Application of Aircraft Sequencing to Minimize Departure Delays at a Busy Airport

by

Alexandre Paul Sahyoun

B.S, Ecole Centrale Paris (2012)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

Signature redacted

Author ....
Sloan School of Management
May 16, 2014

Signature redacted

Certified by ....
Amedeo R. Odoni
Professor of Aeronautics and Astronautics
and of Civil and Environmental Engineering
Thesis Supervisor

Signature redacted

Accepted by ....
Dimitris Bertsimas
Boeing Leaders for Global Operations Professor
Co-director, Operations Research Center

Application of Aircraft Sequencing to Minimize Departure Delays at a Busy Airport

by

Alexandre Paul Sahyoun

Submitted to the Sloan School of Management
on May 16, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Operations Research

Abstract

In the face of large increases in the number of passengers and flights, busy airports worldwide have been trying to optimize operating efficiency and throughput and minimize congestion on a daily basis. In the case of departures, measures can be taken at the gate, on the taxiway system or at the runway queue to minimize departure delays and/or the cost of unavoidable delays. This cost includes needless fuel consumption and noxious emissions. In this thesis, we focus primarily on runway queue optimization.
The first part of this work consists of designing a generic simulation which models specific days of operations at an airport. Using as input the schedule of operations specific to the modeled airport, the simulation processes all departures and stores the characteristic times of the process for each departing aircraft. The quantities of interest are either incrementally computed by the simulation or modeled using probability distributions derived from airport-specific data.
We then present a dynamic programming approach to sequencing departing aircraft at the runway queue. Two algorithms are presented based on the idea of Constrained Position Shifting, which maintains a high level of fairness in the order in which aircraft gain access to runways, while also improving efficiency by comparison to First Come First Served sequencing. The objective of the first algorithm is to minimize makespan, and that of the second to minimize delays. We then focus on a specific airport, which has been experiencing one of the fastest growth rates in the industry. We analyze the output of our simulation as applied to this airport and accumulate insights about congestion at the departure runways. We next apply this sequencing algorithm to this specific airport using multiple demand profiles that represent both the current traffic levels, as well as anticipated future ones that would result in more congestion. We give quantitative arguments to confirm the positive impact of the optimization on the airport's operations. We also emphasize the importance of the aircraft mix on the techniques' performance and show that the sequencing algorithms provide higher benefits (in terms of reducing delays) as the mix becomes more heterogeneous.

Thesis Supervisor: Amedeo R. Odoni
Title: Professor of Aeronautics and Astronautics
and of Civil and Environmental Engineering

Acknowledgement

It has been almost two years since I joined the Operations Research Center at MIT for my master's degree. All along the way I have learned more than I could possibly have imagined, both intellectually and personally. But I could not have done it alone, and I would like to thank everyone that has been in my life during this journey.

I will start by thanking my advisor, Professor Odoni, for his unlimited guidance and support throughout the past two years. Amedeo, I would first like to express my deepest gratitude to you for choosing me for this research project when I arrived at MIT. I was lucky to work on an applied project, which gave me an opportunity to have a real-life impact. Your incomparable intuition and massive knowledge of the field of air traffic management have directed me throughout my time here. I know how busy and solicited you are every day, but you still always found time to meet with me when I needed advice. Finally, I am extremely grateful for your valuable input and for the numerous hours you spent editing my work. Thanks again for everything, I hope one day I will be able to inspire others as much as you inspire those around you.

I would like to thank many people at the Operations Research Center. I will start by thanking the staff, especially Laura for being understanding of my challenges with deadlines. I would also like to thank Dimitris and Patrick for working so hard on making this place a great community to learn and evolve in.

Thanks also to my roommate and best friend Dani. I was extremely lucky to end up sharing this apartment at Sidpac with you since my first day at MIT. We have been through great times and I learned a lot from you, as a programmer and as a person.

I would also like to thank my friends at the ORC, both first years and second years, for all the weekends and fun times we had together. I will miss you guys, and I hope you will find a way to replace our 540 social gatherings. Thanks to Anna in particular for the last fifteen months, for always being here when I needed her, for the great times we spent discovering Boston and for editing my English writing!

Finally, I would like to thank my family, especially my parents for everything they have done for me every day of my life. Thanks for leading me to where I am now, your endless support was my main source of motivation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem and Motivation

During the last few decades, a steady increase in the number of flights and traveling passengers has compelled many airports to try to address severe congestion problems and attempt to optimize their operations to the maximum extent possible. Growth rates have been different in different parts of the world, but have been averaging about 3-4% per year overall. However, in Asia, the total number of aircraft movements experienced a 6.5% increase in 2012, while the number of passengers increased by 8% (ACI). The numbers from 2013 are very similar. Globally, IATA expects a 31% rise in passenger demand by 2017. These demand forecasts imply a much larger increase of delays at many of the worlds busiest airports.

In light of these issues, several initiatives have been proposed to improve the overall efficiency of airport operations. In 1998, Idris identified the runway as the primary bottleneck of the arrival and departure processes and emphasized the need for efficient measures to deal with this critical source of delays [10]. To minimize negative effects, active measures can be taken at the gate, on the taxiway system or at the runway queue itself. Among all the approaches used to improve these processes, we focus on two particular ones in this work. The first primarily deals with optimally timing the release of aircraft from their gates. In other words, the main idea consists of keeping a departing aircraft at its gate when an excessive number of aircraft are active on the taxiway system in order to reduce time on the taxiways and conserve fuel. The second set of techniques we consider comprise the central focus of this thesis and consist of sequencing departures at the runway queue.

There have been many papers published on the subject of sequencing airplanes on the runway to reduce delays and to maximize throughput rate. However, there are not yet any applications of these models using real data that discuss the effects of sequencing on departure delays. Most of the applications of sequencing in the literature have dealt with the sequencing of arrivals as opposed to departures.

The work presented in this paper was done in collaboration with a real airport, which we will refer to as AIRPORT throughout this paper. This airport provided us with data that we use for both modeling and optimization purposes. Our main task consists of studying whether or not the airport can benefit from the above mentioned sequencing methods. We investigated these questions and developed a simulation that models every step of the departure process of all aircraft scheduled to depart from AIRPORT on a given day. This simulation not only helps anticipate traffic problems by identifying peak congestion periods at the airport, but will also be the framework for our computational experiments when we evaluate the performance of the sequencing techniques. We show that the results depend strongly on specific properties of the airport's operations, such as its traffic levels and its aircraft mix (classification of aircraft by weight for air traffic control purposes). Although the current conditions at AIRPORT do not lead to significant improvements as a consequence of departure sequencing, we find strong results with increased traffic and a heterogeneous aircraft mix, especially when focusing on minimizing the average delay per aircraft.

1.2 Literature Review

The body of literature dealing with airport congestion management has greatly expanded over the years and is characterized by a vast array of different methodologies and approaches. For the purposes of summarizing the literature applicable to this work, we provide an overview of the main contributions made in the field with respect to two particular topics. More specifically, we focus on works addressing the evolution of the sequencing techniques introduced above, as well as the theory behind the optimal threshold above which no aircraft should be released from its gate.

We consider first papers dealing with controlling the rate at which aircraft are released from their gate. At congested airports, the taxiing times have been a significant source of delays and fuel loss. In light of this problem, many researchers have focused on methods to reduce taxing times to the minimum necessary. A large number of papers produced in the last decade focus on determining the optimal release time of an aircraft from its gate. The fundamental principle of these techniques relies on keeping the aircraft at its gate when the system is too busy. More precisely, the main motivation behind these methods is the following: as we send aircraft on the taxiway system, the departure throughput intuitively increases at first. However, there exists a threshold above which the throughput does not benefit from sending additional aircraft on the system. When the occupancy is above this threshold, we consider the system as busy. The objective is to develop a control strategy to limit the flow of aircraft getting from their gate to the taxiway system. The first work about congestion management at the apron is [9], in which Pujet, Delcaire and Feron introduce their simple N-control strategy, used in their Departure Planner. Since then, other papers delved into this matter, presenting variants of these methods. More recently in [14], Simaiakis, Sandberg, Balakrishnan and Hansman developed their own Pushback Rate Control Strategy (PRC) and tested it to quantify the savings in fuel and delays. This more sophisticated method serves as a more practical approach for real-time processing of flights. In particular, their method predicts the departure throughput over an upcoming 15 minute interval and provides the Airport Traffic Control Tower with a recommended rate at which they should release aircraft from their gate. Although we did not focus on finding the optimal threshold for our airport of interest in this paper, we did limit the number of aircraft allowed on the taxiway system to 10 aircraft. This number is a reasonable critical size for the current traffic at the modeled airport. We increase this threshold to 15 aircraft when we analyze traffic for other demand profiles.

As already noted, we will be largely dealing in this work with applying sequencing of departures at the runway in order to minimize either the total amount of time ("makespan") it takes for a string of aircraft to take off or the total delay these aircraft will suffer while waiting to take off. Roger Dear introduced such sequencing techniques for the first time in 1976, presenting them as *Constrained Position Shifting* methods, or CPS [8]. After observing the inefficiency of the First Come First Served (FCFS) discipline, he identified the need for a dynamic scheduling of landing aircraft, through sequencing in the air. The CPS techniques allow aircraft to be shifted from their FCFS position, while ensuring that the number of shifts does not exceed a pre-specified number.

These constraints on the number of shifts allowed per aircraft are defined to avoid unwanted properties, such as the potential for indefinite delay and the computational intractability of solutions in real time, which is due to the fact that the solutions have to be updated with each new arrival. Dear proves the theoretical efficiency and flexibility of these methods and led the way to other papers focusing on these new optimization methods. Psaraftis [13] focused a few years later on the tractability of these methods using dynamic programming and working with different objective functions. More specifically, he considered the problem of sequencing N identical groups of aircraft and proved these instances to be solvable in polynomial time of the number of aircraft sequenced and exponential time of the number of groups N. These methods are thus practical when the number of groups to be sequenced is small, which will be the case in this paper.

In 1990, Neuman [12] studied several sequencing algorithms and discussed their performance. He also observed that for heavy traffic, the delays vary a lot for different traffic samples, even when the same statistical parameters are used. A few years later, researchers began looking more closely at the optimization of the departures process [1]. Most of the previous research had focused on modeling the arrival flow without considering its complex coupling with departures. They were the first to deal with the sequencing of departures. This same year, Beasley introduced a mixed integer programming formulation for the arrivals sequencing problem and tested it through computational experiments [5]

In 2002, Atkins [2] presented a decision tool which provides accurate predictions of future demand and recommended a departure sequence for each runway that maximizes throughput. This tool was sponsored by the NASA Ames Research Center. Carr (2004 [7]) and Bohme (2005 [6]) also used CPS to model fairness and aimed at finding techniques that could solve efficiently the sequencing problem.

More recently, Balakrishnan and Chandran presented scalable dynamic programming algorithms scheduling arrivals under CPS [3] [4]. For different frameworks and objective functions, the authors developed an efficient approach which can be considered as based on network optimization. These are the algorithms that we will adopt for our departures sequencing work and will apply to a real airport's operations in this work. In 2008, Lee [11] also presented new dynamic programming formulations, trying to find a compromise between runway throughput and robustness.

1.3 Outline

In this thesis, we first describe in Chapter 2 a generic simulation of the departure processes that we have developed. We present the different types of delays computed by the simulation as well as the records that are maintained every time an aircraft takes off. Chapter 3 provides a detailed description of the sequencing algorithms that will be implemented and used in coordination with our simulation. We introduce two variants of an algorithm first presented by Balakrishnan & Chandran in [3]. The first one aims at minimizing the makespan, i.e. the time spent between the first and last takeoff of the sequence, while the second focuses on minimizing the average delay per aircraft in the sequence. In Chapter 4, we apply our simulation to the specific case of AIRPORT and use the set of both flight variables and global congestion variables stored for each departure to obtain insights about the traffic characteristics of the airport's operations. We then conduct a series of experiments and evaluate the impact of the sequencing algorithms on the queuing delays of departures for both the current traffic levels as well as more congested demand profiles. We show that increasing traffic intuitively improves the impact of the CPS techniques. We analyze the impact of the aircraft mix on these results and conclude that a more heterogeneous mix leads to improved optimal solutions.

Chapter 2

Simulating the departure process

2.1 Goals and Definitions

The first part of this work consists of designing a generic simulation environment that captures the characteristics of airport departure processes. Emphasis is placed on making allowances in the simulation for use of optimization features that can be applied during two specific phases of departure processes. These optimization features, which are likely to be adopted with increasing frequency at the world's busiest airports, are:

- The option to decide whether to release or not a departing flight from its gate, depending on the total number of departing aircraft which are already either taxiing toward the departure runway(s) or waiting for takeoff next to the runway.
- The possibility of sequencing departing aircraft on the takeoff runway(s) - i.e., deciding the order in which they will take off - subject to fairness constraints.

One of the objectives of the simulation will be to act as a tool for evaluating the potential of these optimization techniques to improve an airport's operations.

The main input for the simulation will be an exhaustive schedule of operations (departures and arrivals) on a given day. For each of the departures on this schedule, we would like to compute the total time of its departure process, as well as the different types of delays accumulated all along the process.

To this end, we first need to properly define this departure process, from the aircraft's release from the gate to its takeoff from the runway. We start by providing our definitions of the system $\mathcal{S}$ and of the departure process.

Definition 1. *The system $\mathcal{S}$ at a given time is defined as the set of aircraft that are pushing back from their departure gate, or are taxiing, or are queuing next to the runway for takeoff at that time.*

Definition 2. *The departure process of a given flight X is defined as the sequence of events that occur from flight X's scheduled departure time to its takeoff. We will refer to the total time of this departure process as the TTDP of the aircraft.*

We can split this process into four different phases.

- **Aircraft ready at the gate**

  The aircraft is ready to leave the gate and requests clearance to do so. If the number of aircraft that the system $\mathcal{S}$ contains at that instant is smaller than or equal to a pre-specified number, then clearance is given and the aircraft starts pushing back. Otherwise, the aircraft waits for the traffic load to become lighter.

- **Aircraft is ready to enter the taxiway system**

  After completing pushback, the aircraft is ready to leave the apron and start moving towards its runway.

- **Aircraft queuing at the runway**

  After traveling between gate and runway on the taxiway system, the aircraft reaches the departure runway and begins queuing for takeoff.

