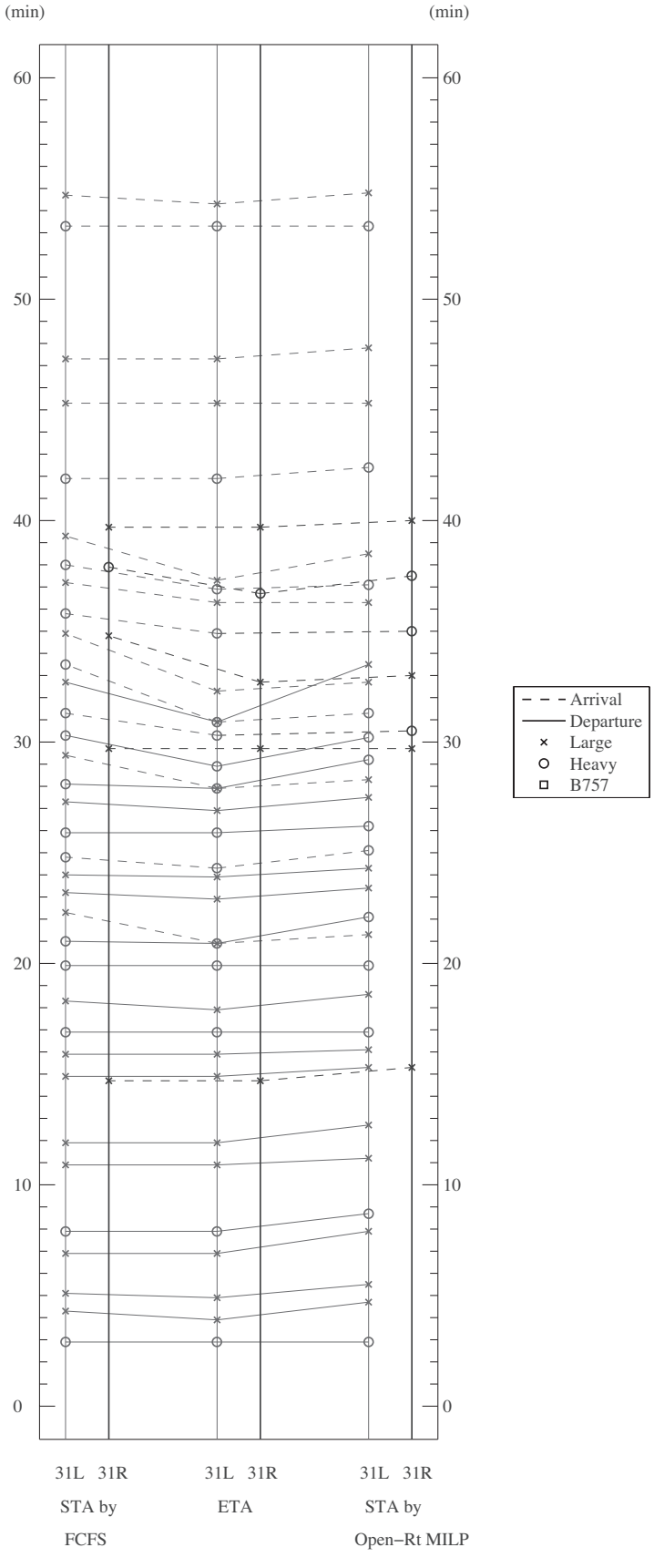
Figs. 3.7 and 3.8 indicate that the open-route MILP algorithm takes full advantage of the freedom in changing route assignments. It directs 14 flights from Runway 31L to 31R in order to release the congestion on 31L. Runway 31R now serves 21 flights, which is 3 times of its original traffic load. The open-route MILP algorithm imposes larger delays on some individual flights than other algorithms do, in order to achieve the smallest overall delay. The fix-route MILP algorithm cannot adjust runway assignment, but it can change sequences to reduce the overall delay. In comparison, the FCFS algorithm is not able to balance the traffic load on the runways or to adjust the sequences.

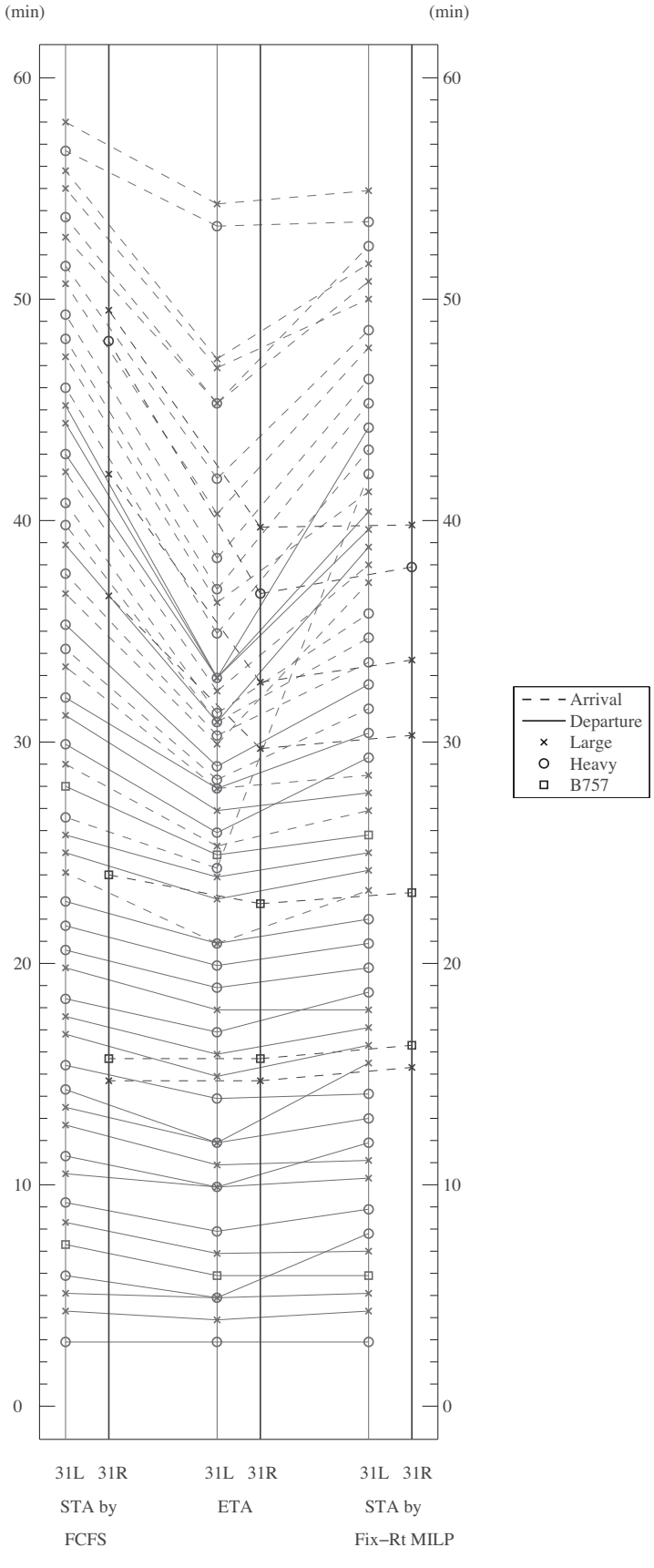
On the other hand, the open-route MILP algorithm requires about two hours of computational time to obtain solutions in the heavy traffic scenario, as shown in Fig. 3.2. Its slow convergence in obtaining solutions makes it impossible for real-time applications. The fix-route MILP algorithm takes about 50 minutes. This may still be infeasible for real-time applications. In comparison, the FCFS algorithm remains the fastest. In fact, its computational times do not change significantly for all test scenarios.

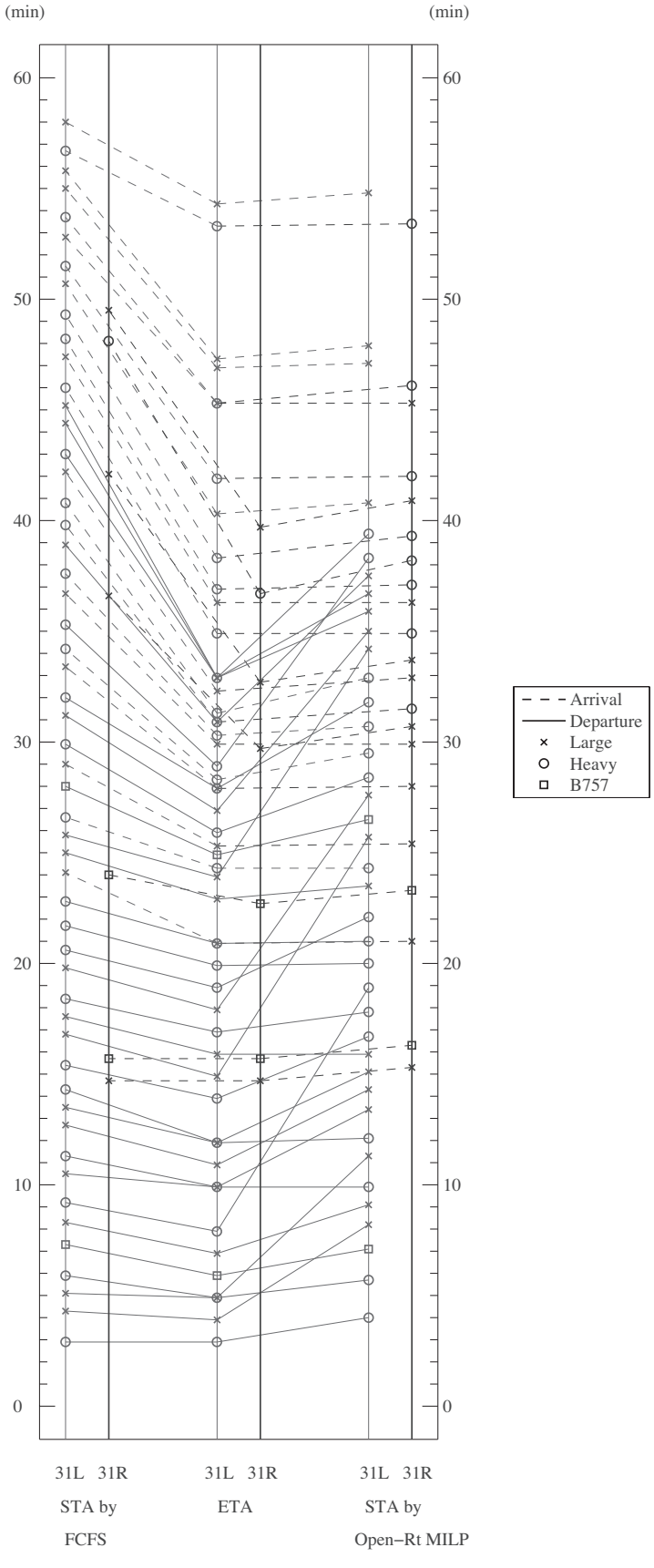

(min)
(min)
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Fix−Rt MILP


(min)
(min)
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Open−Rt MILP


(min)
(min)
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Fix−Rt MILP


(min)
(min)
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Open−Rt MILP


(min)
(min)
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Fix−Rt MILP


(min)
(min)
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Open−Rt MILP

Chapter 4

Dynamic Runway Scheduling

A multiple-point static runway scheduling scheme has been developed in Chap. 3 for the integrated routing and scheduling of arriving and departing traffic. It constitutes the foundation of the proposed dynamic strategies to be introduced in this chapter.

4.1 The Need for Dynamic Scheduling

While the proposed static runway scheduling scheme solves a fairly comprehensive runway routing and scheduling problem, its static nature limits its use in practical real-time applications. Specifically,

- In practical operations, especially during rush hours, arriving and departing aircraft over a terminal airspace form continual streams of traffic. The static algorithms, designed to schedule an isolated batch of traffic within a certain time interval, cannot easily handle continual traffic streams. In fact, the choice of the scheduling window size presents a challenge. Too small a scheduling window cannot accommodate the amount of traffic that are closely spaced, whereas too large a scheduling window can make the solution process excessively long.

- Even if a large scheduling window can be used for a specific application, there is no

guarantee that necessary computations can be completed within a specified time frame, especially in heavy traffic. During rush hours at busy airports, the traffic volume can increase drastically. The computational times of a static scheduling algorithm can increase significantly as the volume of traffic increases.

- Finally, optimal schedules are determined based on ETAs that are typically obtained from aircraft trajectory predictions. As the prediction look-ahead time increases, uncertainties in predicted trajectories increase and can limit the length of meaningful planning horizons using static scheduling algorithms. As a result, statically optimal schedules obtained over an overly large planning horizon are no longer reliable. Desirably, the optimal scheduling process should seek to take advantage of the newest traffic information available.

Therefore, a family of dynamic strategies are developed in this chapter to overcome the above disadvantages.

4.2 Structures of Sequential Dynamic Scheduling

The working time interval of a scheduling algorithm is called the planning horizon, which begins at a certain look-ahead time from now and extends indefinitely into the future. A dynamic scheduling strategy divides the planning horizon into a series of smaller time intervals, called scheduling windows. It then applies an appropriate static scheduling algorithm sequentially to each window. Sizes of the scheduling windows are selected to make sure that the static algorithms remain computationally efficient. A bound is imposed on the maximum computational time over each scheduling window so that feasible solutions can be obtained in real-time, although some solutions may be sub-optimal. In addition to the constraints used in the static scheduling algorithms, additional constraints are needed, called induced constraints, to ensure sufficient inter-aircraft separations among traffic in neighboring windows. As a result, different dynamic

strategies amount to different ways of dividing the planning horizon into smaller scheduling windows and establishing proper induced constraints, as well as a proper selection of the bound on the maximum computational time.

There is another challenge in the dynamic strategy design. Because of the multiple-point scheduling scheme, ETAs of a given aircraft at different scheduling points may fall into different scheduling windows. As a result, STAs of the same aircraft at some scheduling points may have been determined over a particular scheduling window, whereas its STAs at other scheduling points may still be open because their corresponding ETAs fall into future scheduling windows. Furthermore, STAs determined in previous scheduling windows may be revised in the current window. Therefore, the design of a dynamic strategy also includes the specification of open scheduling points for a given aircraft over a certain window.

In general, different design choices of a dynamic strategy can be made to improve the scheduling optimality and computational speed, under specific traffic conditions and operational requirements. Table. 5.1 summarizes key components of dynamic scheduling.

