Scheduling Under Uncertainty: Applications to Aviation, Healthcare and Aerospace

by

Jeremy Castaing

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2017

Doctoral Committee:

Associate Professor Amy E.M. Cohn, Chair
Associate Professor James W. Cutler
Associate Professor Brian T. Denton
Associate Professor Marina A. Epelman

ACKNOWLEDGMENTS

I first would like to thank my advisor and mentor Prof. Amy Cohn. I feel extremely privileged to have had the chance to benefit from her guidance for the past five years as well as the opportunity to join the Center for Healthcare and Patient Safety (CHEPS). There I met wonderful fellow students and researchers, many of whom I now call my friends. I especially want to thank, in no particular order, Spyros, Sarah, Matt, Gene, Abhi, Karmel, Brian and Victor for their enthusiasm and help.

It is also very important that I mention the Seth Bonder's foundation. Thank you for your generous contribution and trusting me to conduct successful and meaningful research.

Finally, I would like to thank my family, Francoise, Nicolas and Victor, and my wife, Steph, for their constant support. This accomplishment is also theirs since their love and encouragement were essential to my success.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Scheduling Under Uncertainty: Applications to Aviation, Healthcare and Aerospace

by

Jeremy Castaing

Chair: Amy E.M. Cohn

When scheduling a project or a mission, it is often challenging to know in advance the exact duration of each task or which resource will be available. Processing times and resource availability are often subject to variability and may only be known at the last minute. Ignoring this uncertainty when planning a project can lead to adverse outcomes such as additional costs, missed deadlines or failed tasks. Conversely, modeling uncertainty in the scheduling decision process has potential to create more robust schedules that will mitigate these negative outcomes. However, the complexity of deterministic scheduling problems is further increased in their stochastic counterpart and many challenges arise when attempting to model and solve scheduling problems subject to uncertainty.

In this dissertation we specifically study four scheduling problems arising from the transportation and the healthcare industries. In each of these four examples, we consider the limitations of deterministic approaches and the impact of uncertainty on the solution's structures and costs. Two problems come from the airline industry. We first create a model to generate flights gate assignments so as to reduce the probability of conflict between planes and mitigate delays. Then we develop a simulation tool to analyze delay recovery strategies under uncertainty. A third project deals with scheduling patient appointment times for chemotherapy infusion under uncertainty of their treatment time. The last area of

application that we consider is satellite mission scheduling. We develop several models to solve the download planning problem for a single satellite while considering uncertainty in the availability of multiple receiving ground stations distributed across Earth.

CHAPTER 1

Introduction

When scheduling a project or a mission, it is challenging to know in advance the exact duration of each task or which resource will be available. Processing times and resource availability are often subject to variability and may only be known at the last minute. Ignoring this uncertainty when planning a project can lead to adverse outcomes such as additional costs, missed deadlines or failed tasks. Conversely, modeling uncertainty in the scheduling decision process has potential to create more robust schedules that will mitigate these negative outcomes. However, the complexity of deterministic scheduling problems is further increased in their stochastic counterpart and many challenges arise when attempting to model and solve scheduling problems subject to uncertainty.

Scheduling is a fundamental domain of the operations research field, putting together many different methods such as linear programming, evolutionary algorithms and simulation-based optimization to solve a wide range of problems sharing some common elements and features such as job processing time, deadlines, and precedence constraints between tasks. In this dissertation we specifically study four scheduling problems arising from the transportation and the healthcare industries. In each of these four examples, we consider the limitations of deterministic approaches and the impact of uncertainty on the solutions structures and costs.

Consider, for example, the problem of assigning patients appointment times for surgery. A natural first step is to schedule each patient based on the average or expected length of his procedure and allow that much time before the next patient appointment time. This so-called *average problem* has the advantage of being simple to model and does not require a lot of information besides an estimate of the surgery duration for each patient. However, we know that surgeries tend to vary in length in unpredictable ways due to complications or patient adverse reactions. By neglecting this uncertainty we ignore the risk of a procedure running over its allocated time slot, causing delays and waiting times for subsequent patients as well as, ultimately, overtime or idle time for the staff which in turn leads to extra

costs or wasted resources. A procedure could also run under, which wastes resources and delays patients who might have been scheduled at an earlier date. This simple example illustrates the fact that solving the average problem can lead to overlook key dynamics that impacts the constraints and objective of our problem. We discuss a way to take uncertainty into account in a similar patient scheduling problem in Chapter 3.