- **Aircraft takes off**

  The aircraft is in first position in the queue for takeoff. We make sure the separation requirements from the previous departure are respected, then check for possible conflict with arrivals using the same runway and release the aircraft for takeoff. Once the aircraft takes off, this marks the end of the departure process.

Note that the first phase of the process involves the concept of a crowded system. We therefore define the critical size of the system $\mathcal{S}$.

Definition 3. *The critical size of the system $\mathcal{S}$ is defined as the number C of active aircraft (pushing back, taxiing, queuing) above which no other departure is allowed to leave its gate.*

To conclude with these definitions, we observe that the evolution of the system $\mathcal{S}$ over a certain timespan is characterized by the departure processes of all flights scheduled to leave their gate in that timespan.
Now that the process has been defined, we need to identify the quantities of interest, to guide the implementation of our simulation.

2.2 Quantities to Measure and Notations

The simulation's main objectives are not only to model realistically an entire day of departure processes, but also to measure and store the different delays and other characteristic times experienced by the aircraft all along the process. These characteristic times include the following:

At the gate

- Gate delay (d_g): If the system $\mathcal{S}$ is too busy, the aircraft is kept at its gate until an aircraft takes off. The additional time spent at the gate will be considered as gate delay. It is computed incrementally during the run of a simulation.

- Runway inspection delay (d_i): Airports around the world typically inspect their runways at pre-specified times during the day for "foreign objects" that may have fallen on the surface of the runway and might impede or pose a hazard to aircraft operations. These inspections are generally referred to as inspections for Foreign Objects Damage (FOD). By far the most famous accident caused by a foreign object on a runway is the Concorde accident at the Charles de Gaulle International Airport (CDG) in Paris in 2000, which resulted in the death of all 109 people on board the airplane and of 4 more on the ground. In general, it is estimated that foreign objects cause around US$4 billion in costs to the air transport industry, due to damages to a aircraft and associated repairs, flight delays and airport maintenance [1]. Our simulation computes any delays to aircraft due to FOD inspections. We will refer to this type of delay as the "runway inspection delay".

[1] `http://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=15394`

- Waiting for clearance ($d_{clearance}$): When the aircraft is ready to leave its gate, it must request clearance before starting the pushback. Some waiting time may be incurred until a response to the request is received.
- Pushback time ($t_{pushack}$): The aircraft needs to pushback from its gate before entering the taxiway system.

On the taxiway system

- UTT: The Unimpeded Taxi Time of an aircraft corresponds to the time it would spend on the taxiway system if there was no traffic, given the aircraft's gate and the runway end from which it will depart.
- Taxiing delay (d_t): This quantity is a measure of the extra-time spent on the taxiway system due to conflicts with other arriving or departing aircraft which are also taxiing.

At the runway queue

- Queuing delay (d_q): This delay corresponds to the time spent by the aircraft on the runway queue until it is in first position. This delay is incrementally computed by the simulation, and is equal to the sum of the separation requirements between departures that were preceding our aircraft in the queue, plus any additional delay these other departures may have incurred due to conflicts with aircraft landing on the same runway.
- Conflicts with arrivals (d_a): When an aircraft reaches the first position in the departures queue, it might still encounter a conflict with one or more arriving aircraft that are expected to land on the runway. The quantity d_a is set equal to any delay due to such conflicts. Note that the quantity d_q defined above already accounts for any delays that preceding departures suffer as a result of conflicts with landing aircraft.

It is important to note at this point a critical modeling choice that has been made with regard to our simulation. We have chosen not to model the conflicts that may occur between taxiing aircraft on the taxiway system, i.e, the conflicts that will cause the taxiing delays d_t. The reason for this choice is that conflicts on taxiway systems are entirely dependent on the local geometry of the airport under consideration. Thus, to simulate taxiway conflicts, one has to develop a finest-grain representation of the taxiway network, as well as model accurately the precise movements of aircraft on that network. The development of a "microscopic" simulation of this type requires a large amount of effort and, even more important, makes it necessary to develop a different simulation tested for each modeled airport. Our objective, instead, is to develop a "mesoscopic" simulation model, which tracks every departing aircraft individually, but does not simulate the way it moves through the taxiway network. For this purpose, we propose to simulate taxiing delays, d_t, only in a statistical sense. To do so, we shall use empirical data from each modeled airport to represent d_t as a random variable with a probability density function that fits the data. (A specific example will be provided in Chapter 4). This approach simplifies greatly the development of the simulation model. It is justified by the fact that the focus of our model is on estimating the benefits that can be obtained from the two optimization features describes in the previous section (i.e., keeping departing aircraft at the gate when the taxiway system is crowded and sequencing of departures).

A number of other specific assumptions made in the simulation model will be described in Chapter 4.

In summary, the sum of all the quantities defined above gives the total time associated with the departure process of any departing aircraft. This sum will be referred to as $TTDP$ (Total Time of Departure Process) from now on. To summarize, for each aircraft:

$$TTDP = d_g + d_i + d_{clearance} + t_{pushback} + UTT + d_t + d_q + d_a \tag{2.1}$$

2.3 Data

2.3.1 Operations Data

Basic Operations Schedule: Input for the simulation

To implement the simulation, detailed data should be provided for the subject airport. Foremost is an actual or hypothetical schedule of daily operations. For each departing flight in a day of interest, this should include the scheduled time of departure, in addition to information such as flight number, gate of departure, type of aircraft, unimpeded travel time between the gate of departure and every possible end of departure runways, etc. An example of the most basic schedule data for a flight is given in Table 2.1 below.

Flight ID	Gate	Terminal	Aircraft Type	Runway End	Scheduled Time of Departure

Table 2.1: *Basic operations data, used as input for our simulation: row of headers*

Detailed TTDP Breakdown: Improved modeling

In addition to the information described above, a set of more detailed data would improve the performance of the simulation model by making it possible to calibrate certain important model parameters. Ideally, this would mean the availability of highly detailed information for a large subset of all the departures from the subject airport. An example is given in Table 2.2, in which, for one of the main airlines using the airport (or for several of the main airlines), data have been reported on the precise times associated with the various phases of the departure process of each of the flights in the dataset. Such data can help improve the modeling of certain quantities such as the pushback time $t_{pushback}$ or, more importantly, the taxiing delay d_t. The idea is to use this extensive dataset to extract some of the raw parameters of the simulation and derive new ones. As a result, we should be able to find reasonable statistical estimates for the quantities of interest.

Flight ID	Date	STD	Taxi Start	Takeoff Time	Pushback	Taxi-out

Table 2.2: *Detailed data about the TTDP breakdown of departures, used for modeling: row of headers*

2.3.2 Auxiliary Data

Unimpeded taxi times

To improve the accuracy of our estimates of taxiing times, we need airport-specific unimpeded travel times from every gate to every runway end that can be used for initiating a takeoff run.

Separation Requirements Matrix

To maintain safety, aircraft must be separated from each other according to specified air traffic control (ATC) requirements during landing and takeoff operations. The ATC separation requirement between any pair of aircraft typically depends on the maximum takeoff weights (MTOW) of the two aircraft. For this purpose, all aircraft are classified into a small number of categories. For example, the International Civil Aviation Organization (ICAO) classifies all aircraft types into four categories: **Light**, **Medium**, **Heavy**, **Super-Heavy**. The required separations are then specified for each aircraft pair. For instance, when the takeoff of a Heavy (H) is followed by the takeoff of a Medium (M) from the same runway, then the required minimum separation between the two aircraft may be 120 seconds. Conversely, if a M aircraft is followed by a H, the separation requirement may be 90 seconds.

These separation requirements are typically provided in the form of a matrix of separations. An example of such a matrix of separations for departures from the same runway is given below as equation 2.2, where the column indicates the leading aircraft of the pair and the row indicates the trailing aircraft in the pair. The separations are given in seconds.

$$T_{separation} = \begin{array}{c c} & \begin{array}{cccc} SH & H & M & L \end{array} \\ \begin{array}{c} SH \\ H \\ M \\ L \end{array} & \begin{pmatrix} 150 & 90 & 90 & 90 \\ 150 & 90 & 90 & 90 \\ 180 & 120 & 90 & 90 \\ 180 & 120 & 90 & 90 \end{pmatrix} \end{array} \tag{2.2}$$

The optimization of departures sequences, which is the main focus of this study, will rely heavily on this matrix, as will be seen in Chapter 3. It should be noted that the separations indicated in the matrix of equation 2.2 vary from airport to airport, as they depend on the practices of ATC operations providers in each country.

2.4 The implementation

We have now described the generic departure process that is modeled by our simulation, as well as defined the different quantities that will be measured and stored.
To execute the departure process according to the defined rules, we have implemented an event-paced simulation using Python as our programming language. We use an object called a Priority Queue as the principal way to capture the processing of departures. This object turns out to be very well suited to our needs. The entries contained in the Priority Queue are kept sorted according to an attribute of our choice, in such a way that the lowest valued entry is retrieved first.

For each of the ends of the departure runways, we create a Priority Queue and include in it all the departures that are scheduled to leave from that end, sorted by scheduled departure time. We then process these queues separately, one at a time. The processing of departures in the Priority Queue associated with any particular runway end on a simulated day works as follows:

All aircraft start in the same state, namely *"At the gate"*. We begin by retrieving the first aircraft from the queue, which will be the first aircraft scheduled to depart from that runway end on that day. Once the aircraft requests clearance and receives a response, the aircraft begins pushing back, as long as no runway runway inspection is taking place and the system $\mathcal{S}$ is not in critical size. The aircraft is now in a new state: *"About to taxi"* and is placed back in the queue with his updated current time. We then repeat the process, retrieving the first aircraft in the Priority Queue and processing it depending on the state it is in. The simulation stops when all of the departures scheduled have gone through the four different stages: *"At the gate"*, *"About to taxi"*, *"At runway"*, *"Taking off"*. At that point, all the departures are processed, which means all aircraft have taken off.
A detailed description of what happens in each stage of the departure process for a given aircraft is as follows:

- *"At the gate"*
    - If the system $\mathcal{S}$ is not in critical size:

        * If there is no time overlap with a period when runway FOD inspections are being conducted:
            - Add clearance delay and pushback time to the aircraft.
            - Append the aircraft to the list of departures on the {*Apron*} system.
            - Set its status to *"About to taxi"* and place back in queue.
        * If runway inspection is currently taking place:
            - Add runway inspection delay and put the aircraft back in the queue.
    - If the system is currently in critical size:
        * Append the aircraft to the list of departures delayed at the gate due to traffic

- *"About to taxi"*
    - Remove the aircraft from the list of departures on the {*Apron*} system.
    - Add UTT and taxiing delay to the TTDP of the aircraft.
    - Append the aircraft to the list of departures on the {*Taxiway*} system.
    - Set its status to *"At Runway"* and place back in the queue.

- *"At the runway"*
    - If this is the first time we check that aircraft at the runway:
        * Remove the aircraft from the list of departures on the {*Taxiway*} system.
        * Append the aircraft to the list of departures on the {*Runway*} system.
    - If the aircraft was already at the runway the last time it was retrieved from the queue:
        * If the aircraft is in first position in the queue:
            - Compute and store its queuing delay given its time of arrival at the runway and update its actual time.
            - Check separation requirements given the previous takeoff's weight class and potentially increment the queuing delay.
            - Check interference with arrivals on the runway and add potential delay due to conflicts with landing aircraft.

        · Set the aircraft's status to *"Taking-off"* and place back in the queue.

      * If the aircraft is not in first position in the queue:

        · Update queuing delay of the aircraft by setting its new time to the time of the most recent takeoff.

        · Place the item back in the queue

- *"Taking-off"*
  - Update positions of all the aircraft in the runway queue.
  - Add a row to the output file with all of the data stored for that flight.
  - If there is an aircraft waiting to leave its gate, store that aircraft's gate delay and place it back in the queue.

2.5 Output of the simulation

When we run the simulation for any specific airport, using as input the schedule of departures it provided us for a given day, we process all flights from their gate to their takeoff. For every single departure being processed, we record the set of event times we are interested in. Table 2.3 shows an example, for a random departure schedule on a given day.

Flight ID	*TTDP*	d_g	d_i	$d_{clearance}$	$t_{pushback}$	UTT	d_t	d_q	d_a
Flight 123	34	7	0	5	8	6	3	3	2

Table 2.3: *Output of the simulation for a random departure. The values were not taken from actual results.*

Using these first level outputs, the simulation also plots for the user the following charts:

- Evolution of TTDP with time of day
- Comparison of actual and computed TTDP (when the simulation is run using historic data).
- Detailed breakdown of the TTDPs of all the departures processed on that day. This chart allows us to measure the impact of each type of delay on the TTDP, as well as the evolution of this impact with the time of day.

Moreover, the simulation has been designed such that at any point in time, we have access to the occupancies of each part of the system. This additional data allows us to output other charts, which display the evolution of these occupancies with the time of day. More precisely, the simulation outputs:

- The total occupancy of the system S
- The occupancies of the apron, the taxiway system and the runway queue in three different charts
- A timeline of whether the runway queue is busy or not, where "busy" is defined as a binary variable which takes the value 1 when there is at least one aircraft in the queue and 0 otherwise. This provides a visualization of the length of the busy periods of the runway during the day.

Chapter 3

The optimization

3.1 Presentation

As introduced in the previous chapter, every aircraft departing from AIRPORT has to go through the departure process, from pushback to takeoff. More specifically, each plane starts at the apron then circulates on the taxiway system before queuing at the runway. Ideally, we would like to optimize the airport's operations by acting at each step of the process. In this work, we did not apply any method to improve the performance of the modeled airport on its taxiway system. At the apron, our main action is to keep aircraft at their gate when the system S is too crowded, by setting C optimally. Although recent research has determined the existence of a threshold above which it is counter-productive to send aircraft on the taxiway, we did not focus on this optimization matter. In this work, we will focus on the runway queue optimization, using sequencing techniques at the runway.

As introduced earlier, two consecutive aircraft taking off from the same runway have to respect separation requirements, which depend on their corresponding weight classes. The aircraft are assigned to one of four classes according to their size: L (*light*), M (*medium*), H (*heavy*), SH (*super heavy*). The matrix of separation requirements then defines the waiting times between every two types of aircraft.

In summary, the sequencing techniques aim at reordering aircraft in the queue in order to minimize delays that are created by these separation requirements. To remain practical, we give an upper bound to the number of shifts allowed per aircraft, adding fairness constraints to our optimization. From now on, we will refer to this upper bound as K.