Table 4.1: Dynamic strategy parameters

Parameters	**Meaning**	**Options in Tests**
T_W	Scheduling window size	5-15 minutes
T_{adv}	Advance time	100%, 75%, and 25% of T_{w} Correspond to 0%, 25% 75% overlaps
T_C	Computational time for each window	A maximum of 90 seconds
Static algorithm	Computation algorithm to produce optimal schedules	Fix-route algorithm Open-route algorithm

For a given dynamic strategy, different solution methods can be used for sub-interval scheduling solutions. In this study, a MILP formulation is used. However, MIP models

generally lack theoretical proofs of bounds on computational complexity. Alternative approaches are available in the literature to solve the MILP formulation that have proofs of computational complexity[8, 33].

4.2.1 Time Relations in Dynamic Scheduling

There are two time concepts in dynamic scheduling: actual time and planning time. Actual time is the current clock time in the real world, whereas a planing time is an instant on the planning horizon that is later than the actual time. These time concepts are illustrated in Fig. 5.1, in which the vertical axis and horizontal axis represent the actual and the planning time respectively.

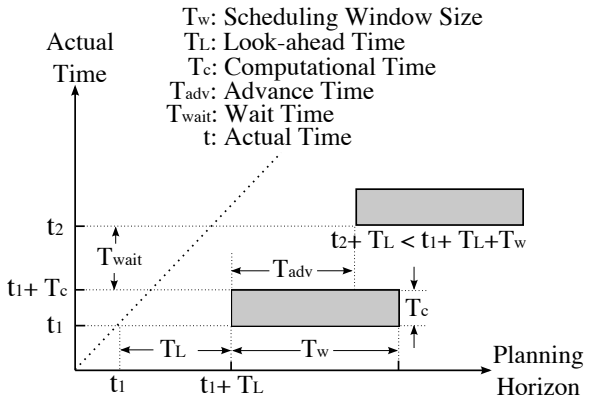


Figure 4.1: The timeline with overlapping scheduling windows.

At an actual time t_1, a dynamic strategy begins planning for future traffic that shall arrive at $t_1 + T_L$, where T_L is called the look-ahead time. The dynamic strategy first considers traffic over the scheduling window of $[t_1 + T_L, t_1 + T_L + T_W]$, where T_W is the size of the scheduling window. T_C represents the computational time in obtaining solutions over this window. After scheduling solutions for this window are obtained, the dynamic algorithm would wait until actual time t_2. It will then begin scheduling traffic that plans to use the airport over the window of $[t_2 + T_L, t_2 + T_L + T_W]$. This process

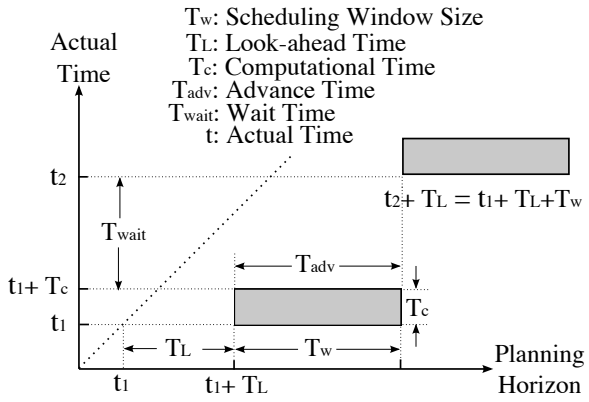


Figure 4.2: The timeline with non-overlapping scheduling windows.

is then repeated.

4.2.2 Choices of Scheduling Window Divisions

There are two general approaches to dividing the planning horizon into smaller scheduling windows: non-overlapping windows and overlapping windows. Fig. 5.1 illustrates the case when scheduling windows overlap, whereas Fig. 4.2 illustrates the non-overlapping scheduling windows. In either case, the advance time between the computations of neighboring scheduling windows is defined as

$$T_{\text{adv}} = t_{n+1} - t_n \tag{4.1}$$

where t_n is the beginning time of the nth scheduling window and $n = 1, 2, \cdots$. If the scheduling windows do not overlap, we have

$$T_{\text{adv}} = T_W \tag{4.2}$$

Otherwise,

$$T_{\text{adv}} < T_W \tag{4.3}$$

When computations for the scheduling window beginning at t_n are completed, computations for the subsequent scheduling window will not start until t_{n+1}. Assuming

that the total computational time for a window is T_C, the waiting time is given by

$$T_{\text{wait}} = T_{\text{adv}} - T_C \tag{4.4}$$

During the waiting time, the most recent traffic data may be acquired for subsequent computations. T_{wait} also serves as a safety buffer between two computations in case the preceding computation exceeds T_C. In any circumstance, we have

$$T_{\text{adv}} \geq T_C \quad \text{or} \quad T_{\text{wait}} \geq 0 \tag{4.5}$$

In the absence of traffic data uncertainties, the scheduling window size needs to be selected to strike a balance between scheduling optimality and computational speed. Intuitively, optimization over a larger scheduling window can produce better results than piecewise optimizations over a series of smaller windows. On the other hand, an overly large scheduling window may contain an excessive number of flights and thus requires a considerable computational time, which can prevent static scheduling algorithms from generating timely solutions. The presence of data uncertainties also practically limits the choices of scheduling window sizes.

Furthermore, proper choices of overlaps between neighboring windows can potentially improve the computational speed, especially when the scheduling window sizes are large. When two adjacent scheduling windows overlap, traffic in the overlapped segment can be optimized in both windows. In most cases, scheduling optimization over the second window only needs to apply small revisions to solutions obtained in the previous window. This often require less computation efforts. By contrast, two adjacent non-overlapping windows contain distinct flights. Each of them has to process a large number of new flights, which takes the static algorithm more time to obtain solutions.

In this study, effects of scheduling window sizes on scheduling performance and computational speed are studied. The differences between non-overlapping and overlapping windows are also examined.

4.2.3 Effect of Multiple Scheduling Points

In a multiple-point scheduling scheme, individual flights are scheduled at several locations over the terminal airspace that include fixes, runway thresholds, and gates. Due to the limited size of a scheduling window, a flight over the terminal air space may span several scheduling windows. As a result, it is possible that ETAs of a given flight at different locations fall into different scheduling windows. This creates a new problem that is absent in the static multiple-point scheduling scheme: which of the arrival times should be optimized in a given scheduling window.

Fig. 4.3 presents an example of how a dynamic strategy schedules a particular flight, **Flight01**, over two scheduling windows. When Window 1 is active, all five ETAs of **Flight01** are either in or after Window 1. Therefore, we allow the dynamic strategy to adjust all ETAs for the best performance. Correspondingly, five STAs are computed, which are then treated as the new ETAs of **Flight01** in subsequent scheduling calculations. When Window 2 becomes the current scheduling window, ETA1 and ETA2 (STA1 and STA2 in Window 1 respectively) are located in Window 1 and are therefore locked by the dynamic strategy. On the other hand, ETA3 (STA3 in Window 1) is now in Window 2 and can still be adjusted by the dynamic strategy. Similarly, ETA4 and ETA5 can also be adjusted for the best performance in Window 2. In this study, the dynamic strategy locks ETAs that are located in previous scheduling windows. It only optimizes or adjusts ETAs that are located in the active scheduling window or in future windows. This technique ensures the schedule stability for the immediate future while it retains schedule flexibility for the longer term.

4.2.4 Induced Constraints

To ensure sufficient separations, locked ETAs of a given flight are delivered to the static algorithm as separation constraints. Other constraints contained in the MILP formulation also remain valid. Furthermore, dynamic scheduling over multiple windows can

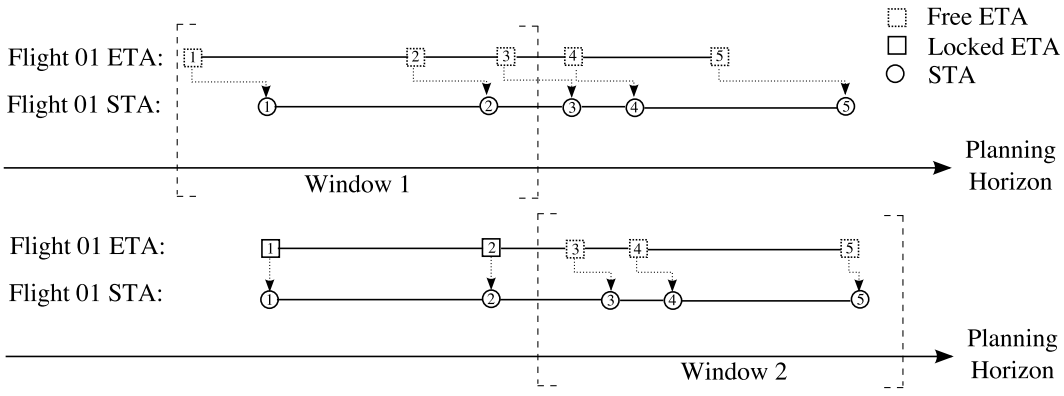


Figure 4.3: Multiple-point scheduling over multiple scheduling windows.

potentially result in separation violations between flights located in adjacent windows because of the sequential nature. To eliminate any potential cross-window conflict, the ETAs of a sufficient number of flights towards the end of the preceding window are also included in the scheduling optimizations during the active window as separation constraints. The static algorithm applied over the active window can then sufficiently separate flights in adjacent windows.

4.3 Simulation Evaluations

A numerical simulation environment is developed to evaluate the performances of various dynamic scheduling strategies. All simulation programs are coded in Java. Lp_Solve[34], an open-source MILP solver, is used for the static runway scheduling.

Figs. 2.2 and 2.3 show the airborne routes and surface routes used in this work. It is assumed that the arriving traffic streams enter the airport through three fixes: LENDY, CAMRN and ROBER. Flights from ROBER land on Runway 31R by default, whereas flights from LENDY and CAMRN use Runway 31L instead. All departures take off from Runway 31L and exit the terminal airspace through NORTH GATE, EAST

GATE, WEST GATE and OCEAN GATE.

Three aircraft classes are considered in the simulation studies: large, heavy and Boeing-757. Different separation requirements can apply to the same aircraft pair depending on if they are on the ground or in the air. The airborne and surface separation requirements are listed in Tables 5.4 and 5.5, respectively. All separation requirements are expressed in time.

Table 4.2: Airborne and runway separations (seconds)

Leader \ Follower	Large	B757	Heavy
Large	50	50	50
757	85	65	65
Heavy	85	85	65

Table 4.3: Surface separations (seconds)

Leader \ Follower	Large	B757	Heavy
Large	30	30	30
757	30	30	30
Heavy	30	30	30

The computational time and the overall delay are used as performance criteria to evaluate different dynamic strategies. Eq. (3.1) shows the overall delay of a dynamic scheduling strategy. It contains the total gate delay for arrivals plus the total fix delay for departures. Besides, both the average and the maximum computational time are considered. The average computational time of a dynamic strategy is obtained by taking the average of computational times used in the series of scheduling windows. By contrast, the maximum computational time represents the largest computational time on a single scheduling window over all the scheduling windows. When the maximum computational time bound is reached at a particular window, the solution process is terminated and the best feasible solutions up to this time are reported.

In this study, a dynamic strategy that produces a smaller overall delay is regarded as more effective. It is important to recognize that different optimization criteria define different notions of being effective. Furthermore, many practical considerations may not be easily expressed mathematically through optimization criteria or constraints. While optimal solutions provide useful references for practical applications, they may not necessarily represent best solutions in the real world.

Solution robustness to data uncertainties is also a desirable property in practical operations. Indeed, ETAs delivered to the scheduling process likely contain errors. In this regard, sequential dynamic strategies are *structurally* robust because they use updated information over each scheduling window. Designs of dynamic strategies for achieving a desired level of robustness shall be considered in the future.

4.4 Results on Scheduling Window Sizes and Overlaps

Dynamic strategies with different window sizes and overlap sizes are first tested. Dynamic strategy parameters are summarized in Table 5.1. The two general static scheduling algorithms developed in Chap. 3, i.e. the fix-route and the open-route algorithm, are used as the underlying static solution algorithms. The fix-route algorithm produces optimal arrival times and flight sequences only, whereas the open-route algorithm also produces optimal route assignments, in addition to optimal arrival times and flight sequences,

Ref. 34 indicates that using the static multiple-point scheduling scheme, a planning horizon longer than one hour can no longer be solved in a few minutes under heavy traffic. By contrast, the dynamic strategy of this study can process traffic over a much longer planning horizon using the rolling window strategies. While exact computational times can vary with implementations of the algorithms and computers, the general idea remains true. The remainder of this section presents performances of various rolling window strategies in processing traffic over a 2-hour horizon.

The first traffic scenario in Table 4.4 is used to test dynamic strategies with scheduling windows ranging from 5 minutes to 15 minutes. This scenario contains a two-hour traffic and represents initially imbalanced traffic on Runway 31L and 31R. Three representative overlap sizes of 0%, 25%, and 50% are tested. A 90-second bound on the maximum computational time is imposed on each scheduling window.

Table 4.4: Traffic demands and distributions

Scenario	Hour	Arrivals	Departures	Rwy 31L	Rwy 31R
Imbalanced traffic	1	42	14	46	10
	2	49	16	54	11
Fairly balanced traffic	1	42	14	27	29
	2	49	16	37	28

4.4.1 Dynamic Strategies with Non-Overlapping Windows

Compared with the use of overlapping windows, the use of non-overlapping windows over a given planning horizon intuitively requires fewer computations to process the same amount of traffic. A larger advance time provides a longer computational time for obtaining solutions. On the other hand, each of the non-overlapping windows contains different traffic, except for flights that are scheduled over multiple windows at the multiple scheduling points. As a result, optimal schedules obtained in previous windows may not be used as initial guesses in subsequent windows.