We now consider the main differences between a deterministic problem and its stochastic counterpart:

- The deterministic model ignores variability and is typically based on average values or some pre-determined percentiles. This assumption potentially leads to several issues when actually carrying out the plan.
  - The resulting objective value might be very different from what we anticipated. In some cases, the variability might smooth itself out and no significant changes happen to the final objective (often the case with linear systems, sales under uncertain demand [Petkov and Maranas, 1997], bin-packing type of problems [Coffman et al., 1980]). However uncertainty might start compounding in some dynamic systems leading to important changes of the objective (wait times in queuing system [Burke, 1956], delays in transportation networks [Fleurquin et al., 2013]).
  - Or the solution might become infeasible. In linear programming, sensitivity analysis shows that a small change of a parameter's value might cause the solution to violate a constraint [Bertsimas and Tsitsiklis, 1997]. This is especially true for equality type constraints or binding constraints of an optimal solution. In the event of a violation occurring, the decision maker needs to modify the original plan to satisfy the constraint, which is known as *recourse* in the literature [Birge and Louveaux, 2011a].
- The stochastic version of the problem incorporates uncertainty. By considering different possible realizations for each uncertain parameter we can often decompose the scheduling problem into a planning and a recourse phase, where recourse is defined as actions taken once the uncertainty is realized. We discuss the concept of recourse more in depth in chapters 3 and 4. Each phase typically has its own objective or costs and the model aims to minimize a linear combination of the planning and recourse costs. The costs associated with the recourse phase are usually high, representing the fact that dynamically changing your plan on the fly is typically expensive (e.g., staff overtime, accommodating passengers and hotel fees after a flight cancellation...).

This dissertation contains four main chapters, each focusing on a real-world scheduling problem with significant sources of uncertainty. The examples considered come from applications of scheduling to aviation, healthcare or aerospace.

Chapter 2 considers the problem of assigning flights to gates in order to reduce the impact of *gate blockage*. A gate blockage happens when a flight arrives at its scheduled gate but has to wait because the preceding aircraft is still occupying that gate. These conflicts occur as a result of variability in the system: flights leaving/arriving early or late from/to their destination gate. A template for gate assignments is typically first created solely based on scheduled times and follows the first in first out paradigm, ignoring uncertainty in departure and arrival times. The assignment can then be slightly adjusted to accommodate additional constraints: international flights have to go to specific gates or terminals, flights part of connection banks would go to adjacent or close gates etc. However, we recognize that the system variability is, to some extent, predictable and that incorporating that knowledge in the gate assignment model could lead to more robust schedules. Our approach is to study the difference between scheduled and actual operations in a large aviation database, compute an estimate of the probability of a gate blockage happening between each given pair of flights and solve a deterministic network flow model based on these parameters to generate a gate assignment. This project is a good example of a case where current airline operations suffer from variability in the system and how better modeling uncertainty can help reducing delays.

One of the shortcomings of the approach developed in Chapter 2 is that we are not considering recourse when things go wrong: what to do when a blockage occurs? In Chapter 3 we explore the concept of recourse in a healthcare related project, we assign patient appointment times at an outpatient chemotherapy infusion center, under uncertainty of treatment times. In addition to appointment times, we also consider resources such as nurses and infusion chair, as well as recourse in the case perturbations occur in the system: waiting, idle time and chair assignment are all dynamically adjusted depending on the actual realization of treatment times. This naturally leads to a two-stage model approach where appointment times are set in the first stage and chair assignment as well as delays are evaluated in the second stage.

The previous model considered two stages (appointment times and then chair assignments), however in some cases, this is not possible. Indeed, dynamic systems might require sequential decisions to be made in more than two stages. These multi-stage problems are often much harder to solve. In Chapter 4 we study an example coming from the aerospace industry: planning operations for a satellite mission. We specifically look at scheduling downloads of a satellite that collects data while orbiting Earth. This satellite has limited

space available in its memory and therefore needs to regularly transmit its stored data to stations on the ground. We aim to enhance a deterministic model by adding uncertainty in resource availability, modeling the fact that ground stations might not be unable to receive data at a certain time, because of a technical failure or a conflict with another satellite's communication. We model this uncertainty using binary random variables which represent the availability of each ground station. We study different modeling techniques to create download policies and see how they impact the ability to solve the problem and what the corresponding solutions represent in the context of this satellite mission planning.