The input for these methods consists of a list of aircraft to sequence, associated with their First-Come First-Serve (FCFS) arrival times at the runway queue. Suppose an aircraft arrives in position p_1 at the runway queue according to the First-Come-First-Serve process. Then this aircraft can occupy $2K+1$ positions p_2 in the reordered queue:

$$p_1 - K \leq p_2 \leq p_1 + K \tag{3.1}$$

This chapter focuses on the methodology and introduces the two main versions of the sequencing algorithm, which differ from each other only by the objective function to minimize. We will then apply these alternatives to a specific airport in Chapter 4.

3.2 The sequencing algorithm

Roger Dear was the first one to write about sequencing techniques at the runway (Dear, 1976 [8]). In the literature, these techniques are known under the name of *Constrained Position Shifting* methods, or CPS methods. Since then, some research has been done to find efficient algorithms for practical implementation. We chose to implement the algorithm introduced by Balakrishnan & Chandran [4]. In this section, we will present their work.

3.2.1 Sequencing to minimize makespan

The first version of the algorithm aims at minimizing what we call the makespan, i.e. the difference between the takeoff times of the first and last departures. The specificity of this algorithm is that it involves building an initial network, where each feasible sequence is represented by a path in the network. Our first focus will consist of a detailed description of this *CPS* network.

The CPS network

We let n be the number of aircraft to be sequenced and we consider a series of such aircraft labeled according to their FCFS order at the runway queue, i.e. $(1, 2, ..., n)$.

The CPS network is made of n stages, where a stage is an index associated with a position in the queue (1 through n). The network follows the following rules:

- The position of an aircraft in a node corresponds to a position in the final sequence. More specifically, the last aircraft from a node in stage p leaves in position p. For example, let $K = 1$ and $p = 4$. If (i, j, k) is a node from stage p, then for every path of the network which includes this node, i is the aircraft which leaves in position $p - 2 = 2$, j in position 3 and k in position 4.
- A node in stage p is of length $\min(2K + 1, p)$.
- The nodes in stage p represent all feasible combinations of aircraft at that stage. Using the same example as above, $(3, 2, 4)$ is a possible combination of aircraft leaving in second, third and fourth position: $(3, 2, 4)$ is therefore a feasible node in stage 4. However, $(4, 2, 3)$ is not a feasible node, since aircraft 4 would leave in position 2, which is not allowed when $K = 1$.
- After building the network, we perform both a forward and a backward search to remove nodes that do not belong to any path: we are pruning the network.

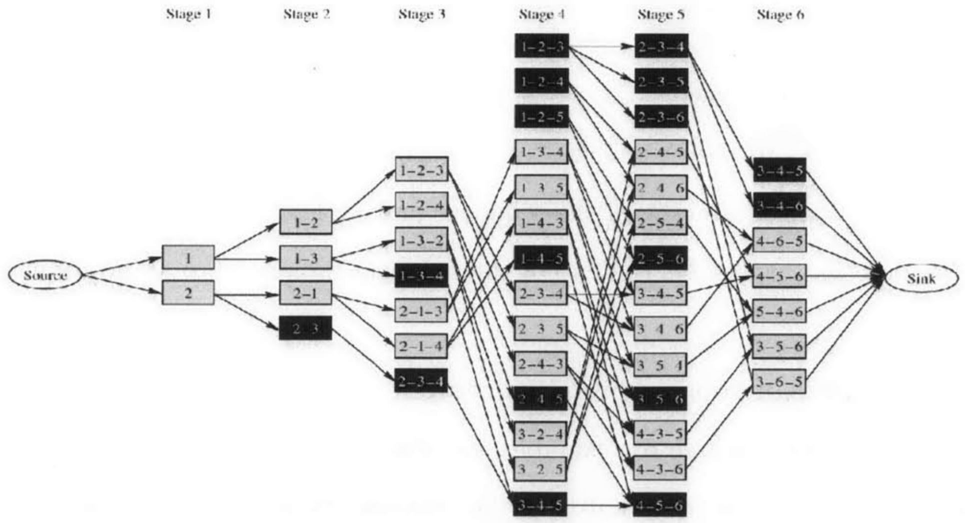

Figure 3-1: *CPS network in the case $n = 6$, $K = 1$. The dark nodes are the ones that were removed from the network after pruning. This image was directly taken from [4].*

As an example, we can build the network in the case where $K = 1$ and $n = 6$ shown in Figure 3-1. Consider six aircraft labeled 1 to 6 according to their FCFS position. Building the CPS network would work as follows:

Stage 1

In stage 1, a node includes a single aircraft. The only two aircraft which can leave in first position in the reordered sequence are the ones that were originally in positions 1 and 2 in the FCFS sequence.

Stage 2

In stage 2, a node includes two aircraft, namely the ones leaving in first and second position in the corresponding new sequence. The aircraft that can leave in second position are the ones labeled 1, 2 or 3. Given this observation we build the nodes $(1,2)$, $(1,3)$, $(2,1)$, $(2,3)$. Then we link these new nodes to their predecessors in stage 1.

Stage 3

From stage 3 to stage 6, all nodes are of length $2K+1=3$. A node in stage 3 contains thus the aircraft leaving in first, second and third positions. Again, given that each aircraft obviously occupies exactly one position and there are three candidates for each of these positions, we add a node for each combination of aircraft which satisfies the fairness constraints defined by K: $(1,2,3)$, $(1,2,4)$, $(1,3,2)$, $(1,3,4)$, $(2,1,3)$, $(2,1,4)$, $(2,3,4)$. Finally, we add links from these new nodes to their predecessors in stage 2.
We complete the network following the same procedure for the last three stages, always adding links from the new nodes to their predecessors. To complete the initial network, we add a source node corresponding to the beginning of the sequence and we link it to nodes in stage 1. Similarly, we add a sink node that represents the end of the sequence, linking it to all nodes in stage n.

The last step consists of pruning this initial network by completing a forward then a backward search. During the forward search for example, we observe that node $(4,5,6)$ in stage 5 does not have any successor and we thus remove it from the network. Moreover, everytime we delete a node, we need to delete its predecessors that do not have any successor other than that node. In this specific case, the absence of successor for node $(4,5,6)$ leads to the deletion of the following nodes:

- $(4,5,6)$ from stage 5
- $(3,4,5)$ from stage 4

- $(2,3,4)$ and $(1,3,4)$ from stage 3
- $(2,3)$ from stage 2

Regarding the implementation of this CPS network, we made it more efficient by only building nodes from stage p starting from the nodes from stage $p-1$. This allows us to avoid building nodes that do not have a predecessor, which makes the pruning easier. For large values of n, this method proves to be very useful. We can now focus on the optimization itself, which highly relies on the CPS network.

Notations

Other than the number of aircraft n, the matrix T and the K parameter, we also use the following notations to describe the algorithm:

- $\mathcal{N}_{CPS}$ is the pruned CPS network.
- $x(i)$ is the takeoff time of the final aircraft of node i.
- $x^*(i)$ is the takeoff time of the final aircraft of node i in the optimal sequence.
- $\mathcal{N}(p)$ is the set of nodes in stage p.
- $e(i)$ is the FCFS time of arrival at the runway queue of the last aircraft in node i.
- $P(i)$ is the set of predecessors of node i in the CPS network.

The constraints

There are three sets of constraints that we involve in our optimization.

- The separation requirements T_{ij} between two consecutive nodes i and j. We define the separation requirements between two nodes i and j to be the separation requirements between the last aircraft of node i and the last aircraft of node j.
- The maximum number of shifts allowed per aircraft, that we refer to as K. In practice, K can be equal to 1 or 2.

- The arrival time at the runway queue for the last aircraft of node i, namely $e(i)$, which we will need to include in the implementation to make sure that no departure can take off before its arrival at the runway.

Formulation

In summary, if i_p is a node in stage p from the $\mathcal{N}_{CPS}$ network, we would like to find a solution to the following instance:

$$\begin{aligned} \underset{(i_1,\dots,i_n)\in\mathcal{N}_{CPS}}{\text{minimize}} \quad & x(i_n) \\ \text{subject to} \quad & x(i_p) - x(i_{p-1}) \geq T_{i_{p-1}i_p},\ p = 1,\dots,n. \\ & x(i_p) \geq e(i_p),\ p = 1,\dots,n. \\ & i_p \in \mathcal{N}(p),\ p = 1,\dots,n. \end{aligned} \tag{3.2}$$

Dynamic programming for optimization

We use dynamic programming to find the feasible sequence that minimizes the makespan. The idea is to compute recursively the values of $x^*(j)$ for all nodes j in all stages.
More specifically, for a node i_p in stage p, we compute $x^*(i_p)$ using the following formula.

$$\forall p \geq 2, \quad \forall i_p \in \mathcal{N}(p) \qquad x^*(i_p) = \max\{e(i_p), \min_{i_{p-1}\in P(i_p)} [x^*(i_{p-1}) + T_{i_p i_{p-1}}]\}, \tag{3.3}$$

$$\forall i_1 \in \mathcal{N}(1) \qquad x^*(i_1) = e(i_1) \tag{3.4}$$

The optimal makespan is then obtained by solving:

$$\min_{i_n\in\mathcal{N}(n)} x^*(i_n) \tag{3.5}$$

To recover the optimal sequence $(i_1^*,\dots,i_n^*)$, we start with our optimal node i_n^* in $\mathcal{N}(n)$, then recover its best predecessor i_{n-1}^* from stage $n-1$. By repeating the process until we get to stage 1, we extract the optimal sequence. More precisely:

$$\begin{aligned} i_n^* &= \arg\max_{i_n \in \mathcal{N}(n)} x^*(i_n), \\ i_{n-1}^* &= \arg\max_{i_{n-1} \in P(i_n^*)} x^*(i_{n-1}), \\ &\dots \\ i_1^* &= \arg\max_{i_1 \in P(i_2^*)} x^*(i_1), \end{aligned} \tag{3.6}$$

Complexity

- A node from the network has at most $2K+1$ positions filled by as many aircraft. Each of these positions can be occupied by $2K+1$ different aircraft. The number of nodes in each stage is therefore $O((2K+1)^{2K+1})$ and the number of nodes in the network is $O(n(2K+1)^{2K+1})$.

- Moreover, a node in stage p is obtained from its predecessor only by removing the first aircraft and adding one of $2K+1$ possible aircraft as the p^{th} aircraft to take off. From this observation, we note that a node has at most $2K+1$ predecessors. For each node in the network, we can therefore create at most $2K+1$ links to the previous stage; the total numbers of arcs in the network can then be estimated as the number of nodes in the network multiplied by $2K+1$. In other words, the network counts $O(n(2K+1)^{2K+2})$ arcs.

- Our forward and backward searches imply going through every arc at most twice.

- The dynamic programming goes through every arc once.

The complexity of the algorithm is thus equal to the number of arcs in the CPS network, which is $O(n(2K+1)^{2K+2})$.

We can also point out that solving this problem is equivalent to solving a shortest path problem within the CPS network, where each link between nodes is associated with a cost equal to the corresponding separation requirement.

3.2.2 Sequencing to minimize average delay

In this alternative, we aim at minimizing the average delay per aircraft. Although we still perform the optimization using dynamic programming through a network, changing the objective function actually involves significant changes in the implementation. We will describe here the detailed algorithm introduced by the same authors in 2007 [3], starting with the definition of the average delay network.

The average delay network

To build the average delay network $\mathcal{N}_{Avg\,Delay}$, we start from the basic CPS network $\mathcal{N}_{CPS}$, introduced previously. The idea behind this network is slightly more involved than the previous one. We will hence need additional notations to present our new optimization problem.

- For a given final sequence defined by nodes $(i_1, \ldots, i_n)$, we let $x(i_p)$, $p \in \{1...n\}$, be the departure time of the p^{th} aircraft of that sequence, exactly as in the previous formulation.
- Every node i_p from stage p in the original CPS network is associated with $n-p+1$ subnodes in the new network, that we will refer to as (i_p, j), where $j \in \{1...n-p+1\}$.
- Given a partial sequence $(i_1, \ldots, i_p)$, each subnode j of the node i_p is associated with the value $f(i_p, j) = x_(i_1) + x(i_2) + \ldots + x(i_{p-1}) + jx(i_p)$.

To understand the intuition behind the use of the function $f(i_p, j)$, we need to start by looking at the objective function. We would like to find the sequence that minimizes the sum of all the delays. To do so, we start by defining the delay of an aircraft in the sequence.

Definition 4. *Let $(i_1, \ldots, i_n)$ be a feasible sequence from the CPS original network $\mathcal{N}_{CPS}$.*
Let $x(i_p)$ be the takeoff time of the p^{th} aircraft in the sequence, and $e(i_p)$ be the FCFS arrival time at the runway queue of the last aircraft in node i_p. Then, the delay of the p^{th} aircraft to take off in the final sequence is defined as:

$$x(i_p) - e(i_p) \tag{3.7}$$

Given definition 4, we can now define our new optimization problem as follows:

$$\begin{aligned} \underset{(i_1,\ldots,i_n)\in\mathcal{N}_{CPS}}{\text{minimize}} \quad & (x(i_1)-e(i_1))+(x(i_2)-e(i_2))+\ldots+(x(i_n)-e(i_n)) \\ \text{subject to} \quad & x(i_p)-x(i_{p-1}) \geq T_{i_{p-1}i_p},\ p=1,\ldots,n. \\ & x(i_p) \geq e(i_p),\ p=1,\ldots,n. \\ & i_p \in \mathcal{N}(p),\ p=1,\ldots,n. \end{aligned} \tag{3.8}$$

This minimization is done over all possible sequences $(i_1, i_2, \ldots, i_n)$, i.e for all paths from the original CPS network. Moreover the constraints are the same than the ones introduced in the previous formulation. It is important to note that, for any path $(i_1, \ldots, i_n)$ in the CPS network, each aircraft at the queue is the last aircraft of exactly one node in the sequence. The quantity $\sum_{i=1..n} e(i_p)$ is therefore equal to the sum of the FCFS arrival times at the queue of all n aircraft, and is thus independent of the selected path. We can therefore write the problem in a more compact way as follows:

$$\begin{aligned} \underset{(i_1,\ldots,i_n)\in\mathcal{N}_{CPS}}{\text{minimize}} \quad & x(i_1)+x(i_2)+\ldots+x(i_n) \\ \text{subject to} \quad & x(i_p)-x(i_{p-1}) \geq T_{i_{p-1}i_p},\ p=1,\ldots,n. \\ & x(i_p) \geq e(i_p),\ p=1,\ldots,n. \\ & i_p \in \mathcal{N}(p),\ p=1,\ldots,n. \end{aligned} \tag{3.9}$$

We can then make the following observations:

- Solving our problem is equivalent to looking at all paths of the network and choosing the one with the minimum value of $f(i_n, 1)$. Note that there is one subnode per node in stage n.