Figs. 4.4 and 4.5 show the computational times for non-overlapping windows from 5 to 15 minutes. Under the same traffic conditions and a given static algorithm, a smaller scheduling window contains fewer flights and runs faster in most cases. For example, the average computational time with the 5-minute window is only one third of that with the 15-minute window. The maximum computational times in Fig. 4.5 follow a similar trend. Compared with the average computational time, the maximum computational

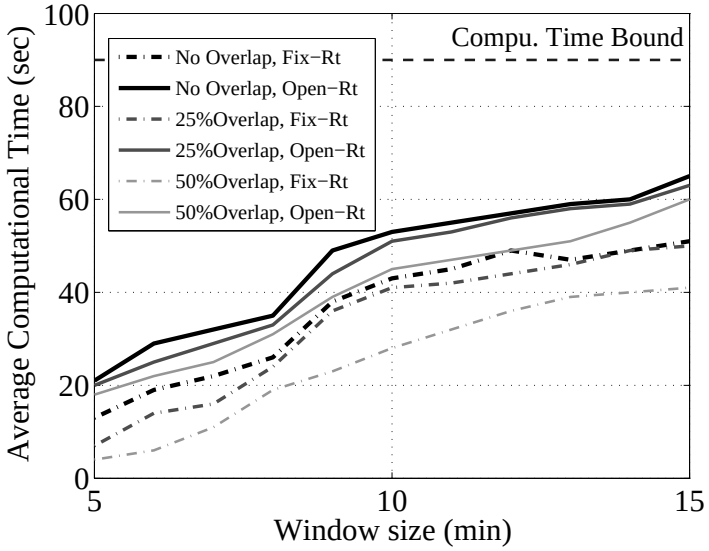


Figure 4.4: Average computational times with various dynamic strategies.

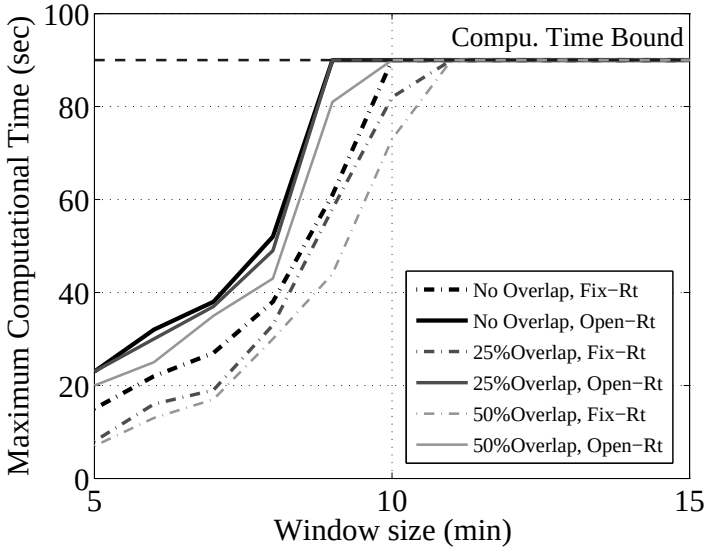


Figure 4.5: Maximum computational times with various dynamic strategies.

time increases more drastically as the windows become larger. With either static algorithm, the 90-second bound starts to limits the maximum computational time as the window size exceeds 10 minutes.

The two static algorithms demonstrate different levels of efficiency. With a given window size, the fix-route algorithm is always faster than the open-route algorithm, as shown in Figs. 4.4 and 4.5. This is consistent with results in the previous studies of static runway scheduling[**?**]. As the window size varies, the open-route algorithm reaches the 90-second bound before the fix-route algorithm does (Fig. 4.5). In these tests, the average computational time using the fix-route algorithm is always 6-8 seconds less than that using the open-route algorithm. Trends of the maximum computational time are similar before reaching the 90-second bound.

Before the maximum computational times reach the 90-second bound, the overall delay and the window size vary inversely as shown in Fig. 4.6. A dynamic strategy with larger window sizes produces less delays. As soon as the bound on the maximum computational time is reached, on the other hand, this inverse relation between the overall delay and the window size no longer holds. Instead, further increase in the window size produces more delays. This is caused by the suboptimal nature of solutions when computations are terminated at 90 seconds. Computational loads on the static algorithms grow as the scheduling window size increases further, and these increased computational loads degrade the solution optimality.

4.4.2 Dynamic Strategies with Overlapping Windows

Intuitively, overlaps between neighboring scheduling windows may potentially speed up the computations, especially for large windows. This is because for two adjacent windows, schedules of the overlapping traffic that have been optimized in the previous window serve as good initial guesses for fine tuning in the subsequent window. This fine tuning can be completed in a shorter time.

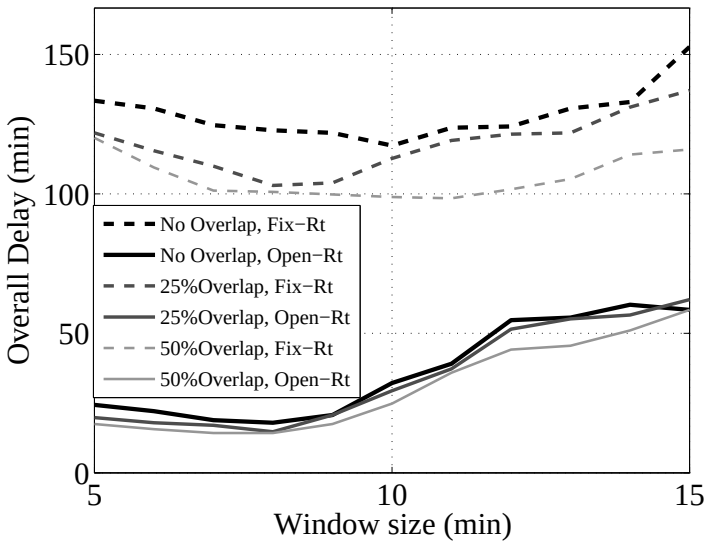


Figure 4.6: Overall flight delays with various dynamic strategies.

As shown in Figs. 4.4 and 4.5, a 25% overlap between neighboring windows can reduce both the average and the maximum computational times compared with non-overlapping windows. Specifically, the 25% overlap can reduce the average computational time by 2-5 seconds using the open-route algorithm. The reduction is even larger with the fix-route algorithm. It also permits the use of a larger window size before the fix-route algorithm reaches the 90-second bound.

In terms of the overall performance, dynamic strategies with 25% overlaps follow somewhat similar trends as those with non-overlapping windows. Before the maximum computational time reaches the 90-second bound, an inverse relation again exists between the overall delay and the window size as shown in Fig. 4.6. The overall delays with the fix-route algorithm can be reduced substantially if windows are less than 10 minutes. This reduction however diminishes as the window size increases further, as the bound on the maximum computational time starts to become constraining. On

the other hand, the 25% overlap does not provide the open-route algorithm with equal reductions in overall delays. Fig. 4.6 shows that the overall delay would no longer decrease once the maximum computational time reaches the 90-second bound. It even starts rising if the scheduling window size increases further.

Reasonable increase in the overlap can improve both the efficiency and the overall performance. As shown in Fig. 4.4, a 50% overlap between neighboring windows can substantially reduce the average computational times of both static algorithms, especially when window sizes are large. Besides, Figs. 4.4 and 4.5 indicate that the fix-route algorithm results in more reductions in computational times than the open-route algorithm. The 50% overlap also produces reductions in the overall delay, as shown in Fig. 4.6. In particular, the fix-route algorithm results in greater reductions than the open-route algorithm, especially when window sizes are greater than 10 minutes. For overlap sizes within a certain range, the fix-route algorithm is faster than the open-route algorithm but the open-route algorithm produces better scheduling performances than the fix-route algorithm. Overly large overlaps result in redundant computations that may adversely affect the performances of a dynamic strategy.

Overall, the smallest overall delay occurs with a 8-minute window size, a 50% overlap and the use of the open-route algorithm. Reducing the overlap to 25% results in a slight increase in the overall delay. On the other hand, the fastest dynamic strategy employs 5-minute scheduling window, a 50% overlap, and the fix-route algorithm.

4.5 Benefits of Optimal Sequencing and Runway Assignments

To evaluate the benefits of optimal sequencing and runway assignments, four levels of scheduling optimization options are constructed in Table 4.5. A basic First-Come, First-Served (FCFS) scheduling principle is introduced for comparison purposes. In

the FCFS Option, the original runway assignments are maintained and flight sequences are determined on a FCFS basis at merge points. By contrast, the Sequencing Option allows aircraft to switch their sequences at merge points while maintaining their original flight runway assignments. This is achieved by the fix-route algorithm. Next, the Routing Option generates optimal runway and route assignments for individual aircraft while maintaining their original flight sequences at the merge points on a FCFS basis. This option is essentially a tradeoff between the fix-route and the open-route algorithm. Finally, the Comprehensive Scheduling Option permits both flight sequence changes and runway assignment adjustments. This option is equivalent to the open-route algorithm and provides the maximum freedom in runway scheduling. The Comprehensive Scheduling Option is expected to offer the best performance.

Table 4.5: Runway scheduling options to be tested

Option	**Runway Assignment Rule**	**Sequencing Rule**
FCFS	Default Runway Assignments	FCFS Sequences
Sequencing	Default Runway Assignments	Optimal Sequences
Routing	Optimal Runway Assignments	FCFS Sequences
Comprehensive Scheduling	Optimal Runway Assignments	Optimal Sequences

A representative dynamic strategy is assumed, which uses a series of 10-minute long, non-overlapping scheduling windows. This dynamic strategy provides a balance between computational speed and scheduling effectiveness for the two traffic scenarios listed in Table 4.4, which emulate different operational environments. In fact, both scenarios correspond to an arrival rush in the JFK airport but represent different initial traffic loads on Runway 31L and 31R. Specifically, the first scenario represents an imbalanced initial traffic on Runway 31L and 31R. It is used to test the ability of the dynamic strategy in balancing the traffic loads on the two runways. By contrast, the second scenario contains fairly balanced initial traffic loads on Runway 31L and 31R. It is used as a comparison case to study the increase of computational efforts when initial traffic

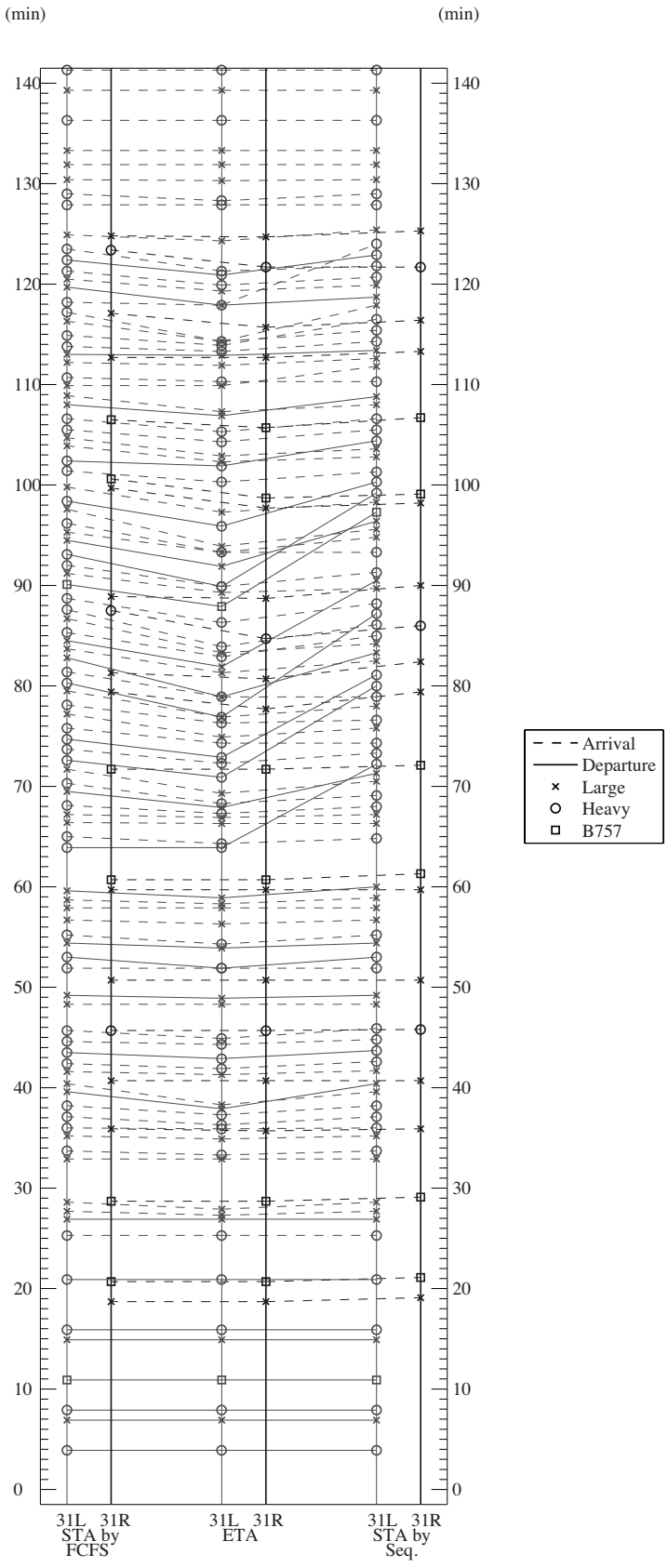
loads are imbalanced as in the first traffic scenario.