In Chapter 5, we study another problem from the airline industry that deals with recovery under uncertainty. In Chapter 2, we introduced the notion of gate blockage as one example of sources of delays. However, there are many other causes of delays, including bad weather, airports congestion, and mechanical problems. Airline companies put a great deal of effort into developing strategies to cope with these delays, mitigating their propagation in the network and ultimately returning to a steady, normal state. This process, called recovery, typically uses mechanisms such as flight cancellation and aircraft swap to combat propagating delays based on the current state of the system. In this project, our goal is to develop a simulation tool to model and compare different recovery strategies and show that there is value in considering uncertainty in future operations when making these recovery decisions.

Throughout this dissertation we are making a number of contributions. First, we model and solve several real-world problems from the industry and propose methods to decrease operating costs, increase customers/patients satisfaction or optimize outcomes across the planning horizon. Second, we advance the literature on scheduling under uncertainty by developing novel algorithms and methodologies.

CHAPTER 2

Reducing airport gate blockage in passenger aviation: Models and analysis

2.1 Introduction

Commercial flights are typically assigned to an arrival gate at their destination *station* (airport) well in advance of their actual departure. Although the gate is scheduled to be available when the flight arrives, this is not always the case in practice. Due to variability in departure and flight times, the arriving flight might arrive early, the previous flight departing from the gate might depart late, or both.

When a flight arrives at its scheduled gate but has to wait because the preceding aircraft is still occupying that gate, we refer to this as *gate blockage*. Gate blockage can have many negative impacts, including passenger delays, missed connections, and increased fuel burn. Our research is focused on incorporating the inherent stochasticity of the system into the planning process to reduce the prevalence and impact of gate blockage.

We begin by conducting an analysis of historical data from a major U.S. carrier, examining the frequency and extent of gate blockage in practice. We demonstrate how different gate assignments can lead to different degrees of gate blockage by incorporating information about the variability in flight arrival and departure times. To leverage this, we develop mixed integer programming *(MIP)*-based models to optimize the expected outcome of the gate assignment under stochastic conditions. We then conduct empirical analyses using real-world data to show both the computational tractability of our proposed approach and the potential benefits to be achieved through incorporating uncertainty in the planning process.

Our contributions are in advancing the literature on airline gate planning by assessing the impact of stochasticity on gate blockage and in proposing MIP-based approaches to reduce this impact. We conduct an historical analysis to highlight the frequency of gate

blockage. We then present two optimization based approaches to reduce this blockage by incorporating system stochasticity, using a unique network design in which gates, rather than aircraft, flow through the system. The first approach assumes all aircraft are compatible with all gates; this is the model that motivated our research and on which most of our computational results are based. We also briefly discuss a second approach that consider the general case where not all aircrafts are compatible with all gates. In both cases, we approximate objective coefficients to represent the probability and severity of gate blockages as a function of gate turns. We provide computational experiments based on real-world data from a major U.S. carrier to show the tractability and effectiveness of our proposed approach.

The remainder of the paper is organized as follows: Section 2 describes our approach, as well as a survey of existing literature on the gate assignment problem, robust scheduling applied to passenger aviation, and other topics relevant to our study. In Section 3 we present an historical analysis of the frequency and patterns of gate blockage. In Section 4 we present two models for solving the gate assignment problem so as to minimize the potential for gate disruption. Section 5 describes the methods used to generate the objective coefficients of these two models and Section 6 is dedicated to various computational experiments. Section 7 presents our conclusions and some ideas for future research.

2.2 Motivation, Problem Statement and Literature Review

2.2.1 Motivation

In the U.S., the majority of commercial flights depart from and arrive at physical gates at the corresponding terminal. [This is in contrast to Europe, where flights frequently arrive at *hard stands* on the tarmac, from which passengers are bussed to the terminal.] Because two flights cannot occupy the same gate at the same time, a schedule is built to ensure available gates for all flights throughout the day. This gate assignment defines a sequence of *gate turns* for each gate.

A gate turn corresponds to an aircraft that leaves a gate (a departing flight) followed by an aircraft subsequently arriving at the same gate. In between, a minimum *gate buffer* or *sit time* (e.g., five minutes) must be allotted to allow the first aircraft to clear the area before the second aircraft can reach the gate.