- For each predecessor i_{n-1} of a node i_n, there are 2 subnodes to consider.

  – Case (1): The n^{th} aircraft left at its FCFS time. In this case, $x(i_n) = e(i_n)$, which implies:

$$\begin{aligned} f(i_n, 1) &= x(i_1) + x(i_2) + \ldots + x(i_n) \\ &= x(i_1) + x(i_2) + \ldots + x(i_{n-1}) + e(i_n) \quad (3.10) \\ &= f(i_{n-1}, 1) + e(i_n) \end{aligned}$$

– Case (2): The n^{th} aircraft left directly after the previous aircraft, which means it had to wait for the corresponding separation requirements. In such case, $x(i_n) = x(i_{n-1}) + T_{i_n i_{n-1}}$, which implies:

$$\begin{aligned} f(i_n, 1) &= x(i_1) + x(i_2) + \ldots + x(i_n) \\ &= x(i_1) + x(i_2) + \ldots + x(i_{n-1}) + (x(i_{n-1}) + T_{i_n i_{n-1}}) \quad (3.11) \\ &= f(i_{n-1}, 2) + T_{i_n i_{n-1}} \end{aligned}$$

- More generally, let i_p be a node in stage p and i_{p-1} be one of its predecessors. By repeating the process introduced above, we observe that for every subnode (i_p, j) of i_p, there are two predecessors among the subnodes of i_{p-1}, namely $(i_{p-1}, 1)$ and $(i_{p-1}, j+1)$, which are linked in the following way:

$$f(i_p, j) = \begin{cases} f(i_{p-1}, j+1) + jT_{i_p i_{p-1}}, & \text{if } x(i_p) = x(i_{p-1}) + T_{i_p i_{p-1}} \\ f(i_{p-1}, 1) + j.e(i_p), & \text{if } x(i_p) = e(i_p) \end{cases} \quad (3.12)$$

This reasoning helps us understand the idea behind these values $f(i_p, j)$ and the links between subnodes at different stages. In summary, we build the network adding links as follows:
For every pair of nodes (i_p, i_{p-1}) where i_{p-1} is a predecessor of i_p:
- Create a link between subnodes of the form (i_p, j) and $(i_{p-1}, j+1)$, associated with a cost of $jT_{i_p i_{p-1}}$.
- Create a link between subnodes of the form (i_p, j) and $(i_{p-1}, 1)$, associated with a cost of $j.e(i_p)$.
- Finally, add links from each subnode in stage 1 to the source node and from each subnode in stage n to the sink node.

Solving for the sequence with the minimum average delay is again equivalent to solving a shortest path problem using the $\mathcal{N}_{AvgDelay}$ network and the costs defined above. Figure 3-2 shows a single sequence from the new network, in the case where $n = 6$, $K = 1$.

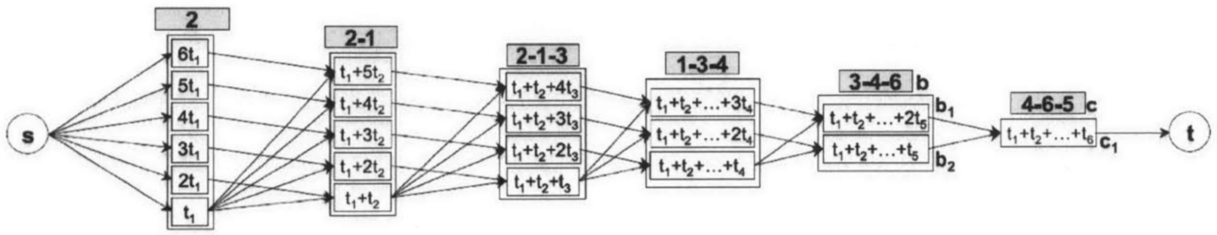

Figure 3-2: *Fraction of the $\mathcal{N}_{Avg\,Delay}$ network, corresponding to the sequence* $(2, 1, 3, 4, 6, 5)$. *The figure only displays the nodes relative to that sequence as well as the subnodes corresponding to each stage. This image was directly taken from [3].*

Dynamic programming for optimization

When looking for the best predecessor of a node i_n in stage n, we loop through its predecessors. For each of these predecessors i_{n-1}, we determine in which case would node i_n's last aircraft take off. This is equivalent to determining which subnode to consider within i_{n-1}, which is done by finding the $\max\left(f^*(i_{n-1}, 1) + e(i_n), f^*(i_{n-1}, 2) + T_{i_{n-1}i_n}\right)$.

We then find the best among these predecessors, storing the predecessor itself from $\mathcal{N}_{CPS}$ as well as the relevant subnode. This is done by solving the following problem.

$$f^*(i_n, 1) = \min_{i_{n-1} \in P(i_n)} \left\{ \max\left(f^*(i_{n-1}, 1) + e(i_n), f^*(i_{n-1}, 2) + T_{i_{n-1}i_n}\right) \right\} \tag{3.13}$$

More generally, at the p^{th} step of the algorithm, we compute the values of $f^*(i_p, j)$ for all nodes

i_p and their subnodes $j = 1, \ldots, n - p + 1$.

$$- \quad \forall i \in \mathcal{N}(1), \quad \forall j \in \{1, ..., n\}, \qquad f^*(i_1, j) = je(i_1), \tag{3.14}$$

$$\begin{aligned} - \quad & \forall p \geq 2, \quad \forall i \in \mathcal{N}(p), \quad \forall j \in \{1, ..., n-p+1\}, \\ & f^*(i_p, j) = \min_{i_{p-1} \in P(i_p)} \{\max \left(f^*(i_{p-1}, 1) + j.e(i_p), f^*(i_{p-1}, j+1) + j.T_{i_p i_{p-1}}\right)\} \end{aligned} \tag{3.15}$$

To extract the optimal sequence, we proceed exactly as we did in the previous case. The only difference is that, while backtracking stage by stage, we store not only the best predecessor from $\mathcal{N}_{CPS}$ but also the subnode from $\mathcal{N}_{AvgDelay}$ that tells us in which case each aircraft took off (at its FCFS time or after waiting to fulfill separation requirements).

Complexity

The average delay network is slightly more complex than the basic CPS network. In the end, this alternative of the algorithm is extremely similar to the first one, the main difference being that we are working at the subnodes level, which means we are dealing with a larger network. The total number of subnodes is approximately equal to the number of nodes from the CPS network, $O(n(2K+1)^{2K+1})$, times the number of subnodes per nodes, which can be upper bounded by n. In other words, the number of nodes is $O(n^2(2K+1)^{2K+1})$. Given that each subnodes only has 2 predecessors, we conclude that the number of arcs is also $O(n^2(2K+1)^{2K+1})$. Therefore, the overall complexity of the algorithm is this time $O(n^2(2K+1)^{2K+2})$.

Chapter 4

Computational Experiments: Application to a Busy Airport

In the previous chapter, we introduced Constrained Position Shifting methods to sequence departing aircraft in the runway queue. Now that we have presented a particular implementation of these sequencing techniques, the next step consists of analyzing their performance on actual data. In this chapter, we apply our implementation to optimize departure operations at a specific airport, that we will refer to as AIRPORT from now on, for privacy issues.

Initially, we use the generic simulation introduced in Chapter 2 and perform multiple runs with data which is specific to the airport's operations. Then, we analyze the results of these runs to get some preliminary insights on the airport's current traffic levels. The idea is to gain a general understanding of the airport's demand and to identify the main congestion periods on the departure runways. Based on this study, we will then run our optimization on specific intervals of time, to maximize the techniques' performance.

We observed that although AIRPORT has been growing particularly fast in the last decade, the current demand is still relatively low compared to other major airports around the world. We will therefore artificially increase traffic in order to apply our sequencing methods to different demand profiles.

4.1 Simulating AIRPORT's departure processes

4.1.1 The modeled airport at a glance

Before we run our generic simulation on the data provided to us by AIRPORT, we first need to introduce basic information about the airport's operations.

Runways

The airport owns two runways, namely 09L-27R and 09R-27L . Most departures take off from 09L-27R . Although some departures do take off from the other runway, the takeoff times are too spread-out for us to obtain interesting results from the simulation and the optimization. We will thus focus on the runway 09L-27R. Table 4.1 shows the number of departing aircraft per runway end.

Runway End	09L	27R	09R	27L
Count	260	170	32	3
Fraction (%)	56	36	7	1

Table 4.1: *Counts of departures per runway end. 09L and 27R are two ends from the same runway. Same for 09R and 27L.*

Schedule of departures

Depending on the time of day, aircraft scheduled to leave from our departures runway will take off either from 09L or 27R, according to the following schedule:

- 00:00-06:00 and 12:00-18:00: Departures take off from 27R.
- 06:00-12:00 and 18:00-00:00: Departures take off from 09L.

Figure 4-1 shows the hourly breakdown of the departures' STDs, which gives us a first idea of peak hours at the airport. Note however that it is more important to us to get information about peak hours at the runway queue itself. We are able to predict this breakdown of the arrival times in the queue using our simulation, as will be shown later.

At a glance, it seems like peak hours at the runway appear around 8:00, 13:00 and later in the day around 17:00. However, the corresponding peak values stay below 30 aircraft per hour.

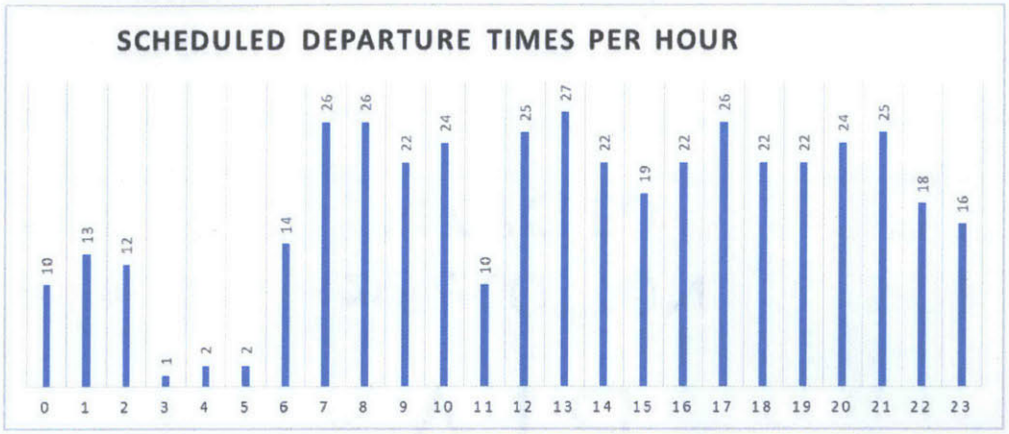

Figure 4-1: *Hourly breakdown of the departures' STDs at AIRPORT.*

We can therefore also expect the rate of aircraft per hour at the runway queue to remain below this threshold. According to Balakrishnan and Chandran [4], trying to minimize makespan does not have any impact for such low demand profiles. These preliminary observations give us a first incentive to artificially increase traffic and create new demand profiles. Only then can we expect our CPS techniques to lead to meaningful results.

Growth

In recent years, the airport has been growing significantly faster than most airports around the world. Figure 4-2 shows the evolution of the total number of aircraft movements (departures and arrivals) in the last five years. The numbers imply a steady increase of more than 5% per year. These observations emphasize the need to anticipate a significant increase of the demand in upcoming years. We will therefore present in this chapter an approximate way to artificially increase traffic and run simulations for higher demand profiles.

Now that we have introduced some basic characteristics of AIRPORT's operations, we are ready to run the simulation introduced in Chapter 2, using their operations data. In the next section, we focus on the outcome of these runs to identify the congestion periods at the runway queue and gain some insight before applying sequencing.

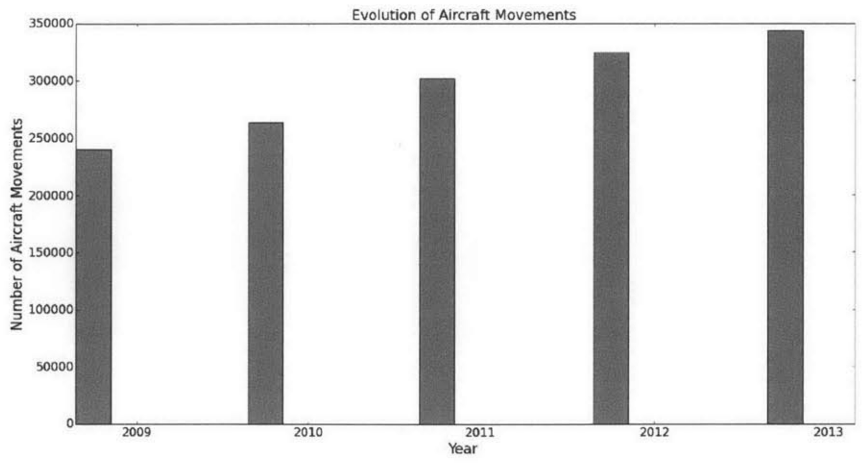

Figure 4-2: *Evolution of the total number of aircraft movements per year at AIRPORT. We observe a stable annual increase of approximately* 5%.

4.1.2 Data

Input for the simulation

The simulation runs are completed using AIRPORT-specific data provided to us for this series of experiments.

The first type of data we use consists of a whole month of scheduled operations (departures and arrivals). From this dataset, we extracted a specific day of operations corresponding to all flights scheduled to depart or land on Friday, October 19^{th}, 2012. This subset of data will be used as input for our simulation runs. Note that we chose to experiment on a Friday, which is the day of the week with the highest demand at most major airports.

Auxiliary Data

Matrix of separation requirements

AIRPORT also provided us with two types of data that will be used for modeling the quantities of interest as well as for the implementation of the simulation.