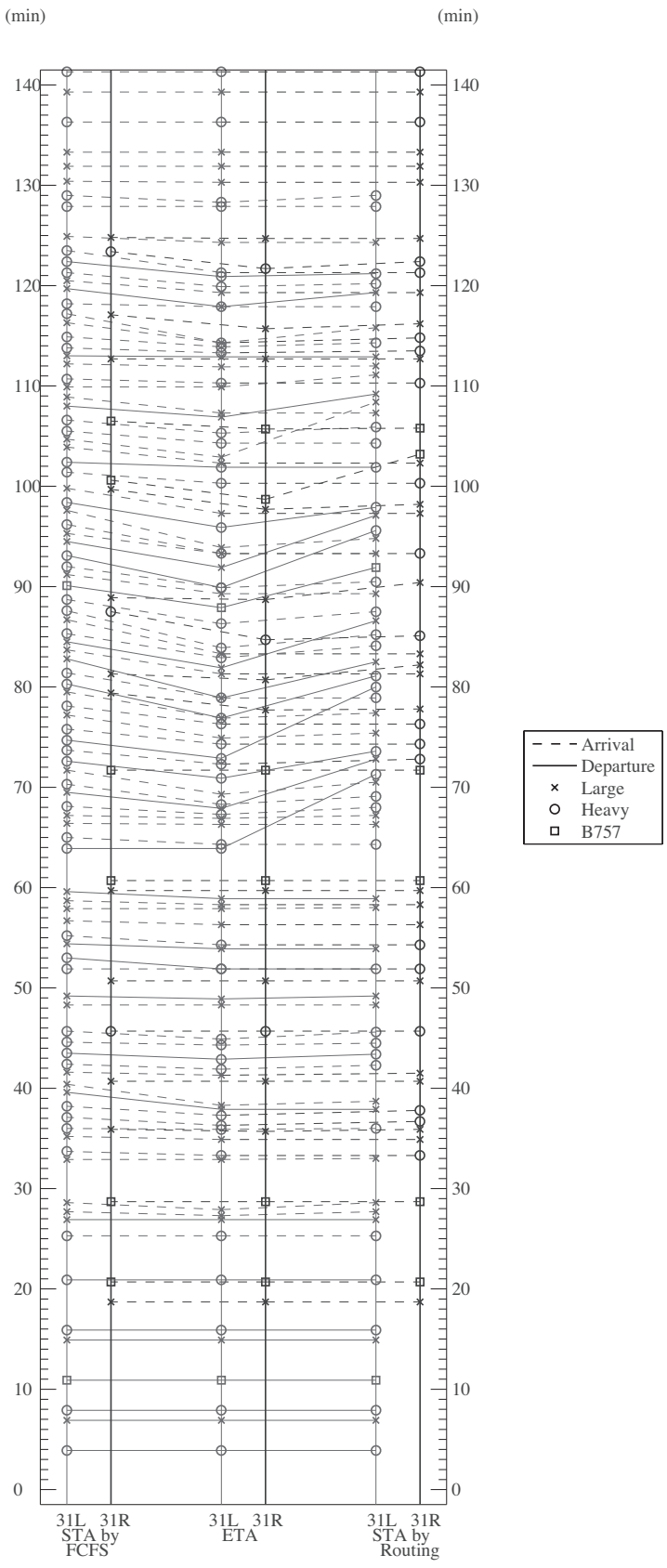
Sequence plots as shown in Figures 4.7-4.15 are devised to display the solutions. Each sequence plot shows individual flights' runway ETAs, STAs, and flight sequences on the runways. In addition, a sequence plot also expresses individual flights' runway delays and their sequence adjustments. In each sequence plot, the two vertical lines in the middle represent Runway 31L and 31R. Each mark on these lines represents the arrival/departure time of a particular aircraft that either lands on or takes off from the corresponding runway. Different types of marks stand for different types of aircraft.

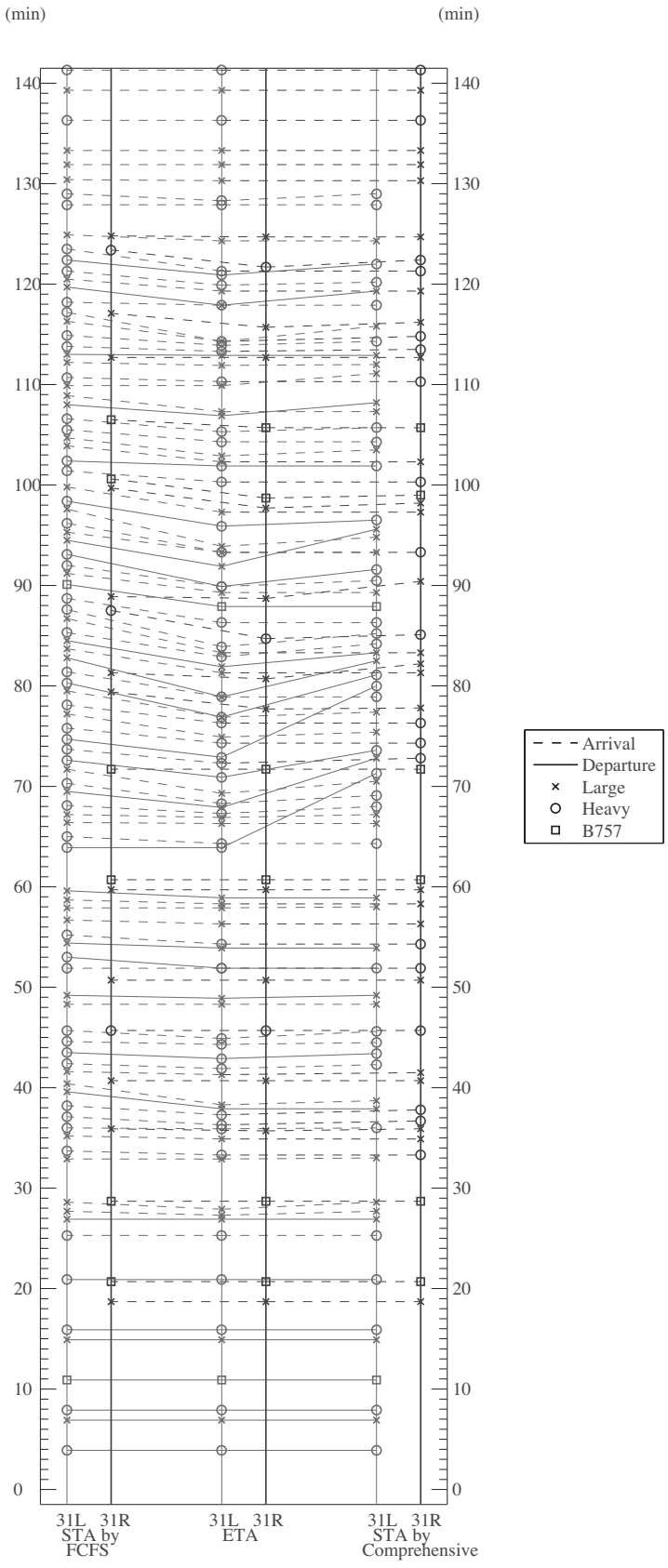
Besides the two vertical lines in the middle, there are two lines on the right and two on the left. These two sets of vertical lines display the STAs produced by two different runway scheduling algorithms, such as the FCFS method and dynamic scheduling with sequencing as shown in Fig 4.7. Similarly, the marks on each vertical line display the STAs and new sequences generated by the dynamic scheduling process. By connecting a flight's ETA and its STA in the figure, it can be clearly seen if the aircraft is delayed or not in the scheduling process. By completing all lines connecting ETAs and their corresponding STAs, one can tell if the flight sequences or runway assignments have been changed. In sequence plots, runway STAs (runway arrival/departure times) are expressed in relative times instead of absolute times. For example, if the scheduling horizon begins at 18:00, the 0 minute on the plot corresponds to 18:00, and 60 minutes corresponds to 19:00.

4.5.1 Results for Imbalanced Traffic

This scenario simulates an initially imbalanced traffic on Runway 31L and 31R (Table 4.4). As shown in Fig. 4.7 and 4.17, neither the FCFS Option nor the Sequencing Option improves the traffic imbalance on the two runways. Also, no arrival traffic on Runway 31L is directed to Runway 31R, which has the potential to serve more flights. In attempting to relieve the congestion on Runway 31L, the dynamic strategy can only


(min)
(min)
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Seq.


(min)
(min)
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Routing


(min)
(min)
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Comprehensive

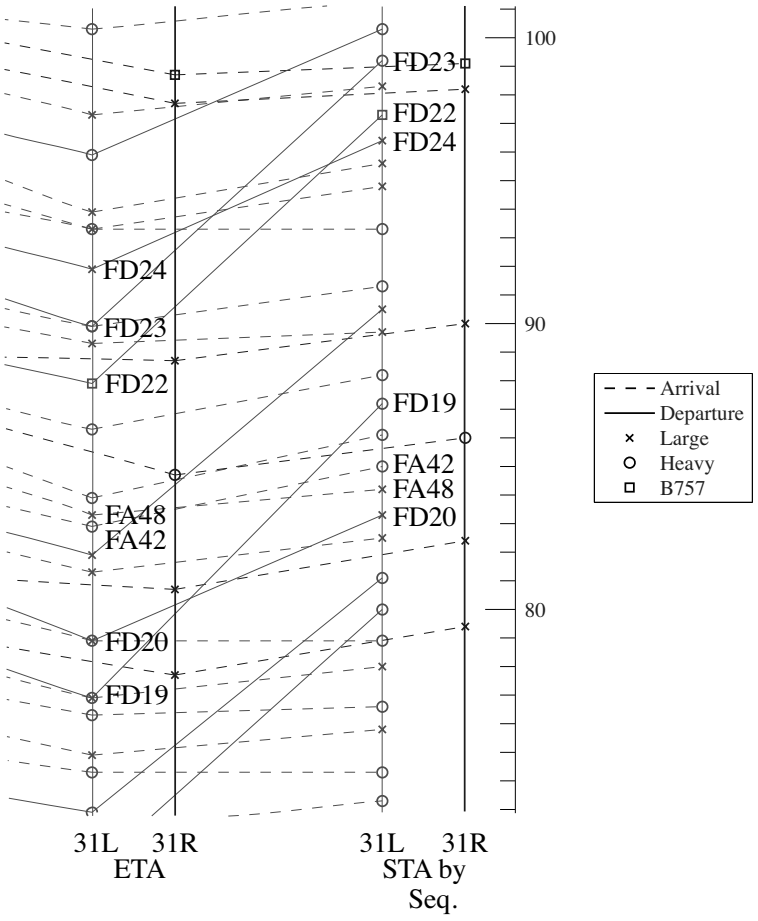


Figure 4.10: Sequencing option with imbalanced traffic (zoom in).

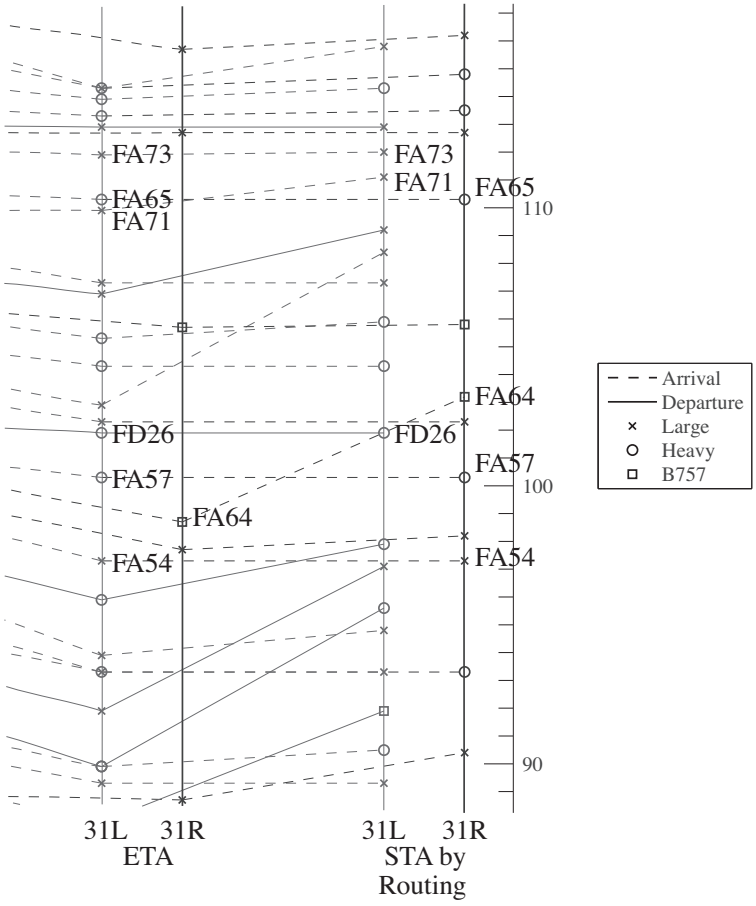


Figure 4.11: Routing option with imbalanced traffic (zoom in).

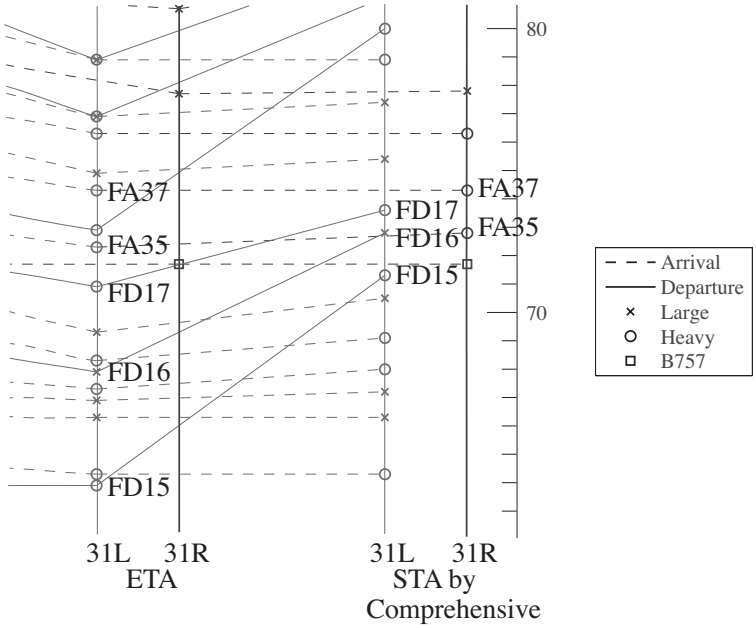


Figure 4.12: Comprehensive scheduling option with imbalanced traffic (zoom in).

adjust the flight sequences on both runways. As a result, several departure flights are held on the ground a little longer so that Runway 31L can serve arrivals first.

For example, departures FD22, FD23 and FD24 are held on the ground in order to accept some arriving flights at the same time period (Fig. 4.10). The three aforementioned departing flights are delayed and then automatically inserted into vacancies in the queue on Runway 31L. Furthermore, FD22 and FD24 are also swapped in the queue. Because FD22 is a B-757, less separation is required when it departs behind the large aircraft FD24 than the other way around. This swap reduces the runway occupancy time for this aircraft pair. In another example, FD20 is delayed by about 3 minutes and then inserted into the vacancy before FA48. This new arrangement reduces the delays imposed on FA42, FA48 and FAD20. Meanwhile, FA42 and FA48 are also swapped for a smaller separation.

These results indicate that the dynamic strategy with the Sequencing Option is able to adjust the flight sequences intelligently. It can properly insert flights into the vacancies of the queues to incur less delays on other flights. The overall performance is improved by using sequence adjustments as shown in Fig. 4.16. Compared with the FCFS Option, the Sequencing Option can reduce the overall delay from over 110 minutes to around 75 minutes.