Note that gate turns represent an outbound flight followed by an inbound flight, whereas an *aircraft turn* is an inbound flight followed by an outbound flight that use the same aircraft. Figure 2.1 illustrates the assignment of three aircraft turns, i.e., three pairs of inbound

and outbound flights $([I_k, O_k])_{1 \leq k \leq 3}$, to a single gate. This assignment corresponds to two resulting gate turns.The gate is occupied when the timeline is bold.

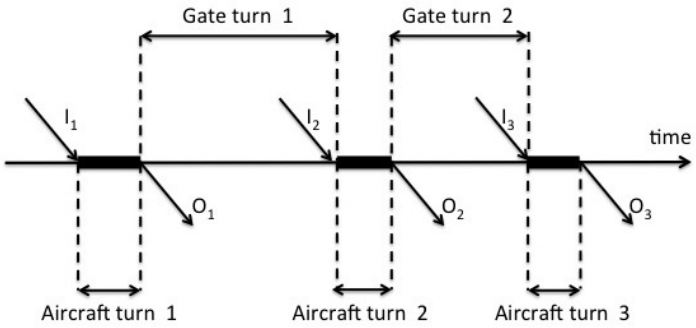


Figure 2.1: Gate schedule with three aircraft turns and two gate turns

In this paper, we focus on assigning aircraft turns (which have been pre-determined in an earlier stage of the planning process) to gates. Our primary objective is to minimize the potential for gate blockage and associated disruptions. Note that other metrics may be of concern as well when assigning flights to gates, such as distance for connecting passengers or effective utilization of ground resources, and we briefly touch on these extensions in our conclusions.

Flight departures are often delayed, for a number of reasons. These include mechanical or weather related problems, ground delay programs, and delays in passenger boarding. In addition, earlier delays in the system can propagate to delay downstream flights. Likewise, there are many reasons why a flight may be early in arriving. There is always buffer built into the system to accommodate variability in departure time, outbound taxi time, flight time, and inbound taxi time. When this buffer is not needed, flights may arrive early.

Gate blockages can have many negative consequences. At a minimum, they inconvenience and frustrate the passengers on board the blocked aircraft. Blockages can also lead to passenger delays (which can lead to missed connections) and propagation of crew and aircraft delays. Furthermore, gate blockages lead to excess fuel burn (with both financial and environmental impacts), increased crew costs, and disruption to the planned utilization of ground resources. Finally, the presence of excess aircraft on the tarmac can lead to increased congestion, which in turn can not only cause more ground delays, but has implications for passenger safety and aircraft damage as well.

Example

Consider two outbound flights O_1 and O_2 as well as two inbound flights I_1 and I_2. Suppose that scheduled departure times are 8:30 for O_1 and 8:40 for O_2 and that scheduled arrival times are 8:50 for I_1 and 9:00 for I_2.

One possible gate assignment is to pair O_1 and I_1 in one gate and O_2 and I_2 in another gate. This assignment is reasonable since it allow a 20 minute gate turn time for both outbound-inbound pairs. Suppose, however, that based on historical data, we know that inbound flight I_1 often arrives late whereas inbound flight I_2 frequently arrives early. We further assume that outbound flights O_1 and O_2 leaves constistently on time. In that case, a more robust gate assignment would be to pair O_1 and I_2 in one gate and O_2 and I_1 in another gate.

2.2.2 Problem Statement

Given a set of aircraft turns (each defined by either an inbound flight followed by an outbound flight using the same aircraft, an outbound flight that is the day's first use of a given aircraft, or an inbound flight that is the day's last use of a given aircraft) and a set of gates for a single station, assign each aircraft turn to a gate, so as to maximize robustness, under the constraint that a gate can handle at most one aircraft at any given time.

We consider four different objective functions representing four different measures of robustness:

- Expected Number of Blockages (objective P): The first objective we consider is to minimize the *expected number of gate blockages*. To compute this, we have determined (based on historical averages) the probability of experiencing gate blockage associated with each possible gate turns. The objective is to then minimize the sum of these probabilities across all gate turns assigned in the optimal solution.

- Expected Total Time of Blockage (objective X): One limitation of the first objective metric is that it ignores the duration of the blockages, weighting a short blockage no differently than a long blockage. Therefore, in our second metric, we minimize the *expected total time of blockage*.