The first type is the airport's specific matrix of separation requirements between departures, shown in equation 4.1, where the columns indicate the leading aircraft and the rows the trailing aircraft. This matrix will be crucial for both the optimization and the computation of the queuing delays of all aircraft departing from AIRPORT.

$$T_{separation} = \begin{array}{c|cccc} & SH & H & M & L \\ SH & 150 & 90 & 90 & 90 \\ H & 150 & 90 & 90 & 90 \\ M & 180 & 120 & 90 & 90 \\ L & 180 & 120 & 90 & 90 \end{array} \tag{4.1}$$

Table of Unimpeded Travel Times (UTTs)

The second type of data consists of a table that gives us access to the set of Unimpeded Travel Times (UTT) from every gate to every runway end that can be used for initiating a takeoff run. These taxiing times will allow us to increase the precision of our taxiing time estimates. Moreover, we will see in the following paragraphs that these UTTs will help derive the taxiing delay from the raw data given to us by AIRPORT. Note that a few pairs (Gate, Runway End) were not associated with a UTT. To deal with this missing data, we simply used a detailed map of the airport which displays all gates and runway ends. We then used simple extrapolation to obtain the missing UTTs, based on the UTTs of neighboring pairs.

Modeling quantities of interest

In Chapter 1, we emphasized the importance of getting access to additional data that would provide us with more detailed information about the departure processes of all flights taking off from the airport. AIRPORT has provided detailed data for one of the most active airlines there, which accounts for approximately 20% of departure from a specific week of operations during that month of October. More specifically, for 20% of departures from that week, we have the following data:

ID	Date	Gate	Runway End	STD	Taxi Start	Takeoff Time	Pushback	Taxi Time

Table 4.2: *Detailed data about the TTDP breakdown of departures at AIRPORT used for modeling: row of headers.*

With a sample size of more than 700 observations, we are able to find reasonable statistical estimates to model pushback time $t_{pushback}$ and taxiing delay d_t.

Pushback time

For the first of these quantities, we constructed a histogram of the values of the pushback time in our dataset and decided to model this variable as if drawn from a normal distribution with mean $\mu = 6$ minutes and standard deviation $\sigma = 2$ minutes. Figure 4-3 displays both the histogram and the fit of the corresponding normal distribution.

Taxi delay

As explained in Chapter 1, we made a critical decision regarding the modeling of taxiing delays at this airport. More specifically, we decided to provide a statistical estimate of that quantity instead of modeling conflicts between aircraft on the taxiway system. From the variables shown in Figure 4.2, we extracted the taxiing delay using the table of Unimpeded Travel Times (UTT). In particular, for each flight in that dataset we defined the taxiing delay as follows:

$$d_t = Taxiing\ Time \ - \ UTT(Gate,\ Runway\ End) \tag{4.2}$$

We then drew a histogram of the values derived taxiing delay d_t in the dataset and decided to model that delay as a random variable drawn from a Gumbel distribution with parameters $\mu = 4.7$ minutes and $\sigma = 4.6$ minutes. Note that if X is a random variable drawn from such a distribution, μ is a location parameter and σ is the standard deviation, such that:

$$\begin{aligned} E[X] &= \mu + \gamma\beta, \\ \sigma &= \frac{\beta\pi}{\sqrt{6}} \end{aligned} \tag{4.3}$$

Note that γ is the Euler-Mascheroni constant and β is a scale parameter. It is interesting to note that the mean of our modeled taxiing delay is approximately equal to 6 minutes and 40 seconds. However, the standard deviation is relatively high, which shows how variable the taxiing delay can be at AIRPORT. Moreover, note that a small number of flights have negative taxiing delay in our dataset. This means that for some flights processed by our simulation, the total time spent on the taxiway system will be slightly lower than the corresponding UTT, which appears to be contradicting the definition of the UTT. Since these negative taxiing delays rarely exceed 1 or 2 minutes and the probability of getting a negative value is relatively low (around 0.1), we will not consider this as an issue for the purposes of our study. Figure 4-4 below shows the histogram and the fit.

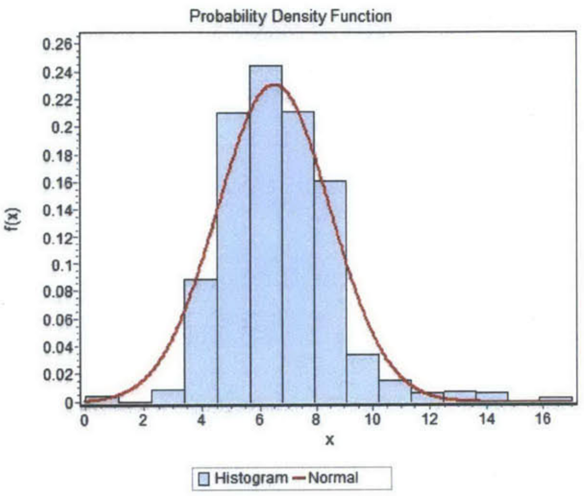

Figure 4-3: *Distribution of the pushback times values, for a sample size of approximately 700 observations.*

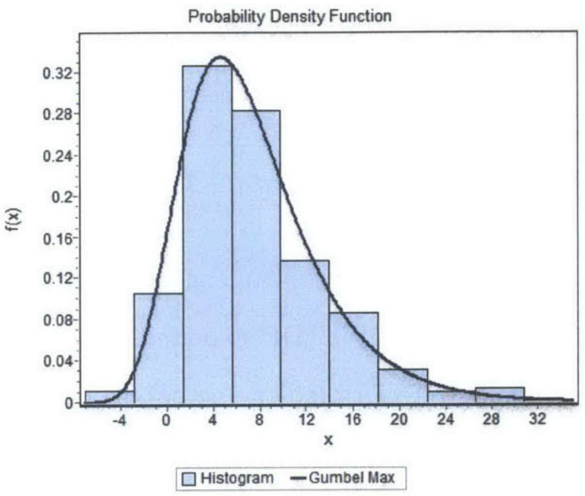

Figure 4-4: *Distribution of the taxiing delay d_t values, for a sample size of approximately 700 observations*

The modeling presented above was integrated in the implementation, hence increasing the precision of the simulation. We are now able to run the simulation for the specific case of AIRPORT. The results are the subject of the next section.

4.1.3 Output of the simulation for the modeled airport

We made several initial runs of the simulation to get a better understanding of the traffic at AIRPORT. This step of the study is crucial because it gives us insight into ways of applying the upcoming optimization for this specific airport. Every major airport, such as the one we are modeling, has its own congestion characteristics. Our generic simulation helps in identifying these airport-specific features through a thorough analysis of its output.

General observations about traffic on the system $\mathcal{S}$

For these first runs, we use as input the schedule of operations from Friday, October 19^{th}, 2012. The critical size $\mathcal{C}$ of the system $\mathcal{S}$ is set equal to 10 aircraft. Above this threshold, the system is in critical size and no new aircraft can leave its gate, as explained in Chapter 2. For each aircraft scheduled to depart on that day, we output the characteristic times of its departure process. More specifically, we store the following quantities, shown in Table 4.3.

$TTDP$	d_g	d_i	$d_{clearance}$	$t_{pushback}$	UTT	d_t	d_q	d_a
34	7	0	5	8	6	3	3	2

Table 4.3: *Output of the simulation for a random departure on Friday, October 19th, 2012.*

The first thing we can consider is the evolution of the TTDP with the time of day and compare it to the actual TTDP experienced by the aircraft on that day. Figure 4-5 below displays the computed TTDP and the actual TTDP for every aircraft.

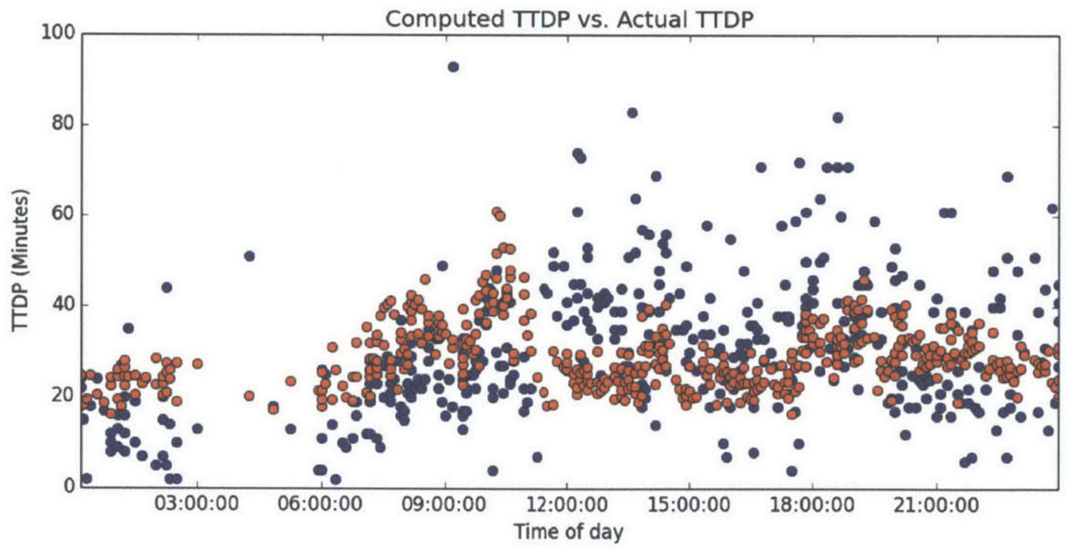

Figure 4-5: *For every departure time on the x-axis, the chart displays a red dot that corresponds to the computed TTDP and a blue dot corresponding to the actual TTDP experienced by the aircraft. There is for example an aircraft scheduled to leave its gate at 03:00. For this aircraft, our simulation predicts a takeoff time around 3:25, while its actual takeoff time was around 3:15.*

This chart allows us to make a few preliminary observations about traffic at the modeled airport. We first observe a high variability of the TTDPs observed in practice, which is much higher than in simulation. The simulation does not capture some of this variability, which is probably due to unusual events such as late boarding of passenger, mechanical problems, etc. Moreover, we can identify the main peak periods of that day, namely:

- **08:00 to 11:00**
- **13:00 to 15:00**
- **18:00 to 20:00**.

Our analysis will therefore focus on these time periods, since we can expect the demand to be too low for the sequencing to perform well outside of these peak hours.

Next, we can look at the breakdown of this TTDP for all aircraft that were processed on that day. The purpose of this is to identify the steps of the process that are causing delays throughout the day. We would also like to spot the periods of time during which aircraft are experiencing higher queuing delays, since this is the type of delay that we would like to minimize using sequencing. Such analysis would help us go a step further towards understanding the departure process at AIRPORT. If we focus for example on the set of aircraft scheduled to leave their gate between 7:00 and 13:00, we observe particularly high queuing delays around 10:00 (See Figure 4-6).
Another indicator of congestion at the modeled airport is the evolution of the gate delay with the time of day. Indeed, an aircraft experiences gate delay if and only if the system S is in critical size at the time the aircraft is supposed to leave its gate. We observe particularly high gate delays around 8:00 and 10:30, which confirms our initial observations about this period being one of the busiest of that day. However, high gate delays do not allow us to draw any conclusions about the congestion at the runway queue. The gate delay can be due to a high occupancy of the taxiway system, which does not necesarily translate into long queues at the runway, since the taxiing delays vary a great deal from an aircraft to another. Let us consider for example the gate delay around 8:00. The aircraft at the runway around this period are those that were scheduled to depart around 7:30. As seen on Figure 4-6, these flights do not show particularly high queuing delays, which means that the gate delays observed at 8:00 are mostly due to congestion on the taxiway system.

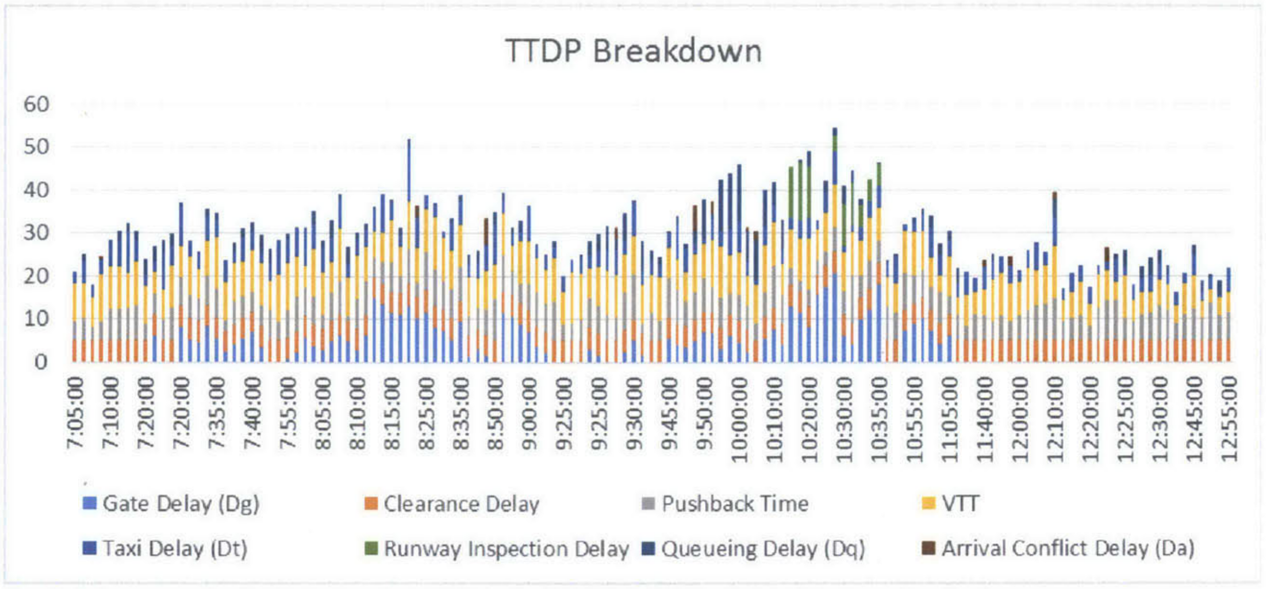

Figure 4-6: *Each bar corresponds to a departure. The height of the bar gives us the flight's TTDP, which is the sum of durations of all characteristic times of the departure process. Only flights scheduled to leave their gate between 7:00 and 13:00 are displayed here. Note that bars corresponding to flights with equal scheduled departure times are displayed next to each other, for comparison purposes.*

Analysis of the traffic at the runway queue

In this study, we are specifically interested in the intensity of congestion at the runway queue. To start looking into this particular part of the system, we can consider the hourly breakdown of the FCFS arrival times at the runway queue, which are stored by the simulation. The chart is shown in Figure 4-7. As explained earlier, this hourly breakdown is a better congestion indicator than the hourly breakdown of the STDs (Figure 4-1). We observe peaks from 8:00 to 9:00, 14:00 to 15:00 and 19:00 to 20:00. Note that these periods could have been intuitively anticipated, given the peak hours at the gate identified in Figure 4-1.