Next, the Routing Option is tested to examine the benefits of optimizing runway assignments. In this simulation setup, flights originally planned to land on Runway 31L are permitted to land on Runway 31R. On the other hand, departures retain their default runway assignments. Since Runway 31L has a heavier traffic load due to the mixed operations, the dynamic strategy is expected to intelligently balance the traffic loads on Runway 31L and 31R to reduce the overall flight delay. As shown in Fig. 4.8, sequences on both runways are mostly maintained, with some overflow traffic directed to Runway 31R. Fig. 4.17 shows that Runway 31R now serves 28 more flights than in the original plan. An average of 14 runway assignment changes per hour is used to

reduce the congestion on Runway 31L.

Fig. 4.11 provides examples of runway assignment adjustments. Arrival FA65, which is originally right behind FA71 with insufficient separation, is directed to Runway 31R with no delay imposed. Also, FA73 can land right after FA71 without a significant delay. Meanwhile, FA57 and FA54 are also directed to Runway 31R to avoid delays. FD26 takes off on time also; benefiting from the new runway assignment of FA57.

In terms of the overall flight delay, the Routing Option is more efficient than the Sequencing Option as shown in Fig. 4.16. The overall delay with the Routing Option is only around 60 minutes, which is 15 minutes less than the Sequencing Option and 40 minutes less than the FCFS Option. As illustrated in Fig. 4.17, a total of 28 flights are directed from Runway 31L to Runway 31R, which produces a ratio of 72:49 between the number of flights on Runway 31L and 31R. These results suggests that the Routing Option is more efficient than both the FCFS Option and the Sequencing Option.

Finally, the Comprehensive Scheduling Option permits adjustments in both flight sequences and runway assignments. As shown in Fig. 4.9, necessary sequence adjustments are conducted along with traffic balancing. Many arrivals are directed to Runway 31R to release the congestion on Runway 31L, and new sequences are produced. Similarly as in the Routing Option, Fig. 4.9 and Fig. 4.17 indicate a total of 28 runway assignment changes. Fig. 4.16 indicates that the congestion on Runway 31L has been greatly released and the overall flight delay is further reduced to 36 minutes, which is the least among all the Options.

Fig. 4.12 offers examples of simultaneous traffic balancing and sequencing. For instance, FA37 is directed to 31R to avoid delays due to the separation requirement between FA37 and FD17. Besides, FD15, FD16 and FD17 are held on the ground until vacancies in the arrival queue appear. The heavy aircraft FD15 is sequenced behind a large arrival aircraft so that a smaller separation between them applies. The new runway assignment for FA35 not only resolves the conflicts among FD15, FD16 and

FD17, but also imposes the minimum delays on these flights.

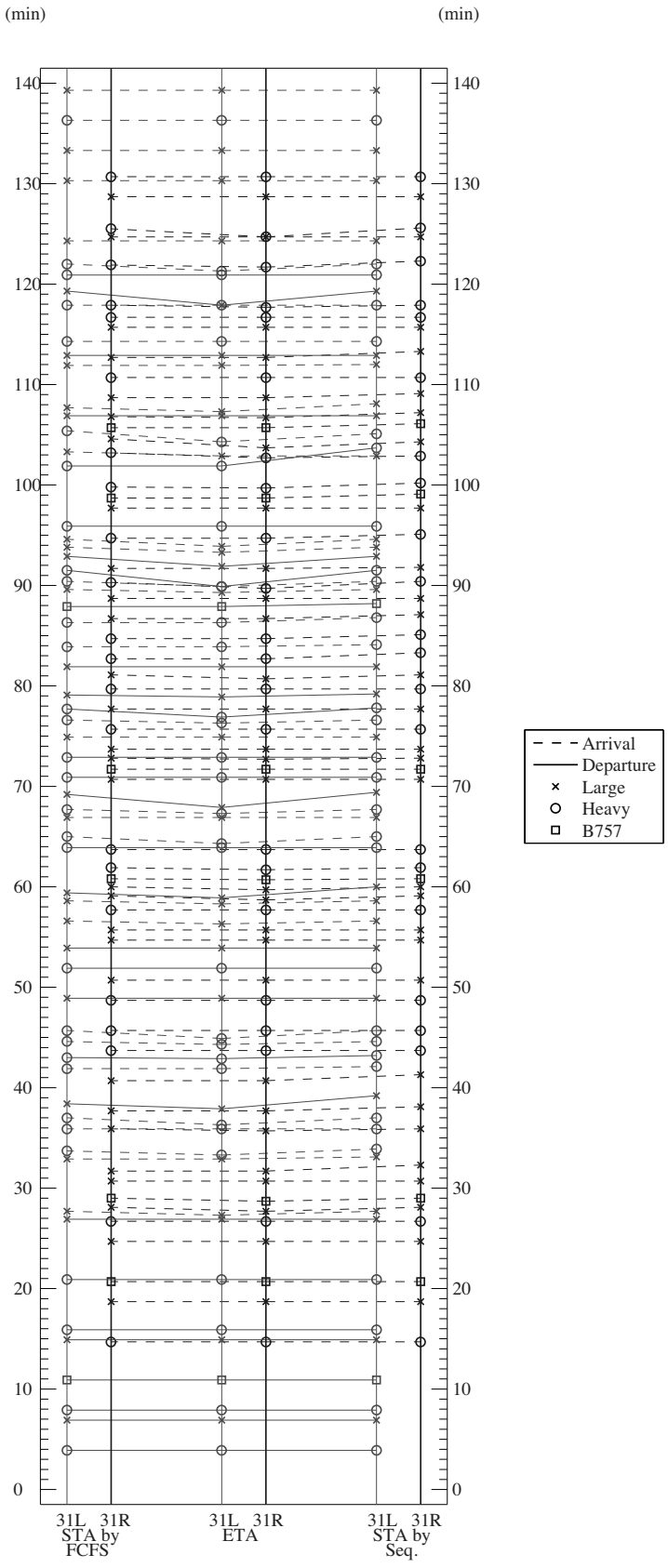
4.5.2 Results for Fairly Balanced Traffic

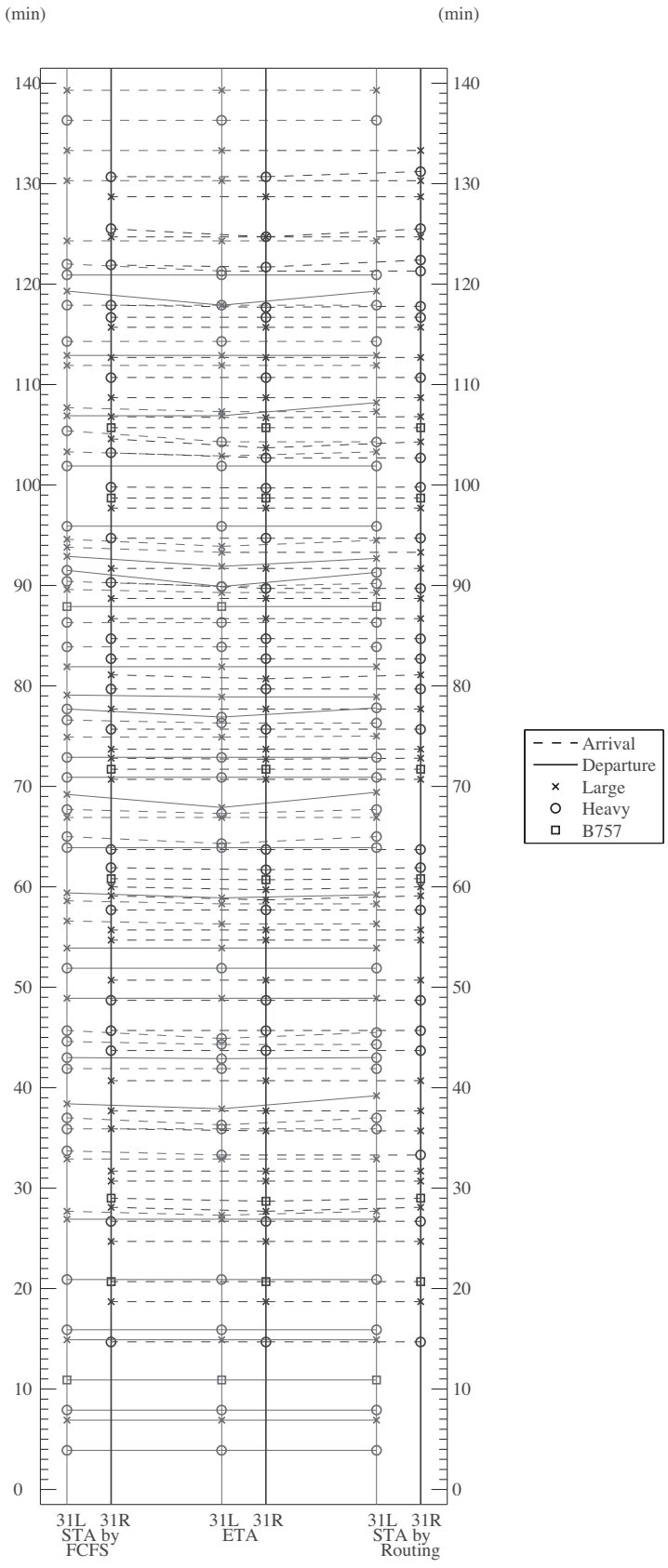
In this scenario, it is assumed that the traffic loads on Runway 31L and 31R are fairly balanced. The dynamic strategy mainly focuses on sequencing and separation adjustments.

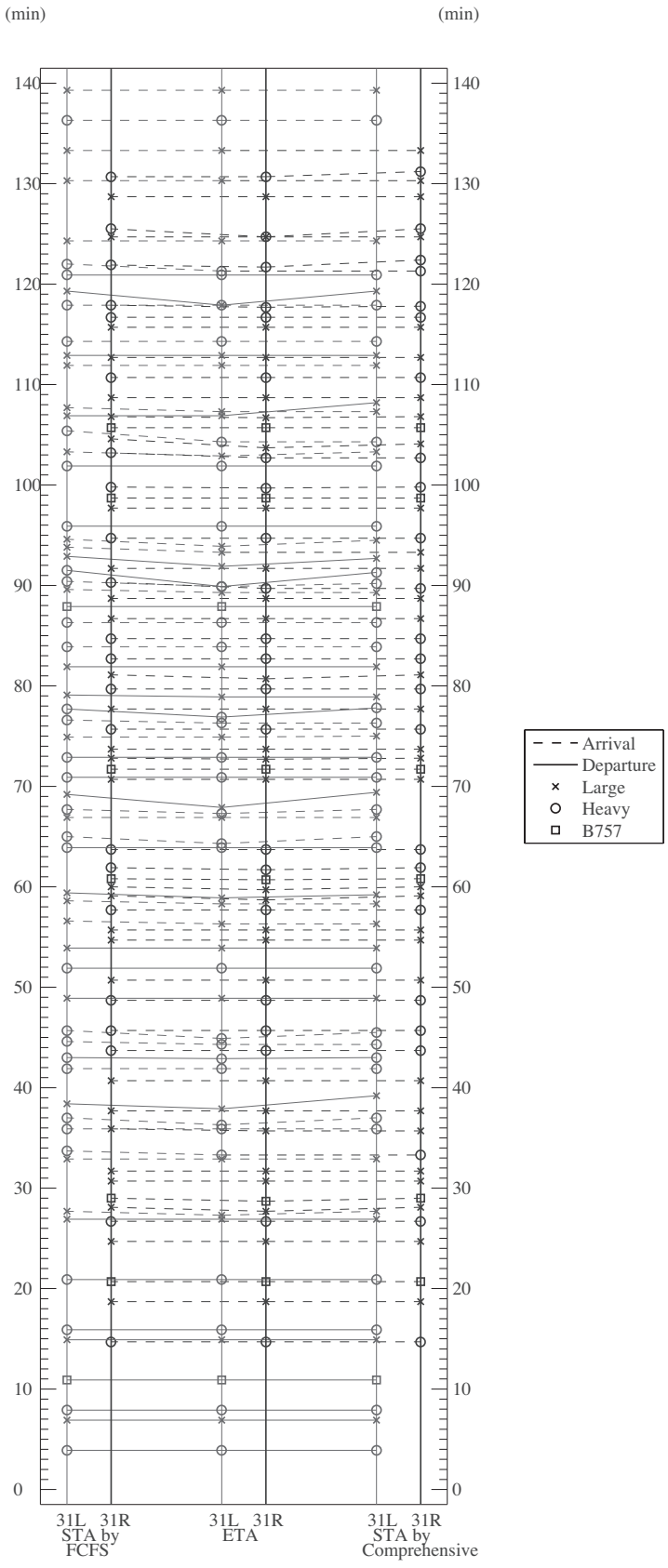
Again, the Sequencing Option only permits sequence adjustments. No traffic is directed from Runway 31L to 31R. As shown in Fig. 4.13 and Fig. 4.17, neither the FCFS Option nor the Sequencing Option has altered the original runway assignments. In this case, only a minimum amount of arrival-departure swaps are produced and the original sequences are basically maintained on both runways. A few departures are held on the ground so that the runway can serve arrivals first. These results suggest that although the initial traffic demand is not challenging, the dynamic strategy can still refine the original sequence to produce less delays. Due to the initially balanced traffic loads on Runway 31L and 31R, the overall delay in Fig. 4.16 is quite small compared to the previous scenario with initially imbalanced traffic. Even under such a circumstance, the Sequencing Option still outperforms the FCFS Option.

Similar as in the initially imbalanced traffic scenario, the Routing Option further reduces the overall delay somewhat, as shown in Figs. 4.14 and 4.16. Both the FCFS Option and the Routing Option maintain most flight sequences. Meanwhile, a total of 6 runway assignment changes are made, which is smaller compared to the first scenario with imbalanced initial traffic between the two runways.

Finally, the Comprehensive Scheduling Option still offers the largest freedom and achieves the minimum overall delay in this scenario. It seems that the Routing Option and the Comprehensive Scheduling Option eventually assign the same runway traffic loads, as shown in Fig. 4.15 and Fig. 4.17. However, the Comprehensive Scheduling Option produces a better sequencing plan with a smaller overall delay (Fig. 4.16). In


(min)
(min)
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Seq.


(min)
(min)
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Routing


(min)
(min)
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
Arrival
Departure
Large
Heavy
B757
31L 31R
STA by
FCFS
31L 31R
ETA
31L 31R
STA by
Comprehensive

all options, the overall delays produced in this scenario are much smaller than those in the first scenario.