- Expected Connecting Passenger Blockage Minute (objective C): The third objective is motivated by the fact that the impact of blockage time is not necessarily linear. A twenty minute gate blockage imposed on a terminating passenger might have far less impact than a ten minute gate blockage imposed on a passenger with a very tight connection. As a first approximation to capture this, in our third measure of robustness we focus specifically on the gate blockage imposed on connecting passengers. Specifically, for each outbound/inbound flight pair forming a potential gate turn, we take the expected length of blockage for the flight pair and weight this by the average number of connecting passengers on the inbound flight.

- Worst Case Expected Blockage (objective W): Finally, as our fourth measure of robustness, we minimize the *worst case expected blockage* – the longest expected blockage that any of the assigned turns would experience. This effectively sets a cap on the longest gate blockage, thereby recognizing the non-linear impact of gate blockages and striving to keep all blockages at a low level.

2.2.3 Literature Review

Despite the potential benefit to be gained by making gate planning decisions more robust, there has been limited attention paid in the literature to this problem. In one example, [Kim and Feron, 2011], robust gating and propagation of delays for a multi-station network are considered. Another example, [Lim and Wang, 2005], takes a stochastic programming approach, focusing on a single station with multiple distributions for incoming flight delays. Perhaps closest to our research is the example of [Li, 2008], which minimizes the expected number of gate blockages, given a formula to predict the distribution of gate blockage between a given outbound/inbound flight pair.

There has been a bit more research on other aspects of gating in passenger aviation. For example, [Ding et al., 2005] considers gate planning from the perspective of minimizing passenger walking distance and connection times. [Cheng et al., 2012] compares the performance of three meta- heuristic algorithms for solving the gate assignment problem, focusing on resource utilization and passenger satisfaction, and [Genç et al., 2012] designs another heuristic using stochastic optimization to solve the gate assignment problem. A stochastic optimization approach to the problem has also been proposed by

[Şeker and Noyan, 2012].

Other areas of robust planning in passenger aviation have been studied more extensively, including flight scheduling [Burke et al., 2010], [Ahmadbeygi et al., 2010], [Lapp et al., 2008] and [Teodorovic and Stojkovic, 1995]; scheduling and routing [Lan et al., 2006]; fleet assignment [Rosenberger et al., 2004]; aircraft routing and maintenance planning [Borndörfer et al., 2010], [Lapp and Cohn, 2012] and [Lapp, 2012] as well as crew scheduling [Klabjan et al., 2001], [Schaefer et al., 2005] and [Yen and Birge, 2006].

Closely related to the issue of robust planning (which attempts to prevent disruptions before they occur, or to mitigate the impact of disruptions) is the issue of recovery and passenger re-accommodation (which addresses disruptions after they have occurred, to minimize their impact). There is extensive literature in this area as well. Examples include [Thengvall et al., 2000], [Eggenberg et al., 2010] and [Yan and Yang, 1996], who consider the recovery of aircraft schedules during periods of disruption; [Beatty et al., 1999] and [AhmadBeygi et al., 2008], who look at the relationship between airline plans and the potential for delay propagation; [Abdelghany et al., 2004] and [Lettovskỳ et al., 2000], who consider crew recovery during irregular operations; [Barnhart et al., 2003] and [Cohn and Lapp, 2010] consider passenger recovery; [Sriram and Haghani, 2003], who consider maintenance schedule re-planning; and [Kohl et al., 2007] and [Filar et al., 2001], who provide surveys of airline disruption management.

Finally, we conclude by suggesting a number of useful survey papers for the novice reader unfamiliar with passenger airline planning and operations: [Barnhart et al., 2003], [Cohn and Lapp, 2010], [Barnhart and Talluri, 1997], and [Gopalan and Talluri, 1998b].

2.3 Case Study for Historical Data Analysis

To demonstrate the importance of our proposed approach to incorporating stochasticity in gate planning, we begin with a historical analysis of the frequency and patterns of gate blockage today. In particular, we want to answer the following questions:

- How often does gate blockage occur?
- When gates are blocked, how long is the blockage?
- Does it vary by station?

- Does it vary by time period?

The answers to these questions will serve as a motivation for the rest of the paper, where we seek to reduce gate blockage by incorporating stochasticity in the planning process.

2.3.1 Methodology

Our analysis focuses on the domestic operations of a single, major U.S. carrier. We summarize the data below, which has been loosely disguised per the request of the carrier. In our analysis, we consider four time periods corresponding to four different flight schedules (note that the length of these periods varies):

- Period 1 - January 2009 - 22 days,
- Period 2 - January 2010 - 31 days,
- Period 3 - December 2010 to February 2011 - 90 days,
- Period 4 - December 2011 to February 2012 - 91 days.