Another interesting way of visualizing runway queue congestion is by drawing a timeline displaying the runway occupancy according to the time of day. To do so, every time an aircraft X gets to the runway, we store whether or not the runway is busy. Note that we consider the runway to be busy if there is at least one aircraft that needs to take off ahead of aircraft X. On the visualization,

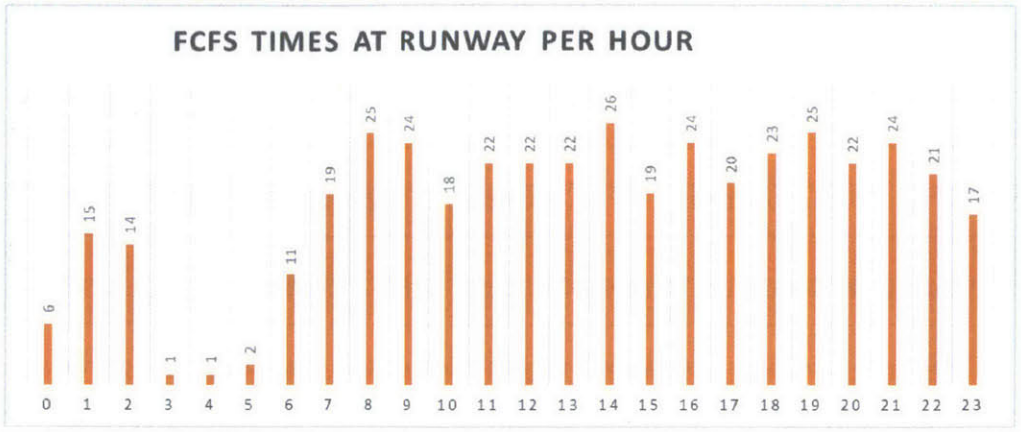

Figure 4-7: *Hourly breakdown of the FCFS arrival times at the runway queue, considering all departures from Friday, October* 19^{th}*, 2012.*

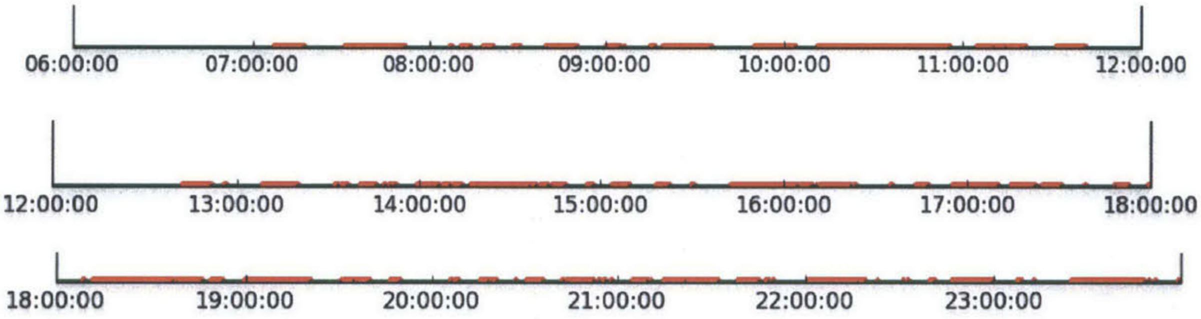

Figure 4-8: *Occupancy of the runway queue with time of day, shown for three time periods: 06:00-12:00, 12:00-18:00 and 18:00-00:00. Every data point represents a flight, with an x-value corresponding to its FCFS arrival time at the runway. The datapoint is colored in red or green depending on whether or not there is a queue when the aircraft gets to the runway.*

we display busy times with red dots, while the rest of the time is displayed using green dots. Figure 4-8 shows the results for the time periods 06:00-12:00, 12:00-18:00 and 18:00-00:00.

As expected from our analysis of the TTDP, we observe a high occupancy rate from 10:00 to 11:00 as well as from 14:00 to 15:00 and 18:00 to 19:00. During these periods, the runway queue seems continuously active in the sense that we almost systematically have at least one aircraft queuing.

More generally, we can also extract from this visualization the fraction of time the runway appears to be busy, as shown in Table 4.4. It is interesting to note that 40% of the time, the departures runway is busy in the sense defined above. Moreover, if we do not consider the 00:00 to 06:00 period, that same departure runway is busy 45% of the time. These relatively high numbers not only provide us with further insight regarding the traffic at the runway queue, but also allow us to anticipate positive results for our CPS techniques.

Time period	00-06	06-12	12-18	18-00	All day
Time with occupied Runway	12%	46%	40%	50%	39%

Table 4.4: *Occupancy of the runway in each of the four daily time periods.*

Now if we look into more details of the occupancy of the system $\mathcal{S}$, we can get additional quantitative insight regarding congestion at the runway queue. As explained in Chapter 2, the simulation keeps track of the occupancies of each part of the system (apron, taxiway system, runway queue). We are therefore able to plot the evolution of these quantities with the time of day, as shown in Figures 4-9, 4-10 and 4-11.

Figure 4-11 shows that the system is in critical size during the peak periods we identified above (08:00-11:00, 13:00-15:00, 18:00-21:00). When we look a level deeper, we observe that within these peak periods, congestion can be due to either a busy taxiway or a busy runway (or both). As expected from our interpretation of Figure 4-6, the gate delay experienced by flights scheduled around 08:00 is mostly due to a high occupancy of the taxiway system: during that period, we observe a taxi occupancy varying between 8 and 10 aircraft while the runway queue occupancy varies between 2 and 3 aircraft. However, we do observe high runway queue occupancies around 10:00 and 18:00, with a number of aircraft at the queue that varies between 4 and 6.
Other than the intensity of the traffic at AIRPORT, there is another factor that can impact the performance of our sequencing methods, namely the homogeneity of what we call the aircraft mix.

4.1.4 The aircraft mix

We call aircraft mix the proportion of aircraft in each weight class, considering all departures on a given day. In their paper, Balakrishnan-Chandran wrote about the importance of this aircraft mix. Their conclusion is that the more heterogeneous this mix is, the better the results will be when applying sequencing. This statement implied a need to look more closely at the aircraft mix in the specific case of our modeled airport.

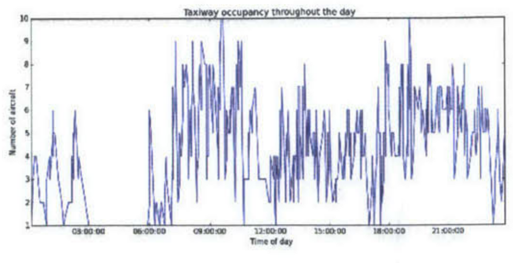

Figure 4-9: *Number of aircraft on the taxiway with time of day.*

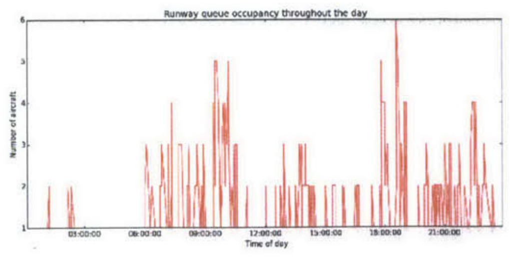

Figure 4-10: *Number of aircraft in the runway queue with time of day.*

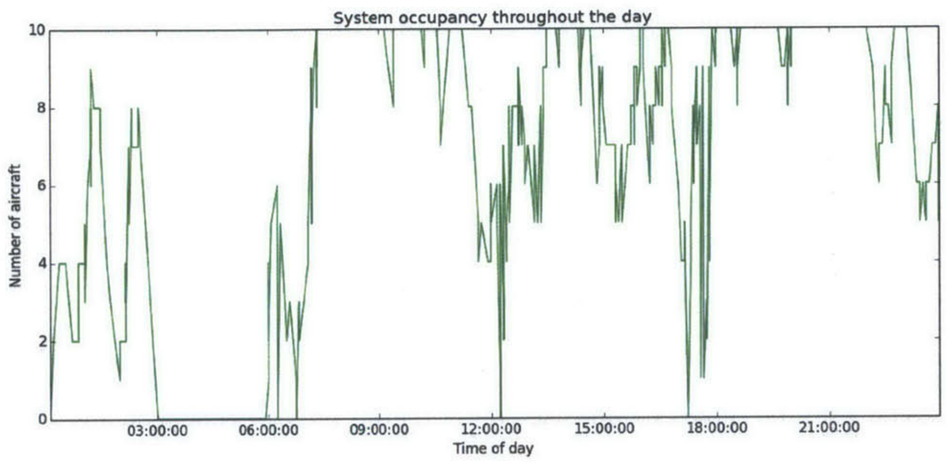

Figure 4-11: *Total number of active aircraft with time of day. By active aircraft, we refer to aircraft that are pushing back, taxiing or queuing for takeoff. Note that the occupancy of the system cannot exceed the critical size set for the simulation, which is 10 in our case.*

It turns out that most aircraft departing from AIRPORT belong to one of two classes: Medium (M) or Heavy (H). Table 4.5 presents the detailed aircraft mix at Airport.

Weight Class	L	M	H	SH
Count	1	250	166	13
Fraction	0%	58%	39%	3%

Table 4.5: *Aircraft mix at AIRPORT.*

This lack of heterogeneity in the aircraft mix will have a negative impact on the performance of the sequencing methods. In particular, since the overwhelming majority of aircraft are either $Medium$ or $Heavy$, we can take a look at the submatrix of the separation requirements, corresponding to the separations between these two weight classes.

$$T_{separation} = \begin{array}{c|cc} & H & M \\ \hline H & 90 & 90 \\ M & 120 & 90 \end{array} \tag{4.4}$$

Note that although the aircraft mix seems particularly homogeneous, this submatrix still clearly indicates that the queue could benefit from a switch between two aircraft from weight classes M and H. Suppose an aircraft X of type H is in first position in the queue, followed by aircraft Y of type M. The initial makespan is equal to 120 seconds. Switching these two aircraft would allow us to save 30 seconds in makespan, which is equivalent to decreasing makespan by 25%. In terms of delays, the initial average time spent in the queue is $\frac{0+120}{2} = 60$ seconds, while switching aircraft X and Y leads to an average delay of $\frac{0+90}{2} = 45$ seconds, which corresponds to a 25% improvement again. This trivial example helps emphasize the potential impact of sequencing on the airport's operations.

4.1.5 Methods of application

To conclude, the current demand at AIRPORT seems relatively low, with rates of aircraft at the runway that remain below 30 aircraft/hour. According to Balakrishnan & Chandran [4], these rates are actually not high enough for sequencing to perform at all in terms of makespan. However, we do observe some particularly congested periods at the runway queue, mainly around 10:00, 15:00 and 18:00. To deal with these particular congestion characteristics, we are going to apply our sequencing methods using two different approaches. Each of these approaches will be tested using both objective functions introduced in the previous section.
Moreover, we will also apply the same set of techniques to other demand profiles. In order to help AIRPORT anticipate its growth, we will artificially increase traffic to measure the impact of sequencing when the traffic is increased by 10, 20 or 30%.

Method 1: Apply to peak periods

One way of applying our sequencing algorithm would consist of focusing on the higher traffic periods. A first approach is thus to apply our CPS implementation to sequences of flights scheduled to arrive at the runway around the previously identified peak periods. More precisely, we will run our sequencing techniques on three different periods: 08:00-11:00, 13:00-15:00 and 18:00-21:00. During a run of the simulation we process the departures one by one. Every time an aircraft gets to the runway during one of these peak periods, it is included in the list of aircraft to be sequenced. Practically, AIRPORT can run our simulation ahead of time using their upcoming schedules of departures. This would allow them to anticipate peak periods for that day and know which departures will probably be included in the sequencing. Note that these peak periods probably do not change much from one day to the next, given the minimal variability of the schedules at the airport.

Method 2: Apply to a larger numbers of long sequences

A more dynamic approach consists of looking more closely at the evolution of the runway occupancy. While running the simulation, each time we observe 4 or more aircraft whose FCFS arrival times at the runway are close enough to one another, we will store this sequence and refer to it as a "long" sequence. We then apply CPS techniques independently to each of these long sequences. Using this method, our optimization essentially consists of consecutively applying sequencing methods to a significant number of series of 4 or more departures.

Note that Method 2 is in a sense "included" in the Method 1 defined above. In the first approach, we sequence a significant number of airplanes scheduled to depart during the peak periods identified previously. In other words, we look at the benefits of sequencing such number of aircraft. Method 2 will then only focus on a fraction of these aircraft. More specifically, we are zooming within the peak periods from Method 1 to find the most congested sequences. The main difference therefore consists of the way the savings are recorded. We will look into more details at the benefits of this approach.

4.1.6 Artificial increase of the demand

The previous sections justified the need for a methodology to artificially increase traffic at AIRPORT. This would allow us to anticipate the airport's growth which is expected to engender a 30% increase of traffic in the next twenty years. Moreover a more congested demand profile can lead to significant improvements for the results of the CPS techniques. This is why we will create demand profiles with traffic increases of 10%, 20% and 30%.

To increase traffic by 20% for example, we consider the initial schedule of departures, sorted by scheduled departure times. Then, we choose to simply move along this schedule and insert a new aircraft after every five departures. This new aircraft will have the following properties:

- Its scheduled departure time will be the average of the scheduled departure times of the two original aircraft between which it was inserted. If we insert an aircraft between aircraft 1 (STD = 10:34) and aircraft 2, (STD = 10:38), the new aircraft will be given a scheduled departure time of 10:36.

- About its weight class, we are going to try two different approaches.

  - The first one will be based on the assumption that the aircraft mix will remain unchanged in the next years. In that case, the new aircraft's weight class will be chosen randomly from the set $[L, M, H, SH]$, where the probability that the class is picked is directly linked to its current presence in the aircraft mix.

  - In a second approach, we will suppose that the aircraft mix will become more heterogeneous in the upcoming years. The more we increase traffic, the better we are able to balance traffic. To do so, we keep the initial homogeneous traffic and associate most of the new aircraft with the weight classes *Light* or *Super Heavy.* In that case, the probabilities of picking a class are chosen as displayed in Table 4.6. Note however that it is very unlikely to actually observe an increase in light traffic in the upcoming years. We will analyze the impact of hypothetical new aircraft mixes on the sequencing results.