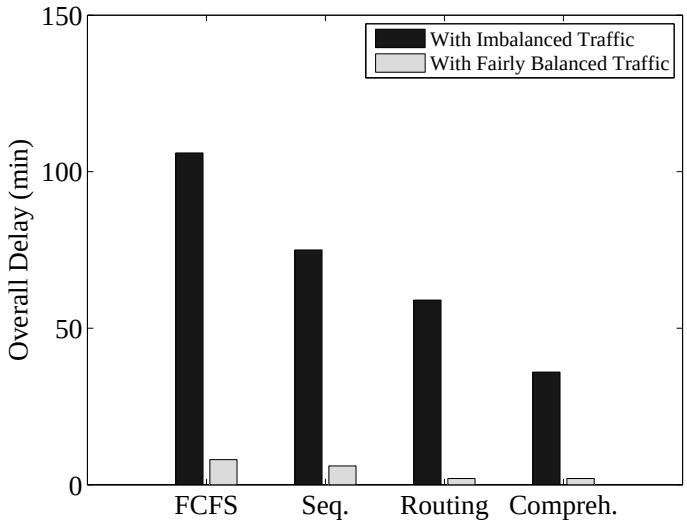


Figure 4.16: Overall delays with various scheduling options.

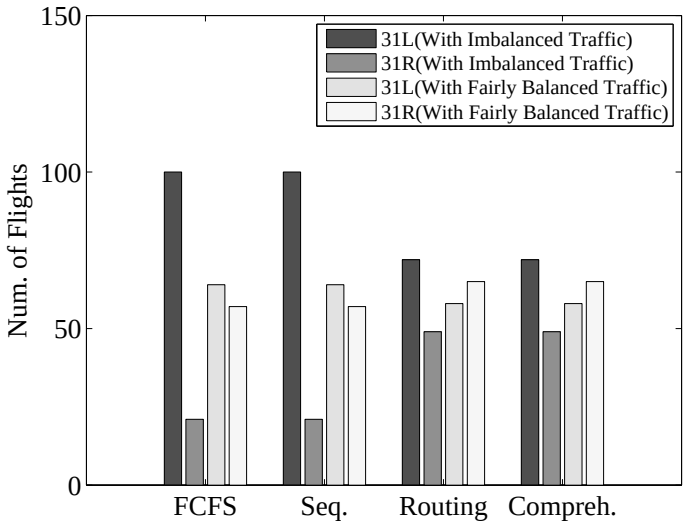


Figure 4.17: Optimized runway traffic loads with various scheduling options.

Chapter 5

Stochastic Runway Scheduling

5.1 The Need for Stochastic Scheduling

Deterministic runway planning assumes that the runway arrival and departure times are explicitly known at the time of planning. However, airport operations often experience various sources of uncertainties. For example, passenger-boarding delay, baggage loading delay, and bad weather condition could alter a flight's arrival or departure time. This impact is eventually propagated to air traffic controller's planning for airport-wide traffic.

At the time of planning, air traffic controllers often have very limited flight information and have to make decisions quickly, even if some information is not known with certain. During the peak hours, controllers do not even have sufficient time to think about the overall traffic condition and have to decide flight sequences, departure/arrival times based on a first-come, first-served (FCFS) basis. Once the traffic demand exceeds certain limits, the FCFS principle could drastically decrease the overall throughput of the airport and result in significant delays on many flights.

In general, runway operations are expected to be efficient, quickly solvable, and robust to various sources of uncertainties. If the original flight schedules are not exactly

known, we either want to update flight schedules frequently with the most up-to-date information, or mathematically capture the random factors of flight schedules and natively optimize results against uncertainties. To achieve these goals, we could develop two solution approaches. The first approach attempts to update flight schedules frequently to capture the latest changes in flight schedules. This approach is regarded as the structural solution, and is compatible with most existing runway scheduling methods. It employs a structure that periodically accepts newest traffic data and schedules a portion of traffic demand. Using a sliding window strategy, this solution approach processes the incoming traffic segment by segment. With frequent updates on traffic condition, it suffers less impact from flight schedule uncertainties that can rise drastically as planning horizon propagates. This approach can use most existing deterministic runway scheduling tools while uncertainties are considered.

The second approach mathematically accounts the stochastic characteristics of various sources of uncertainties in modeling, and adopts a two-stage stochastic program to produce optimal flight schedules. It seeks to reformulate the runway scheduling problem using stochastic optimization methods. It regards the input data as random variables and takes into account their stochastic characteristics in modeling and computation.

5.2 Structural Solution Using Sequential Dynamic Scheduling

The structural solution is derived from the sequential dynamic strategies[35] that were originally invented to handle heavy traffic demand over a long planning window. The structural solution is designed to frequently update flight schedules using newest traffic data available to counteract the uncertainties in flight operations. The compatible sliding window strategy is also developed to guide the structural solution in updating flight schedules. For most of the existing deterministic runway scheduling tools, only minor modifications are required if using an appropriate sliding window strategy. In a

stochastic scheduling scenario, traffic data delivered to each scheduling window is no longer assumed known with certain. Instead, schedules computed in previous windows might become ineffective or even not meaningful. The sliding window strategy helps the structural solution supply new schedules in a timely manner. In order to acquire meaningful schedules in a reasonable amount of time, scheduling window size and computational algorithm must be chosen carefully.

5.2.1 Sliding Window Strategy

In practical runway scheduling, a planning horizon defines the time interval in which flight schedules need to be computed. A planning horizon often begins at a time in near future and extends all the way into the future. The sliding window strategy divides the planning horizon into a series of small time intervals, called scheduling windows. The strategy then uses an appropriate tool to compute newest schedules for flights captured in the current scheduling window. Traffic volume as well as computational load is determined by the size of the window, which could eventually affect the performance of a particular strategy. A timely solution desires a bound on the maximum computational time over each scheduling window. As a result, window size can be adjusted according to the overall traffic demand for timely solutions. Table. 5.1 summarizes all components that can affect the optimality and computational speed of a sliding window strategy.

Table 5.1: Dynamic strategy parameters

Parameters	**Meaning**
T_W	Scheduling window size
T_{adv}	Advance time
T_C	Computational time

5.2.2 A Brief Review in Time Relations

There are two time concepts in dynamic scheduling: actual time and planning time. Actual time is the current clock time in the real world, whereas a planning time is a coordinate on the planning horizon that is ahead of the actual time. These time concepts are illustrated in Fig. 5.1, in which the vertical axis and horizontal axis represent the actual and the planning time respectively.

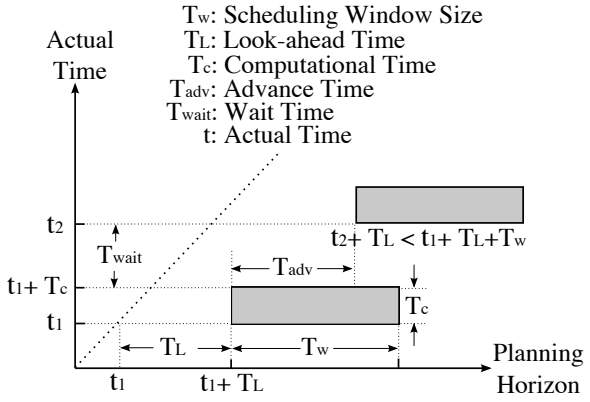


Figure 5.1: The timeline with overlapping scheduling windows.

At an actual time t_1, a dynamic strategy begins planning for future traffic that shall arrive at $t_1 + T_L$, where T_L is called the look-ahead time. The dynamic strategy first considers traffic over the scheduling window of $[t_1 + T_L, t_1 + T_L + T_W]$, where T_W is the size of the scheduling window. In addition, T_C represents the computational time in obtaining solutions over this window. After scheduling solutions for this window are obtained, the dynamic algorithm would wait until actual time t_2. It will then begin scheduling traffic that plans to use the airport over the window of $[t_2+T_L, t_2+T_L+T_W]$. This process is then repeated.

5.2.3 A Brief Review in Designing Scheduling Windows

There are two general approaches to dividing the planning horizon into smaller scheduling windows: non-overlapping windows and overlapping windows. In either case, the advance time between the computations of neighboring scheduling windows is defined as

$$T_{\text{adv}} = t_{n+1} - t_n \tag{5.1}$$

where t_n is the beginning time of the nth scheduling window and $n = 1, 2, \cdots$. If the scheduling windows do not overlap, we have

$$T_{\text{adv}} = T_W \tag{5.2}$$

Otherwise,

$$T_{\text{adv}} < T_W \tag{5.3}$$

When computations for the scheduling window beginning at t_n are completed, computations for the subsequent scheduling window won't start until t_{n+1}. Assuming the total computational time for a window is T_C, the waiting time is given by

$$T_{\text{wait}} = T_{\text{adv}} - T_C \tag{5.4}$$

During the waiting time, the most recent traffic data may be acquired for subsequent computations. In addition, T_{wait} also serves as a safety buffer between two computations in case the preceding computation exceeds T_W. In any circumstance, we have

$$T_{\text{adv}} \geq T_C \quad \text{or} \quad T_{\text{wait}} \geq 0 \tag{5.5}$$

5.3 Algorithmic Solution Using Stochastic Programming

Unlike structural solution, the algorithmic solution seeks to reformulate the runway scheduling problem using a two-stage stochastic program. Meanwhile, input data such as the flight schedule is no longer regarded stochastic. The randomness of the data is

considered in the stochastic program. This approach is inherently a stochastic programming method and is distinct from any other deterministic approaches. Here, a two-stage stochastic program is proposed to compute optimal flight sequence and schedule against uncertainties in original flight schedule.

The first-stage objective function:

$$\underset{T,X,Y,Z}{\text{Min}} \sum_{i=1}^{N} C_i t_i^{\text{final}} + E\{Q(T,Y,\omega)\} \tag{5.6}$$

Subject to

$$\sum_{r=1}^{R} x_i^r = 1 \tag{5.7}$$

$$z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq 0 \tag{5.8}$$

$$z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) - \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \leq 1 \tag{5.9}$$

$$(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(3 - z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{5.10}$$

$$-(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 + z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{5.11}$$

$$t_{i(p+1)} - t_{ip} + \mathcal{M}(1 - x_i^r) \geq \tau_{\min}(c_i, p) \tag{5.12}$$

$$t_{i(p+1)} - t_{ip} - \mathcal{M}(1 - x_i^r) \leq \tau_{\max}(c_i, p) \tag{5.13}$$

The second-stage objective function:

$$Q(T,Y,\omega) = \underset{T,Y}{\text{Min}} \sum_{i=1}^{N} \sum_{p=1}^{P} C_i q_i^{\omega} y_{ip}^{\omega} \tag{5.14}$$

Subject to

$$t_{ip} + y_{ip}^{\omega} \geq \hat{t}_{ip}^{\omega} \tag{5.15}$$

where y_{ip}^{ω} is the resource decision for flight i's arrival time at waypoint p in scenario ω. For the convenience of numerical computation, it is assumed that $\omega \in \Omega = \{\omega_1, \omega_2, ..., \omega_N\}$, and N is the total number of possible scenarios.

In the two-stage stochastic program, the first stages attempts to minimize the mean of arrival/departure times at the last waypoint and the mean of the penalty function due to the uncertainties in predicted flight schedules. The second stage, however, minimizes the mean of penalties in each possible scenario. Although this two-stage stochastic is well constructed and can achieve our goals, it is not computational convenient. A common way to resolve this issue is to introduce its deterministic equivalent, which is given below.

$$\underset{T,X,Z,Y}{\text{Min}} \sum_{i=1}^{N} C_i t_i^{\text{final}} + \sum_{\omega=1}^{\Omega} \sum_{i=1}^{N} \sum_{p=1}^{P} C_i p^{\omega} q_i^{\omega} y_{ip}^{\omega} \tag{5.16}$$

Subject to

$$\sum_{r=1}^{R} x_i^r = 1 \tag{5.17}$$

$$z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq 0 \tag{5.18}$$

$$z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) - \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \leq 1 \tag{5.19}$$

$$(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(3 - z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{5.20}$$

$$-(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 + z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2) \tag{5.21}$$

$$t_{i(p+1)} - t_{ip} + \mathcal{M}(1 - x_i^r) \geq \tau_{\min}(c_i, p) \tag{5.22}$$

$$t_{i(p+1)} - t_{ip} - \mathcal{M}(1 - x_i^r) \leq \tau_{\max}(c_i, p) \tag{5.23}$$

$$t_{ip} + y_{ip}^{\omega} \geq \hat{t}_{ip}^{\omega} \tag{5.24}$$

The algorithmic solution mathematically formulates the uncertainties in modeling and computation. Its results are natively optimized against the uncertainties in flight schedules. However, compared to the deterministic mixed-integer program, this two-stage stochastic program contains extra decision variables and constraints. Thus, it requires more computational effort than the original deterministic integer program demands.

5.4 Random Errors and Stochastic Characteristics

The computation of optimal flight sequence relies on the accuracy of the predicted arrival/departure time. However, uncertainties in flight operations make it difficult to estimate an accurate arrival/departure schedule at the time of planning. For example, passenger boarding process, baggage transportation, and weather condition can often delay a flight's pushback. On the other hand, it is not rare to observe delay on arriving traffic due to congested runway or unexpected bad weather.

To simplify the calculation, two types of errors are considered in flight schedule: pushback time prediction error and arrival time prediction error. It is also noticed that the predicted arrival time can be more uncertain due to the traffic and weather condition both inside and outside the airport. Therefore, we further assume that the arrival time prediction error to be time correlated. In other words, flights with longer remaining flight time (further away from the airport) have more uncertainties in their arrival schedules than those closer to the airport.

After studying the previous work in pushback time prediction[36, 31], we use a shifted Chi-Square distribution to characterize the pushback time error. To simplify the computation, the probability distribution has a zero minute mean and a standard deviation of 5.65 minutes. The PDF begins from -15 minutes and extend to positive infinite. The probability density function is shown in Fig. 5.2.