For each of these periods, we evaluate a panel of 15 of the largest airports in the carrier's network. These stations are described in Table 2.1.

The carrier also provided us with information for each flight including scheduled and actual departure from origin and arrival to destination, as well as the origin and destination gates that were used.

We do *not* have access, however, to explicit gate blockage because it is not readily available in the carrier database. Thus, to conduct our analysis, we have to reverse-engineer the available data to estimate gate blockages.

To do so, we first note that an upper bound on the gate blockage associated with a given inbound flight can be found by substracting the time that the flight landed (*wheels on*) from the time that it reached the gate. Some of this time, however, will be necessary taxi time (i.e., the time that it takes to physically travel from the runway to the gate) and some of it may be delays in taxi that are caused by something other than a blocked gate (e.g., congestion on the tarmac). We have therefore chosen to approximate gate blockage in the following way:

Station	Average Number of Flights per Day	Average Flights per Gate per Day
1	200-210	8.9
2	180-190	6.1
3	160-170	5.6
4	140-150	7.4
5	130-140	8.0
6	110-120	5.3
7	100-110	8.0
8	90-100	6.5
9	80-90	7.4
10	60-70	6.8
11	50-60	7.0
12	40-50	6.7
13	30-40	3.1
14	10-20	3.9
15	10-20	1.6

Table 2.1: Main characteristics of the airports in our panel

First, for each station during each time period, we calculate a nominal taxi time that we define as the median of all taxi times. We choose the median as a way of disregarding longer taxi times that are caused by external factors such as airport congestion, disruptive weather conditions, and in fact the presence of gate blockages themselves. At first glance, it seems that we should use the smallest observed taxi time as nominal taxi time. However, we have chosen median instead of using a lower percentile in recognition of the fact that (a) different flights are going to different gates, which will introduce some variability into the nominal taxi time and (b) flights are also coming in from different runways and under different airport configurations.

Figure 2.2 presents the distribution of taxi times for over 10000 flights arriving at a single station. As expected, most of the flights have a taxi time of between 2 and 4 minutes — 3 minutes being the median — and the distribution has a tail containing a few flights with much longer taxi times, due to the reasons mentioned above.

Next, for each flight on each day in the time period, we take the wheels on time and add to this the nominal taxi time. This is our estimated gate arrival time.

We then consider the flight departing from the same gate prior to the arrival and add to its actual departure time the minimum gate buffer time to determine when the gate became available.

If the estimated gate arrival time is earlier than the gate available time, then we record