Weight Class	L	M	H	SH
Unchanged Mix	0	0.58	0.39	0.03
Heterogeneous Mix (10%)	0.06	0.52	0.35	0.07
Heterogeneous Mix (20%)	0.08	0.48	0.33	0.11
Heterogeneous Mix (30%)	0.09	0.45	0.37	0.09

Table 4.6: *Probabilities of weight class assignment for artificially inserted aircraft.*

4.2 Computational Experiments

To display the results of our experiments, we will successively discuss the performance of Method 1 and Method 2. For each of these methods, we will present the relative improvements according to both objective functions and for different demand profiles. Each of the performance numbers introduced in this section will be the result of 100 simulation runs. Before we begin modeling, we need to properly define our improvement metrics for both objective functions.

Minimizing makespan

To measure the performance of the CPS method on the total makespan, we will use the following metric:

$$\frac{[x(i_n^0) - x(i_n^*)]}{[x(i_n^0) - x(i_1^0)]} \tag{4.5}$$

where we use the following notation:
- $(i_1^0, \ldots, i_n^0)$ is the set of nodes that corresponds to the FCFS sequence.
- $(i_1^*, \ldots, i_n^*)$ is the set of nodes that corresponds to the optimal sequence.
- $x(i_t)$ is the departure time of the last aircraft of node i_t from stage t.

Minimizing average delay

Similarly, we define another metric to measure the performance of the CPS method on the average delay:

$$\frac{\sum_{p=1..n}(t_p^0 - e(i_p^0)) - \sum_{p=1..n}(t_p^* - e(i_p^*))}{\sum_{p=1..n}(t_p^0 - e(i_p^0))} \tag{4.6}$$

where we use the following notation:

- $(i_1^0, \ldots, i_n^0)$ is the set of nodes that corresponds to the FCFS sequence.
- $(i_1^*, \ldots, i_n^*)$ is the set of nodes that corresponds to the optimal sequence.
- $e(i_p)$ is the FCFS arrival time at the runway of the last aircraft of node i_p from stage p.
- $(t_1^0, \ldots, t_n^0)$ is the set of takeoff times of each aircraft in the FCFS sequence.
- $(t_1^*, \ldots, t_n^*)$ is the set of takeoff times of each aircarft in the optimal sequence.

Note that each aircraft X is associated with exactly one position p_1 in the initial sequence and one position p_2 in the optimal sequence, which means that there exists a unique pair (p_1, p_2) such that $e(i_{p_1}^0) = e(i_{p_2}^*) = FCFS(X)$. Therefore, $\sum_{p=1..n} e(i_p^0) = \sum_{p=1..n} e(i_p^*)$. Given this observation, our metric actually becomes:

$$\frac{\sum_{p=1..n} t_p^0 - \sum_{p=1..n} t_p^*}{\sum_{p=1..n} (t_p^0 - e(i_p^0))} \tag{4.7}$$

4.2.1 Method 1

Current demand

We first focus on the performance of our CPS implementation on the current level of operations at AIRPORT. As observed in the previous section, the rate of aircraft at the runway queue does not reach the threshold of 30 aircraft per hour. In agreement with Balakrishnan & Chandran's results in [4], we notice that minimizing makespan does not yield any improvements for AIRPORT when we apply Method 1 to its current operations. This is why we will focus on the impact of sequencing on the average delay per aircraft. A summary of the results is displayed in Table 4.7 below.

First, we observe that the current demand profile leads to an average queuing delay per aircraft of approximately one minute during the peak hours. In other words, an aircraft spends on average a minute in the runway queue before taking off.

When every aircraft is allowed to move up or down one position from its original FCFS position at the runway queue (K=1), the sequencing techniques lead to a 4 to 5% decrease in average delay, which corresponds to approximately 2 to 3 seconds saved per aircraft.

For K=2, the aircraft is allowed to move up or down two positions from its FCFS position. Such

Time Period	# of Aircraft Sequenced	Average Delay per Aircraft	Average Delay Saved per Aircraft	
			K=1	K=2
8-11	67 (22/hr)	60 sec	3.0 sec (4.8%)	4.1 sec (6.7%)
13-15	49 (24/hr)	54 sec	2.3 sec (4.2%)	3.1 sec (5.4%)
18-21	69 (23/hr)	54 sec	2.6 sec (4.7%)	3.8 sec (6.7%)

Table 4.7: *Results obtained when applying Method 1 to AIRPORT's operations. For each demand profile, CPS methods were applied 100 times on each time period. We display here the average number of aircraft sequenced in a run, the average delay per aircraft experienced in the queue as well as the absolute and relative improvements for $K = 1$ and $K = 2$. Only the results relative to the average delay optimization are shown here because the makespan minimization does not lead to interesting results.*

relaxation of the fairness constraints intuitively leads to a higher improvement in average delay of 5 to 7% per aircraft. Therefore, the results are better when the aircraft is slightly less constrained. We will also observe that higher traffic can lead to higher gaps between the performances of $K = 1$ and $K = 2$.

Higher demand profiles

The idea is to observe whether or not increasing traffic significantly improves the performance of sequencing. We ran simulations for different demand profiles with traffic increases of 10%, 20% and 30%. The analysis of these three new profiles is relevant for the case of AIRPORT since it will allow us to anticipate the potential levels of future delay and assess the extent to which sequencing of departures could mitigate some of this delay. The results are displayed in Figure 4.8.

The first thing to observe is a steady increase of the average delay as a function of the traffic increase. This delay doubles from one to two minutes when the traffic is increased by 30%. The analysis shows that the more we increase traffic, the higher the average delay per aircraft and the more seconds we save per aircraft. Moreover, the results obtained for $K = 2$ are again better than

Traffic Increase	Time Period	# of Aircraft Sequenced	Average Delay per Aircraft	Average Delay Saved per Aircraft	
				K=1	K=2
+10%	8-11	75 (25/hr)	79 sec	4.5 sec (5.56%)	6.3 sec (7.8%)
	13-15	55 (27/hr)	73 sec	4.3 sec (5.6%)	5.7 sec (7.5%)
	18-21	77 (26/hr)	71 sec	4.1 sec (5.7%)	5.6 sec (8.0%)
+20%	8-11	82 (27/hr)	130 sec	8.9 sec (6.7%)	14.5 sec (11.1%)
	13-15	60 (30/hr)	115 sec	11.2 sec (9.5%)	17.0 sec (13.6%)
	18-21	84 (28/hr)	104 sec	9.1 sec (8.5%)	12.8 sec (11.9%)
+30%	8-11	87 (28/hr)	145 sec	11.7 sec (7.7%)	17.1 sec (11.9%)
	13-15	65 (33/hr)	155 sec	17.2 sec (10.8%)	27.0 sec (17.0%)
	18-21	90 (30/hr)	124 sec	11.3 sec (9.2%)	16.4 sec (12.7%)

Table 4.8: *Table of results when applying Method 1 to higher demand profiles at AIRPORT. For each demand profile, CPS methods were applied 100 times on each time period. We display here the average number of aircraft sequenced in a run, the average delay per aircraft experienced at the queue as well as the absolute and relative improvements for $K = 1$ and $K = 2$. Again and for similar reasons, only the results relative to the average delay optimization are shown here.*

the ones obtained for $K = 1$, and the gap between these two different sets of constraints increases with the average delay experienced by aircraft, as seen in Figures 4-12 and 4-13.

We ran another set of simulations for the same demand profiles, but this time assuming a more heterogeneous aircraft mix, as explained in the previous section. Note that our approach to artificially insert planes in the schedule implies that the higher the traffic, the more heterogeneous the mix becomes because we are mostly injecting *Light* and *Super Heavy* aircraft in the schedule. The results show that a more heterogeneous mix leads to even higher reductions of the average delay, although such a change in the mix slightly increases the initial average delay per aircraft. We compared the results above with the ones obtained from this new aircraft mix in Table 4.9. For comparison purposes, we only display the results for the time period 18:00-21:00; the other periods experience similar improvements.

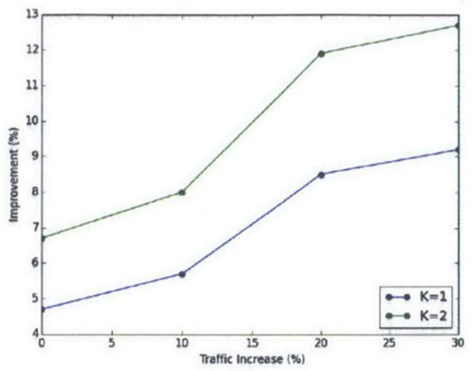

Figure 4-12: *Evolution of the relative improvements of sequencing with the traffic increase, for K=1 and K=2. We display the results obtained for the 18:00-21:00 period.*

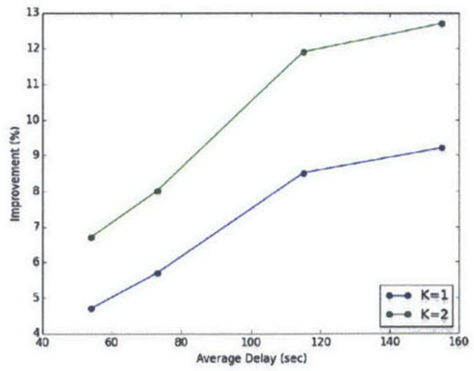

Figure 4-13: *Evolution of the relative improvements of sequencing with the average delay, for K=1 and K=2. We display the results obtained for the 18:00-21:00 period.*

Traffic Increase	Homogeneous			Heterogeneous		
	Average delay	K=1	K=2	Average delay	K=1	K=2
+10%	71 sec	5.7%	8.0%	79 sec	6.73%	9.65%
+20%	104 sec	8.5%	11.9%	127 sec	8.93%	13.8%
+30%	124 sec	9.2%	12.7%	140 sec	10.3%	14.3%

Table 4.9: *Comparison of the performance of sequencing for the heterogeneous and homogeneous aircraft mixes. Results are displayed for traffic increases of 10, 20 and 30%. Note that the higher the traffic, the more heterogeneous the mix.*

4.2.2 Method 2

The previous method did not allow any improvements regarding the makespan of the sequences. We will now focus on the second approach, which brings additional optimization opportunities to AIRPORT. The idea is to apply sequencing to every sequence of four aircraft or more whose FCFS arrival times at the runway queue are close to each other. More specifically, we consider two aircraft in the queue close to each other if there are less than 60 seconds between their corresponding FCFS times. Depending on the traffic intensity, we will see that the number of series which are sequenced per run varies and is particularly low when we analyze the current demand. However, our CPS implementation aiming at minimizing makespan will perform better on such "long series".

Current Demand

We first focus on applying Method 2 to the current operations at AIRPORT.

It is important to note that in this framework, the heteregoneous aircraft mix has a negative impact on the raw results. Let's prove this point through an example. Given the domination of the *Medium* weight class, some of the congested series we stored might be uniquely composed of aircraft of type M. For such series, sequencing does not bring any improvement. When we compute our results for Method 2, we take the average over 100 simulation runs. For each of these runs, the simulation stores the congested sequences, among which it is likely to find such trivial series because the average number of aircraft per stored series is no more than 4. The greater the number of such series in the simulation, the more the computed performance will be underestimating the actual value of the techniques.Therefore, we decided to compute the average improvements over series that include at least one *Heavy* aircraft. Although this measure will not be sufficient to entirely counteract the underestimation, it will still help reduce it. The results of the experiments are shown in Table 4.10.

Average Delay	Delay saved		Average makespan	Average makespan saved	
	K=1	K=2		K=1	K=2
115 sec	4.6s (4.0%)	7.9s (6.9%)	344s	11.1s (3.3%)	21s (5.9%)

Table 4.10: *Results obtained when applying Method 2 to the current operations at AIRPORT. The numbers are averages computed over 100 simulation runs. We display here the average delay and makespan relative to these series and the improvements that are made with both objective functions, for $K = 1$ and $K = 2$.*

We observe that for these long series of 4 aircraft or more, the average delay per aircraft is slightly under 2 minutes, which is approximately twice the average delay experienced during the two to three hour long peak periods experiencing the same demand. These high delays are intuitively expected, since we are looking at the most congested series within these peak periods. Note that with the current demand, we are only able to spot on average 3 congested long series per run. Moreover, we computed an average of 4.4 aircraft per sequence with the current demand, which means that only a small number of aircraft (13 per run on average) are involved in the optimization. It is therefore important to keep in mind that although the analysis may lead to high improvements, only a few aircraft are actually impacted by this method.

Regarding the average delay, queuing theory asserts that higher delays should lead to higher improvements in percentages. However, as explained earlier, the results in terms of average delay are lower than they should be due to a certain number of trivial series. The figures displayed should therefore not be considered as a contradiction, but should rather be seen as a limitation for Method 2 with such an homogeneous mix.

More importantly, the main result obtained from Method 2 consists of the average makespan saving obtained from sequencing. The average makespan of the long series studied in Method 2 is about 6 minutes. The CPS techniques allow us to save approximately 11 seconds when $K = 1$ and 21 seconds when $K = 2$. Note that these improvements are in complete agreement with the ones obtained by Balakrishnan and Chandran [4]. Indeed, for a slightly more heterogeneous mix than the one we deal with and for a rate of 60 aircraft per minute, they computed average improvements of 3% for $K = 1$ and 6% for $K = 2$ (See 4-14).

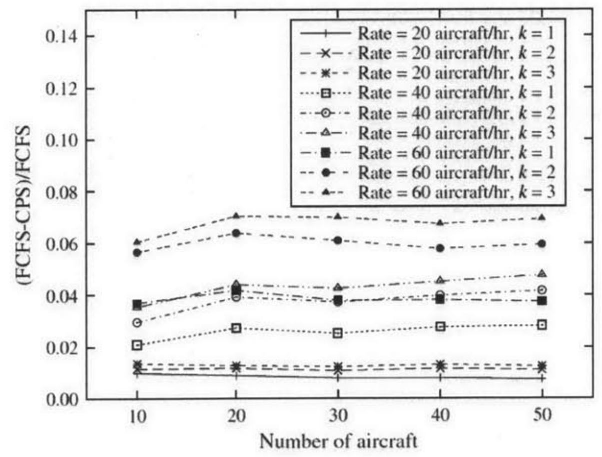

Figure 4-14: *Percentage improvement in makespan using CPS over FCFS for a slightly more heterogeneous mix than the one at AIRPORT. The chart was directly taken from Balakrishnan, Chandran [4]*

From now on, in the analysis of Method 2's benefits, we will only focus on the makespan minimization, since the average delay minimization does not add significant value to the results obtained in Method 1.