For simplification, we assume that the arrival time prediction error has zero mean and a standard deviation dependent on the remain flight time. In general, a longer remaining flight time leads to greater uncertainty. The distribution is described below. A visualized description is provided in Fig. 5.3

$$\mathcal{N}(0, (\frac{1}{15}t_{\text{remain}} + 2)^2), \text{ when } 0 < t_{\text{remain}} \leq 60 \tag{5.25}$$

$$\mathcal{N}(0, 6^2), \text{ when } 60 < t_{\text{remain}} \tag{5.26}$$

Since the algorithmic solution schedules hours-long traffic in one computation, the

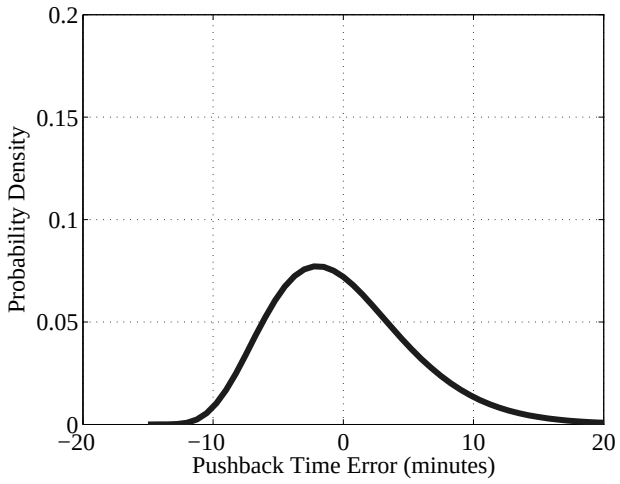


Figure 5.2: Pushback time prediction error, Chi-Square distribution.

arrival time prediction error can grow drastically. The structural solution, on the contrary, processes traffic using a series of small windows, which yields to smaller arrival time perdition error. In each window, newest traffic data can be updated and the remaining flight time is reset. Therefore, the remaining flight time in each window always starts from 0 with minimum errors in predicting arrival time.

Both solutions have their pros and cons. In general, algorithmic solution may encounter greater uncertainties in predicted arrival times than structural solution does. Therefore, the structural solution may produce better flight sequence because of the better understanding of the arrival flight schedule. On the other hand, the structural solution is inherently a piecewise optimization method. Under the same traffic condition, it might be outperformed by other global optimization methods, such as algorithmic solution or even FCFS global optimization. The trade-off of accurate flight schedule versus global optimization is studied in the remainder part.

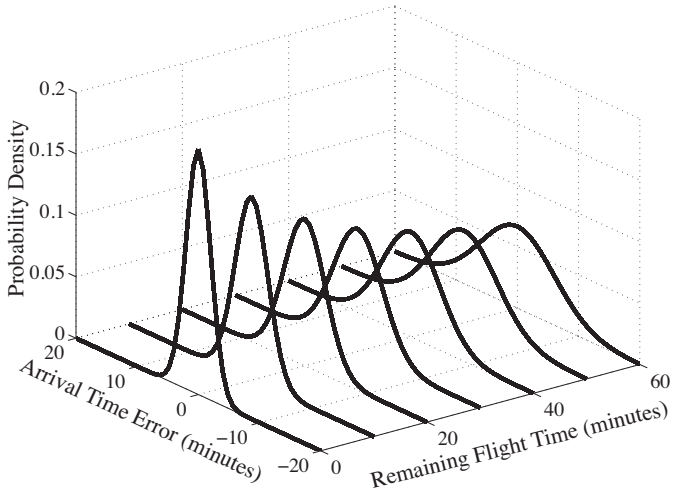


Figure 5.3: Arrival time prediction error vs. remaining flight time to terminal area.

5.5 Simulation Setup

5.5.1 Test Schemes

To evaluate the algorithmic solution and structure solution under uncertainties, two simulation schemes are carefully designed. A random error generator is also devised to generate stochastic flight schedules.

Fig. 5.4 explains the algorithmic solutions test scheme. Firstly, the two-stage stochastic program reads all possible realizations of flight schedule and computes the optimal flight sequence under uncertainty. This sequence records the relative position of each flight in the entire queue. Secondly, this "optimal" sequence is passed onto the Solution Examiner, which test such sequence against flight schedules with randomly generated errors provided by Random Error Generator. The second step is repeated until all test samples are evaluated.

In the above process, adding random errors to the original predicted schedule generates perturbed schedule. For example, the perturbed ETA of flight i at waypoint p is

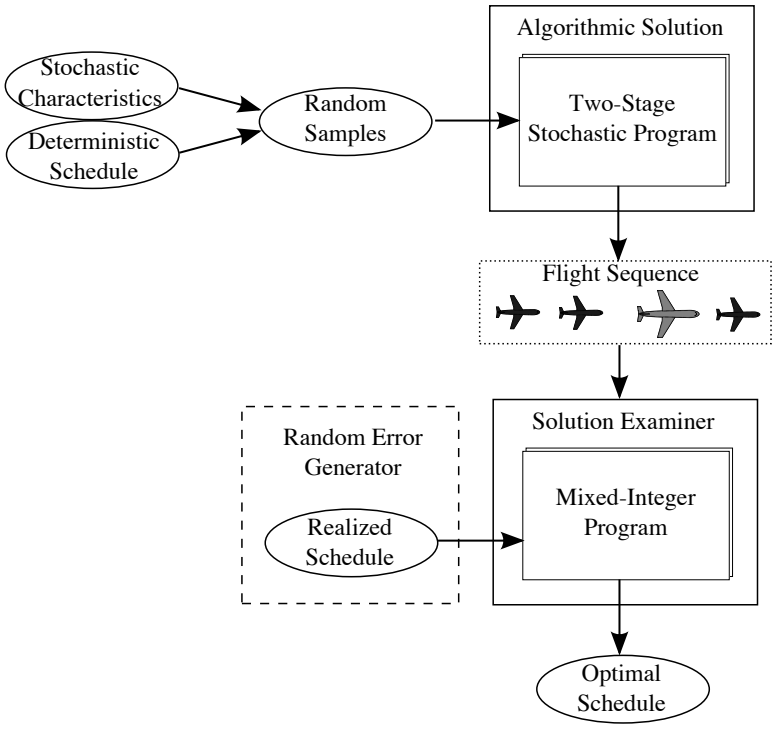


Figure 5.4: Algorithmic solution test scheme.

expressed as $(\hat{t}_{ip} + \delta\hat{t}_{ip})$. The stochastic characteristics of random error $\delta\hat{t}_{ip}$ have been introduced in Figs. 5.2 and 5.3.

As briefly mentioned before, this two-stage stochastic program can bring in an excessive number of constraints considering many possible realization of $(\hat{t}_{ip} + \delta\hat{t}_{ip})$ for each flight. For example, even for single-point scheduling, n flights would generate 10^n scenarios if $\delta\hat{t}_{ip}$ can take up 10 different values. Limited by the computational power, the two-stage stochastic program cannot process a huge number of scenarios in a timely manner. Thus, the ETA prediction error $\delta\hat{t}_{ip}$ only takes from three levels in the tests: $-\sigma$, 0, and σ, with equal probability of $\frac{1}{3}$. To further reduce the number of scenarios in the multiple-point scheduling scheme, each flight's ETAs at different waypoints always adopt the same level of uncertainty. For example, if $\delta\hat{t}_{ip} = \sigma_{ip}$, then $\delta\hat{t}_{i(p+1)} = \sigma_{i(p+1)}$,

etc. A total of 100 scenarios are generated and tested.

Fig. 5.5 explains the test scheme for the structural solution. Here, perturbed schedules are generated window by window. The Sliding Window Traffic Coordinator identifies the traffic demand in current window, and calls the Random Error Generator to add random errors to flight schedule in current window. This perturbed flight schedule is sent to the Mixed-Integer Program and an optimal schedule is then computed. In the last, the Sliding Window Traffic Coordinator records the optimal schedule computed in current window before rolling to next window. This process is repeated until all flights are processed. The entire process is repeated until all test samples are evaluated.

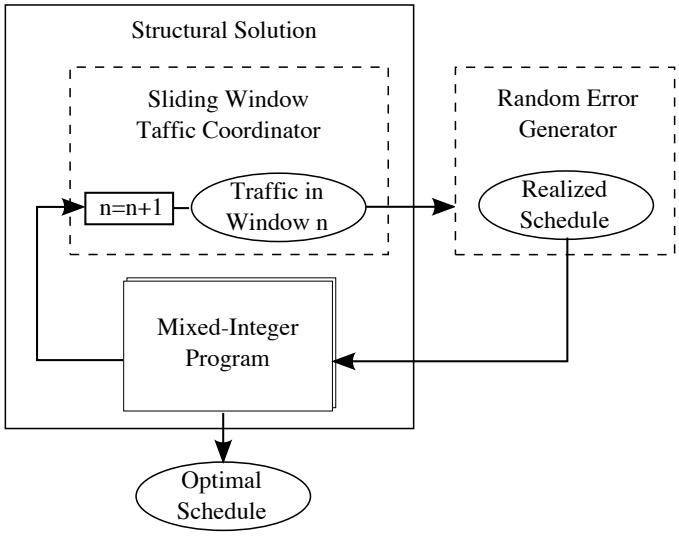


Figure 5.5: Structural solution test scheme.

5.5.2 Airport Model and Traffic Demands

Arrival rushes with 100-180 flights within 2 hours are simulated using a simplified JFK international airport model shown in Figs. 2.2 and 2.3. Arrival and departure rates are listed in Table 5.2. It is assumed that the arriving traffic streams enter the airport

through three fixes: LENDY, CAMRN and ROBER. Flights from ROBER land on Runway 31R by default, whereas flights from LENDY and CAMRN use Runway 31L. All departures take off from Runway 31L and exit the terminal airspace through NORTH GATE, EAST GATE, WEST GATE and OCEAN GATE. The ratio between arrival and departure traffic is maintained at around 7:3 in each test.

Three aircraft weight classes are considered: Large, Heavy and Boeing-757. Two traffic mixes are tested across all demands. As in Table 5.3, mix 1 has a majority of Large flights and some Heavy flights and Boeing-757. Mix 2 however has comparable number of Large flights and Heavy flights. The airborne and surface separation requirements are listed in Tables 5.4 and 5.5, respectively. All separation requirements are dependent upon aircraft pair weight classes and are expressed in time.

Table 5.2: Traffic demand rates (flights/hour/runway)

Number of flights	Arrival rate	Departure rate
100	17.5	7.5
120	21.0	9.0
140	24.5	10.5
160	28.0	12.0
180	31.5	13.5

Table 5.3: Traffic mixes

Type	Mix 1	Mix 2
Large	80%	50%
757	10%	10%
Heavy	10%	40%

Table 5.4: Airborne and runway separations (seconds)

Leader \ Follower	Large	B757	Heavy
Large	50	50	50
757	85	65	65
Heavy	85	85	65

Table 5.5: Surface separations (seconds)

Leader \ Follower	Large	B757	Heavy
Large	30	30	30
757	30	30	30
Heavy	30	30	30

5.6 Results

Both algorithmic solution and structural solution are examined against the FCFS approach. Figs. 5.6 and 5.7 suggest that algorithmic solution and structure solution are both able to reduce overall delay and runway delay. As traffic demand increases, all three approaches introduce more delays in the results. Due to its sliding window strategy, the structure solution is more robust to growing uncertainties than the algorithmic solution behaves until the traffic volume become huge. By that time, the piecewise optimization nature of the structural solution might kick in and can reduce its performance. The algorithmic solution now becomes the best choice among the three.

In most times, the structural solution is more efficient in runway operations than the algorithmic solution, especially when the traffic demand is moderate. However, once traffic demand exceeds certain point, the structural solution may suffer decreased performance due to piecewise optimization nature. This change in performance makes it less efficient than the algorithmic solution, which becomes the best choice. Furthermore, FCFS has an acceptable performance under light traffic. As traffic demand increase and

uncertainties introduced, FCFS is no longer as efficient as other solutions.

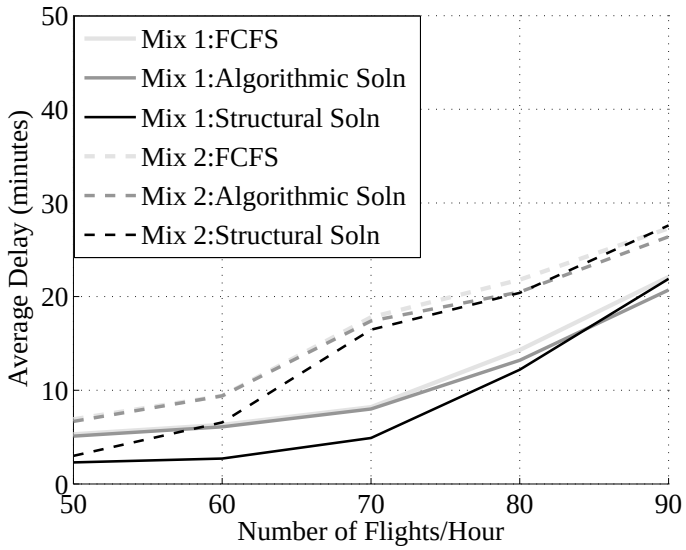


Figure 5.6: Average delay (minutes).

5.7 Conclusions

A systematic study is conducted to understand efficient runway scheduling in the presence of uncertainties. This study first develops a structural runway scheduling solution that can utilize existing runway scheduling/sequencing tools under uncertain traffic demand. In the structural solution, a sliding window strategy is adopted, which processes the incoming traffic section by section sequentially. It can take advantage of the newest traffic data in each computation and suffers less impact from flight schedule uncertainties. Then, the algorithmic solution is developed mainly based on a stochastic program. It mathematically describes flight schedule uncertainties in its model and use the two-stage stochastic program to compute optimal results. The algorithmic solution needs one computation and is inherently a global optimization method, but suffers the fast

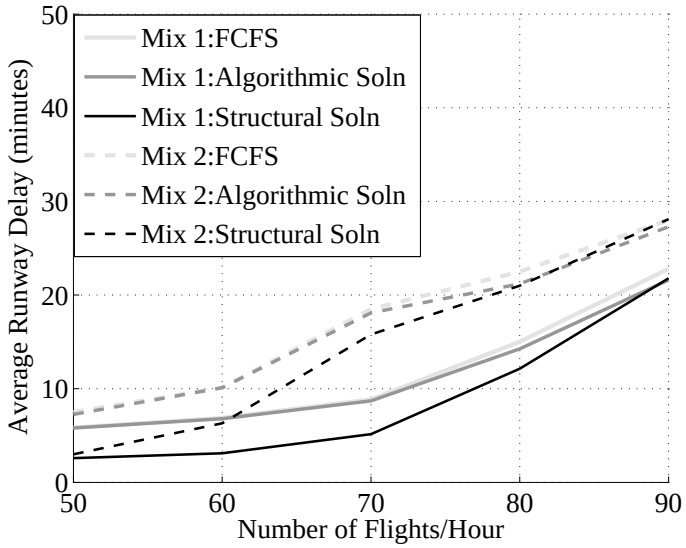


Figure 5.7: Average runway delay (minutes).

growing uncertainties as prediction time increases. On the other hand, the structural solution experiences less uncertainties in flight schedules, but it may suffer performance decrease under dense traffic due to its piecewise optimization nature.

Simulation results suggest that under sparse traffic, the structural solution has impressive performance compared to the algorithmic solution and FCFS. As the traffic demand further increases, the structural solution experiences much steep decrease in performance and might be outperformed by algorithmic solution under some circumstances. Both solution approaches devised in this work outperform the FCFS method in most traffic conditions.

In the future, the possibility of combing the algorithmic solution and the structural solution will be studied. The combined solution will maintain a dynamic sliding window structure while adopting a stochastic program mathematical model. It is our curiosity to study its performance over the existing solutions.

Chapter 6

Conclusion and Recommended Future Work

This thesis conducts a systematic study of terminal area traffic management using optimization methods. The importance of airport in national airspace system is addressed. The challenges in todays terminal traffic management are explained. The necessity and importance of the research is addressed. To help human air traffic controllers manage complex terminal traffic under challenging demands, our strategy is to calculate each flight's optimal arrival/departure schedule, to minimize the overall flight delay and runway congestion in the entire airport.

In this thesis, three solutions are presented, including the static solution, the dynamic solution, and the stochastic solution. The static solution uses one computation and attempts to optimize the schedules of many flights arriving/departing the airport within a wide time window. It employs an integrated runway scheduling framework using multiple-point scheme. The dynamic solution attempts to divide the entire traffic flow into a series of small pieces, and optimize flight schedules piece by piece. It collects the latest traffic information before each computation and experiences far less computational load than the static solution. The stochastic solution assumes that traffic

information is not known with certain due to a variety of random factors in actual flight operations. It is mathematically and structurally designed to handle multiple sources of uncertainties in managing terminal traffic.

The static solution adopts the integrated runway scheduling framework that solves both aircraft scheduling and routing problems. Unlike most of the existing work, it considers both arriving and departing traffic in the entire terminal airspace. It models a single airport in this work, but allows multiple runways to be jointly operated and can be extended to multiple airports. An airport is represented by a number of arriving/departing routes with a series of check points attached. These check points are called the scheduling points. Each aircrafts arrival times at these scheduling points are adjusted such that the entire airport is operated more efficiently. The mixed-integer linear programming (MILP) is adopted to compute these arrival times, as well as route (runway) assignments and flight sequences during arrival and departure. Aircraft performance limits, operational constraints, and separation requirements are imposed. The overall delay is minimized.

Three algorithms are developed in the static solution. The open-route MILP algorithm produces optimal route selections, sequences and schedules. The fix-route MILP algorithm fixes the route assignments but selects optimal sequences and schedules. The First-Come-First-Served (FCFS) algorithm provides baseline solutions. Numerical results suggest a clear tradeoff between scheduling efficiencies and computational times offered by different solution algorithms. In general, the open-route MILP algorithm has the smallest overall flight delay, but is the slowest among the three. The FCFS algorithm is the fasted, but gives the worst delay. The fix-route MILP algorithm sits in between. Under light traffic, the FCFS algorithm is on par with the MILP based algorithms. But when traffic becomes excessive, its performance decreases drastically.

Although the static solution is comprehensive and can schedule short-term traffic, it

still cannot be directly used in real-world operations. These limitations include: computational power limits the size of the scheduling window; even with a huge scheduling window, fast and reliable solution is not guaranteed; growing uncertainties in trajectory prediction can make the computed optimal schedules inapplicable. Therefore, a dynamic solution is developed to resolve the above-mentioned issues.

The dynamic solution employs the dynamic strategy that divides the planning horizon into a series of smaller scheduling windows, and use an appropriate static scheduling algorithm to obtain optimal schedule for each window. Dynamic strategys structure and parameters are also explained. These parameters include scheduling window size, advance time between two successive windows, and the static computational algorithm. By choosing appropriate parameters, dynamic strategies can adjust its performance in between of computational speed and optimality.

With the same parameters, simulation results suggest that overlapping-window dynamic strategy is faster due to lighter computational load. For the overall flight delay, more freedoms in scheduling give less delay, but need longer computational times. Specifically, dynamic strategies adjusting both sequences and runway assignments offer minimum flight delays. When only one choice becomes available, changing runway assignments is more efficient than adjusting sequences, especially under imbalanced traffic across the runways. Even when flight sequences and runway assignments are fixed, arrival times at different scheduling points can still be optimized.

Both static and dynamic solutions assume the predicted traffic situation to be always accurate. They are inherently deterministic solutions. In real world, uncertainties always exist in flight operations. They might contaminate the well-computed schedule and make it inefficient or even inapplicable. Many sources are accounted for the contributions to uncertainties, such as the passenger boarding delay, baggage transportation and loading delay, and adverse weather condition. Deterministic runway scheduling does not consider these random events and cannot optimize runway operations under

uncertainties.

To handle the uncertainties, the stochastic runway scheduling solution is then developed. It assumes that traffic information is not known with certain due to a variety of random factors in actual flight operations. This stochastic solution has two options, the structural solution and the algorithmic solution. They are structurally and mathematically designed to handle multiple sources of uncertainties in managing terminal traffic. The first solution employs a structure that periodically accepts newest traffic data and schedules a portion of traffic demand at a time. With a sliding window strategy, this solution processes the incoming traffic segment by segment. With frequent update on traffic situation, it suffers less impact from flight schedule uncertainties that can rises drastically as prediction time increases. This approach can use most existing deterministic runway scheduling tools even if uncertainties are incorporated. The second solution uses stochastic optimization algorithm. It mathematically models the uncertainties and attempts to produce optimal flight schedules weighted against all possible traffic situations. This solution adopts stochastic optimization algorithms making it fundamentally different from the other solutions.

Simulation results suggest that under sparse traffic, the structural solution has impressive performance compared to the algorithmic solution and FCFS. As the traffic demand further increases, the structural solution experiences more rapid decrease in performance and can be outperformed by the algorithmic solution under very heavy traffic. Both solution approaches outperform the FCFS method, especially under heavy traffic.

We suggest that the future study should examine the possibility of combining the algorithmic solution and the structural solution. The combined solution maintains a dynamic sliding window structure while adopting a stochastic optimization algorithm to compute results. We are also interested in stochastic optimization algorithms other than the two-stage method used in this work. Due to the excessive volume of traffic in

runway scheduling, parallel computation in computer programming is quite necessary. Therefore, we also suggest a future research in improving computational speed. Multiple airport scheduling can also be investigated by extending the integrated runway scheduling framework. A multiple airport scenario is essentially equivalent to a huge "airport" with a number of runways that are apart from each other, which can be handled by the integrated scheduling framework offered in this thesis. Runway configuration selection can also be studied.

References

[1] JPDO, "Next Generation Air Transportation System (NextGen)," `http://www.jpdo.gov/library/20110113_FP_Report_Final_v3.pdf`.

[2] Dear, R. G. and Sherif, Y. S., "An algorithm for computer assisted sequencing and scheduling of terminal area operations," *Transportation Research Part A: General*, Vol. 25, No. 2-3, March-May 1991, pp. 129–139.

[3] Ernst, A. T., Krishnamoorthy, M., and Storer, R. H., "Heuristic and exact algorithms for scheduling aircraft landings," *Networks*, Vol. 34, No. 3, 1999, pp. 229–241.

[4] Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D., "Scheduling aircraft landings–the static case," *Transportation Science*, Vol. 34, No. 2, May 2000, pp. 180–197.

[5] Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D., "Displacement Problem and Dynamically Scheduling Aircraft Landings," *The Journal of the Operational Research Society*, Vol. 55, No. 1, Janurary 2004, pp. 54–64.

[6] Kupfer, M., "Scheduling Aircraft Landings to Closely Spaced Parallel Runways," Eighth USA/Europe Air Traffic Management Research and Development Seminar, Napa, CA, 2009.

[7] Chandran, B. and Balakrishnan, H., "A Dynamic Programming Algorithm for Robust Runway Scheduling," *Proceedings of American Control Conference*, IEEE, Piscataway, NJ, 2007, pp. 1161–1166.

[8] Lee, H. and Balakrishnan, H., "Fuel cost, delay and throughput tradeoffs in runway scheduling," *Proceedings of the American Control Conference*, IEEE, Piscataway, NJ, June 2008, pp. 2449 –2454.

[9] Rathinam, S., Wood, Z., Sridhar, B., and Jung, Y., "A Generalized Dynamic Programming Approach for a Departure Scheduling Problem," AIAA Guidance, Navigation, and Control Conference, Chicago, IL, August 2009.

[10] Atkin, J. A. D., Burke, E. K., Greenwood, J., and Reeson, D., "The Effect of the Planning Horizon and the Freezing Time on Take-off Sequencing," *Proceedings of 2nd International Conference on Research in Air Transportation (ICRAT2006)*, June 2006.

[11] Gupta, G., Malik, W., and Jung, Y., "A Mixed-Integer Linear Program for the Airport Departure Scheduling Problem," AIAA Aviation Technology, Integration, and Operations Conference (ATIO), Hilton Head, SC, September 2009.

[12] Smeltink, J., Soomer, M., DeWaal, P., and Van Der Mei, R., "An Optimisation Model for Airport Taxi Scheduling," Thirtieth Conference on the Mathematics of Operations Research, Lunteren, The Netherlands, January 2005.

[13] Marin, A., "Airport Management: Taxi Planning," *Annals of Operations Research*, Vol. 143, 2006, pp. 191–202.

[14] Roling, P. and Visser, H., "Optimal Airport Surface Traffic Planning Using Mixed-Integer Linear Programming," *International Journal of Aerospace Engineering*, Vol. 2008, No. 732828, 2008.

[15] Rathinam, S., Montoya, J., and Jung, Y., "An Optimization Model for Reducing Aircraft Taxi Times at the Dallas Fort Worth International Airport," 26th International Congress of the Aeronautical Sciences, 2008.

[16] Keith, G., Richards, A., and Sharma, S., "Optimization of Taxiway Routing and Runway Scheduling," AIAA Guidance, Navigation and Control Conference, Honolulu, HI, August 2008.

[17] NASA, "Center TRACON Automation System," `http://www.aviationsystemsdivision.arc.nasa.gov/research/foundations/index.shtml`.

[18] Garcia, J., "MAESTRO–A Metering and Spacing Tool," *Proceedings of the 1990 American Control Conference*, IEEE, Piscataway, NJ, 1990, pp. 501–507.

[19] Volckers, U., "Arrival Planning and Sequencing with COMPAS-OP at the Frankfurt ATC-Center," *Proceedings of the 1990 American Control Conference*, IEEE, Piscataway, NJ, 1990, pp. 496–501.

[20] Erzberger, H., Davis, T. J., and Green, S. M., "Design of Center-TRACON Automation System," AGARD Guidance and Control Symposium on Machine Intelligence in Air Traffic Management, Berlin, Germany, May 1993.

[21] Davis, T. J., Erzberger, H., Green, S. M., and Nedell, W., "Design and Evaluation of an Air Traffic Control Final Approach Spacing Tool," *Journal of Guidance, Control and Dynamics*, Vol. 14, No. 4, July-August 1991, pp. 848–854.

[22] Erzberger, H., "Design Principles and Algorithms for Automated Air Traffic Management," AGARD Lecture Series, Rhode- Saint-Genèse, Belgium, 1995.

[23] Robinson III, J. E., Davis, T. J., and Isaacson, D. R., "Fuzzy Reasoning-Based Sequencing of Arrival Aircraft in the Terminal Area," AIAA Guidance, Navigation, and Control Conference, New Orleans, LA, August 1997.

[24] Wong, G. L., "The Dynamic Planer: the Sequencer, Scheduler, and Runway Allocator for Air Traffic Control Automation," *NASA/TM-2000-209586*, NASA Ames Research Center, Moffet Field, CA, 2000.

[25] Cheng, Y., "A Knowledge-Based Airport Gate Assignment System Integrated with Mathematical Programming," *Computers & Industrial Engineering*, Vol. 32, No. 4, 1997, pp. 837–852.

[26] Bolat, A., "Assigning Arriving Flights at An Airport to the Available Gates," *The Journal of the Operational Research Society*, Vol. 50, No. 1, 1999, pp. 23–34.

[27] Kim, S. H., Feron, E., and Clarke, J., "Assigning Gates by Resolving Physical Conflicts," AIAA Guidance, Navigation, and Control Conference, Chicago, IL, 2009.

[28] Capozzi, B., Atkins, S., and Choi, S., "Towards Optimal Routing and Scheduling of Metroplex Operations," AIAA Aviation Technology, Integration, and Operations Conference, Hilton Head, SC, 2009.

[29] Bianco, L., Dell'Olmo, P., and Giordani, S., "Dynamic Algorithms for TMA Traffic Management," *Proceedings of the 8th IFAC/IFIP/IFORS Symposium on Transportation Systems*, Pergamon–Elsevier Science, New York, NY, 1998, pp. 41–46.

[30] Mohleji, S. C. and Jacobs, F., "Airspace Design and Arrival/Departure Planning for Brussels National Airport," 3rd USA/Europe Air Traffic Management R&D Seminar, Napoli, Italy, June 2000.

[31] Solveling, G., Solak, S., Clarke, J.-P., and Johnson, E., "Runway Operations Optimization in the Presence of Uncertainties," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 5, 2011, pp. 1373–1382.

[32] Cormen, T. H., Leiserson, C. E., and Rivest, R. L., *Introduction to Algorithms*, Massachusetts Institute of Technology, 3rd ed., 2009.

[33] Dmitruk, A. and Kaganovich, A., "The Hybrid Maximum Principle is a consequence of Pontryagin Maximum Principle," www.optimization-online.org/DB_FILE/2006/11/1511.pdf.

[34] http://lpsolve.sourceforge.net/.

[35] Chen, H. and Zhao, Y. J., "Sequential Dynamic Strategies for Real-time Scheduling of Terminal Traffic," *Journal of Aircraft*, Vol. 49, No. 1, 2012, pp. 237–249.

[36] Srivastava, A., "Improving departure taxi time predictions using ASDE-X surveillance data," 30th IEEE/AIAA Digital Avionics Systems Conference (DASC), McLean, VA, October 2011.