Therefore, if the airport is particularly interested in minimizing makespan, Method 2 is an interesting approach since it allows the airport to focus on the most congested sequences within the peak periods, for which sequencing aircraft can have a significant impact on the makespan. However, with the current demand levels, we are only able to find on average 3 of such long series per run, which means that the overall impact on the airport's operations is still low. We will now show that increasing traffic allows us to involve more aircraft in the optimization when applying Method 2.

Higher demand profiles

For higher demand profiles, we only display the results from the makespan optimization, since this is the main contribution from Method 2. We observe that the average length of a long series only increases slightly with traffic, from 4.4 per sequence with the current demand to 4.8 per sequence when traffic is increased by 30%. Thus, the average makespan does not change much and the savings per sequence are similar to the ones obtained with the current demand. The main difference between demand profiles consists of the number of long series found by the simulation. Whereas the optimization only involves on average 13 aircraft with the current demand, it can involve around 45 aircraft when the traffic is increased by 30%. In terms of fraction of the total number of departures, this is equivalent to a change from 3% to 8% of aircraft directly involved in the optimization.

Traffic Increase	Average number of sequences per run	Average length of a long sequence	Average makespan	Average makespan saved	
				K=1	K=2
+10%	5	4.5	353 sec	12.1 sec (3.2%)	23.8 sec (6.5%)
+20%	8	4.7	370 sec	14.7 sec (3.7%)	27.6 sec (7.1%)
+30%	9	4.8	370 sec	13.5 sec (3.5%)	24.0 sec (6.2%)

Table 4.11: *Complete table of results when applying Method 2 to AIRPORT's operations. For each demand profile, the CPS methods were applied 100 times for both objective functions. We display here the average number of series that are sequenced per run, the average size of these series, the average delay per aircraft computed over these series and the improvements that are made, for* $K = 1$ *and* $K = 2$*.*

4.2.3 Final remarks

We can summarize the results of these experiments as follows:

- **Method 1** does not help minimize the makespan during the peak periods. However, from an average delay of 60 seconds per aircraft, we manage to save 5% to 7% depending on how we choose the parameter K. With higher demand profiles, the average delay per aircraft doubles and the CPS implementation leads to even higher improvements that can go up to 17% per aircraft when the average delay is about 155 seconds per aircraft.

- **Method 2** involves many fewer aircraft but helps improve the makespan of a few series of 4 aircraft or more. This makespan can be improved by up to 5%. Although this zoom into the peak periods has a positive impact on the makespan, Method 2 does not add value to Method 1 in terms of average delay saved per aircraft.

- The aircraft mix has an impact on the performance of sequencing. We observed it in the case of Method 1, but we could also show similar results for the makespan in Method 2. In summary, the more heterogeneous the mix of aircraft at AIRPORT over the next years, the higher the initial average delay per aircraft is and the more benefits are obtained from sequencing aircraft at the departures runway. Moreover, for the specific case of Method 2, the homogeneous mix leads to underestimating performance, as seen in terms of average delay.

- To conclude, depending on its goals, AIRPORT could benefit from combining both methods presented in this chapter to take advantage of all given opportunities. While sequencing to minimize delays on peak periods, it is also possible to re-sequence the most congested series within these periods and minimize their makespan. The combination of both methods can therefore provide significant optimization opportunities to help improve AIRPORT's operations.

4.2.4 A word about the benefits of keeping aircraft at their gate

As part of the study, we also obtained a rough estimate of the taxiing delay that can be saved from limiting the number of aircraft on the system at any given time.

Using our hypothetical capacity C of 10 aircraft, we computed that approximately 900 minutes are spent by departing aircraft at their gate due to this measure. This number was obtained by summing the gate delays d_g of all aircraft scheduled to depart from AIRPORT during that day. In other words, if we were not applying this late release approach, 900 extra minutes would have been spent by aircraft on the taxiway with their engine turned on. If we approximate the cost of a minute spent on the taxiway to be \$60, this is equivalent to saving \$54000 on that day.

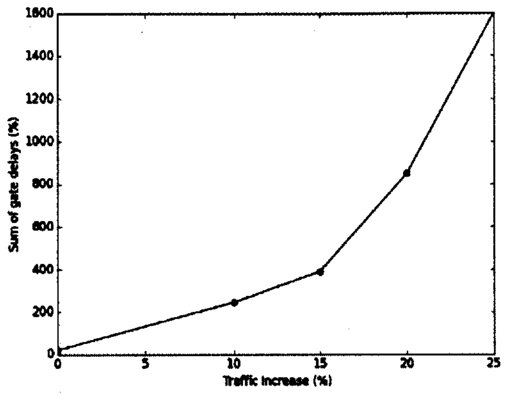

Figure 4-15: *Evolution of the gate delays with the traffic increase.*

We then decided to study the evolution of this total gate delay with the traffic levels at the modeled airport. In order to do so, we applied the same rough analysis for different demand profiles and for a same critical size C of 15 aircraft. The output is shown in Figure 4-15. We observe a non-linear increase of the gate delays at AIRPORT, which means that the higher the demand is, the more savings can be obtained from keeping aircraft at their gates. For example, when the demand is increased by 25%, the extra-time spent by aircraft at their gate is approximately 1600 minutes, which corresponds to saving \$96000 on that day.

Although we have not analyzed in detail the impact of this measure taken at the gate, the simulation therefore suggests that very substantial benefits can be obtained from a well determined limit C of the system occupancy. A more rigorous analysis would allow AIRPORT to find the optimal control strategy, in terms of optimal threshold C.

Chapter 5

Conclusion

5.1 Summary of the results

This thesis illustrates the benefits of sequencing departures at the runway queue for a specific airport which provided us with their operations data. We first designed a generic simulation which allows us to model the processing of aircraft from their gate to their takeoff. This simulation allows us to keep track of the characteristic times of the departure processes of all aircraft as well as the evolution of the system occupancy throughout the day. Using this tool, we managed to get a deep understanding of the airport's actual traffic characteristics. More specifically, we were able to identify the peak periods and provide quantitative estimates of the evolution of the system with the time of day. We then applied state of the art sequencing algorithms to explore the optimization opportunities at AIRPORT.

To do so, we developed two potential optimization frameworks to apply our Constrained Position Shifting (CPS) techniques. In the first approach, namely Method 1, we sequence aircraft on each of the peak periods of time identified through the analysis of the simulation output. With the current traffic levels, each of these peak periods can involve up to 70 aircraft, which spend on average 60 seconds in the runway queue. The optimization can help reduce this delay by up to 7%, which is equivalent to 4 seconds per aircraft. However, this method does not allow any improvements regarding the makespan of the sequence. The second approach, or Method 2, consists of applying sequencing independently to the most congested series from these peak periods. These series are made of four aircraft or more which arrive at the runway very close to one another. For these very congested series, we manage to save up to 20 seconds of a 340 second average makespan, which corresponds to a 6% improvement. However, with the current traffic levels, we only find three such long series per run of the simulation. Therefore, with the current demand at AIRPORT and its strongly homogeneous aircraft mix, the improvements obtained from sequencing turn out to be relatively small.

To anticipate the potentiel levels of future delay, we then artificially increased traffic and created new demand profiles. The delays saved by sequencing departures at the runway queue were shown to be an increasing function of the traffic levels at AIRPORT. Moreover, we tested our optimization models for two different methods for increasing traffic. In the first approach, we made sure to maintain the current aircraft mix while adding departures to the schedule. In the second one, we assumed a more heterogeneous aircraft mix by injecting more *Light* and *Super-heavy* aircraft in the schedule. We observed that the more heterogenous the mix is, the better we performed when sequencing aircraft in the queue. For a traffic level increased by 30% (which is a reasonable estimate of the airport's operations for 2030), the first framework can involve up to 90 aircraft per peak period that experience delays in the queue of more than 2 minutes. We are able to reduce this average delay by 30 seconds, which corresponds to a 17% improvement. For this same artificially inflated traffic of 30%, we now detect 9 long series as opposed to only 3, and we still save up to 6% per series on the makespan. The second framework therefore involves more than 40 aircraft, i.e. three times as much as it did with current traffic levels.

To conclude, given its fast expected growth, AIRPORT could significantly benefit from sequencing its departures in the queue. These techniques would help optimize their operations, particularly in terms of average delay per aircraft. Moreover, AIRPORT could largely benefit from a more heterogeneous aircraft mix, although such a mix would imply a higher average initial delay per aircraft. In other words, the sequencing would perform even better if, as an example, AIRPORT's operations start including more small aircraft in their fleet. However such change in the airport's fleet seems for now unrealistic.

Finally, we managed to get a rough estimate of the benefits obtained from the late release of aircraft from their gate. In case the system is in critical size, we control the flow of aircraft reaching the taxiway system by preventing new aircraft from pushing back from their gate. Using a reasonable critical size C for AIRPORT, we approximate the savings to more than $50000 per day with the current demand. Moreover, these benefits increase non-linearly when we increase the traffic levels. Although we did not look for the optimal threshold since it was not the focus of the thesis, we emphasized the positive impact which can be obtained from adopting such control strategies.

5.2 Further Research

A first area for future work would consist of improving the generic simulation designed to model the departure processes of all aircraft from an airport. More precisely, we did not validate rigorously the output of the simulation. We noticed for example in Figure 4-5 a high variability of the TTDP in real life that we do not capture using our simulation. It would be for example interesting to work on stochastically estimating this external delay, which is probably due to rare factors such as mechanical problems. Focusing more on such modeling points could add value to the simulation and lead to a more precise analysis of the traffic, which implies a better understanding of the congestion characteristics.

A rigorous study of control flow strategies would be another major area for future research. In other words, we could focus on finding the optimal threshold C above which the throughput does not benefit from sending additional aircraft on the taxiway system. If a rough estimate such as the one we presented in this work leads to such significant savings, we can then expect to obtain even higher benefits in terms of fuel and money from finding this optimal critical size. Indeed, choosing an arbitrary C like we did can lead to an overutilization of the apron: keeping aircraft at their gate for too long implies for example the unavailibility of the corresponding gates for arriving aircraft. For airports in the United States, it has been shown that the optimal threshold varies between 10 and 20 aircraft. For example, in the specific case of Boston Logan airport, Simaiakis [15] used his Pushback Rate Control (PRC) strategy to find an optimal maximum rate of 15 aircraft per minute, with a total available space of the system S estimated to be 30 aircraft.

When we applied sequencing to AIRPORT's operations, we did not take into account the interference with arrivals. In other words, the delays d_a due to conflicts with arrivals have an impact on the time spent by aircraft in the runway queue. Although adding such constraints could lead to better solutions in terms of average delay per aircraft or total makespan, it will not highly alter the optimal sequence since the number of arrivals on the departures runway at the modeled airport is relatively low. Dealing with this issue would basically consist of dynamically changing the costs associated to links of the CPS network, checking at every stage whether or not there will be interference between the current aircraft and an arriving one.

Finally, we chose to consider operations from a Friday at AIRPORT, which is the day of the week which is the most likely to experience congestion. It would be interesting to measure the impact of sequencing for other days of the week in order to find a yearly estimate of the savings that could be obtained from CPS techniques. Similarly, we chose to consider a day of October and looking at other months of the year may lead to different results. Although this is unlikely given the generally stable weather conditions at AIRPORT, there might be particular seasonality issues to take into account in the model, such as bird migration.

Bibliography

[1] Ioannis Anagnostakis, John-Paul Clarke, Dietmar B-ograve, hme, Uwe V-ograve, and lcker. Runway operations planning and control: Sequencing and scheduling. *Journal of Aircraft*, 38(6):988–996, 2001.

[2] Stephen Atkins and Christopher Brinton. Concept description and development plan for the surface management system. *Journal of Air Traffic Control*, 2002.

[3] Hamsa Balakrishnan and Bala Chandran. Efficient and equitable departure scheduling in real-time: New approaches to old problems. In *USA/Europe Air Traffic Management R&D Seminar*, Barcelona, Spain, June 2007.

[4] Hamsa Balakrishnan and Bala Chandran. Algorithms for scheduling runway operations under constrained position shifting. *Operations Research*, 58(6):1650–1665, 2010.

[5] John E Beasley, Mohan Krishnamoorthy, Yazid M Sharaiha, and D Abramson. Scheduling aircraft landingsthe static case. *Transportation science*, 34(2):180–197, 2000.

[6] Dietmar Böhme. Tactical departure management with the eurocontrol/dlr dman. In *6th USA/Europe Air Traffic Management Research and Development Seminar, Baltimore, MD*, 2005.

[7] Francis R Carr. *Robust decision-support tools for airport surface traffic.* PhD thesis, Massachusetts Institute of Technology, 2003.

[8] Roger Dear. *The Dynamic Scheduling of Aircraft in the Near Terminal Area.* PhD thesis, MIT, 1976.

[9] Eric R Feron, R John Hansman, Amadeo R Odoni, Ruben B Cots, Bertrand Delcaire, William D Hall, Husni R Idris, Alp Muharremoglu, and N Pujet. The departure planner: A conceptual discussion. 1997.

[10] Husni R Idris, Bertrand Delcaire, Ioannis Anagnostakis, William D Hall, Nicolas Pujet, Eric Feron, R John Hansman, John-Paul Clarke, and Amedeo Odoni. Identification of flow constraint and control points in departure operations at airport systems. In *AIAA Guidance, Navigation and Control Conference*, 1998.

[11] Hanbong Lee. *Tradeoff evaluation of scheduling algorithms for terminal-area air traffic control.* PhD thesis, Massachusetts Institute of Technology, 2008.

[12] Frank Neuman and Heinz Erzberger. *Analysis of delay reducing and fuel saving sequencing and spacing algorithms for arrival traffic.* Citeseer, 1991.

[13] Harilaos N Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research*, 28(6):1347–1359, 1980.

[14] I Simaiakis, M Sandberg, H Balakrishnan, and RJ Hansman. Design, testing and evaluation of a pushback rate control strategy. In *International Conference on Research in Air Transportation*, 2012.

[15] Ioannis Simaiakis. *Analysis, modeling and control of the airport departure process.* PhD thesis, Massachusetts Institute of Technology, 2013.