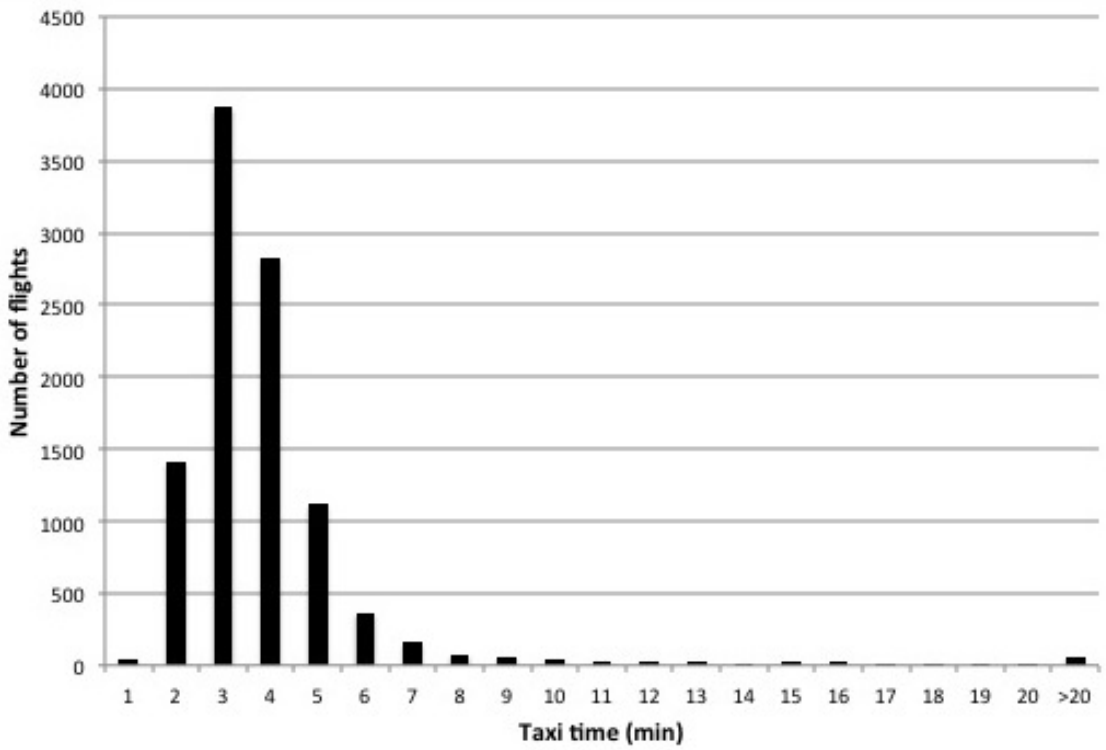


Figure 2.2: Distribution of taxi time at one station

the difference as a gate blockage.

Note that we do not consider the *actual* arrival time of the flight to the gate — i.e., if the arrival time of the flight to the gate is later than the gate available time, we do not count that interval in the blockage. This is because we assume that other factors must have caused this delay.

Figure 2.3 presents two possible outcomes of a gate turn, the first one without gate blockage and the second one with gate blockage.

2.3.2 Analysis

We study the frequency and the duration of all gate blockages (both those caused by late outbound departures and those caused by early inbound arrivals) for the four periods and fifteen airports previously described. We begin with Table 2.2, a detailed summary of the results obtained for period 3 which is the period we used to run our computational tests (see Section 2.6). The three last lines are the total across all stations for each column as well as the percentage of flights and the percentage of blocked flights in each bucket.

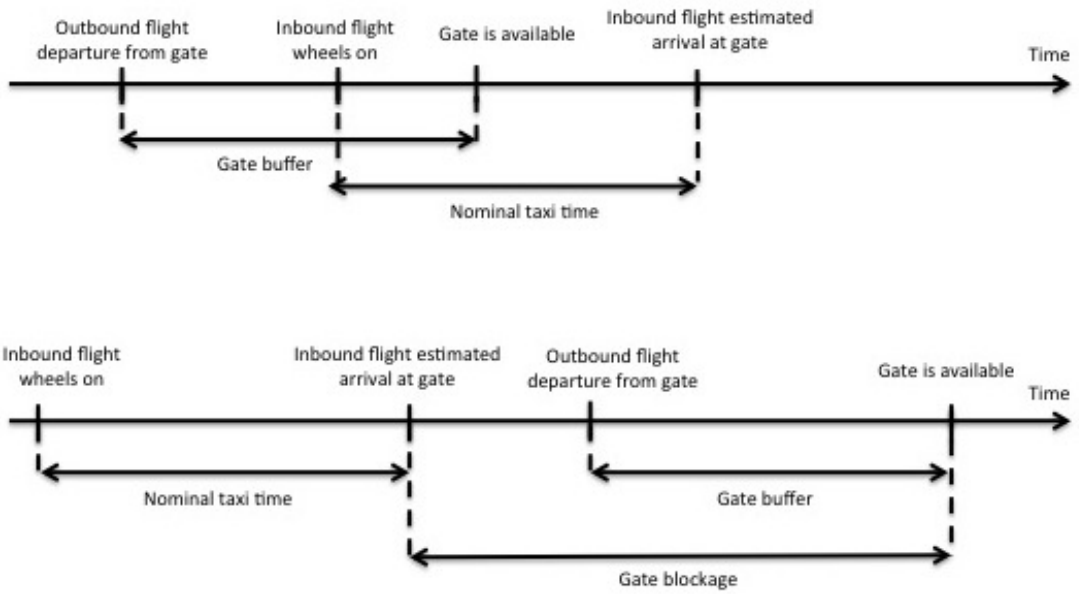


Figure 2.3: Two possible outcomes of a gate turn

The fields in Table 2.2 have been computed as described below:

- The frequency of gate blockage: Total number of gate blockages / Total number of arriving flights.
- The conditional average length of a gate blockage: Total sum of gate blockage minutes / Total number of gate blockages.
- The total number of flights whose blockage length is in the intervals (in minutes): $[1, 5]$, $[6, 10]$, $[11, 15]$, $[16, 20]$, and greater than 20 minutes.

We observe that although gate blockages are fairly rare on a percentage basis (averaging about 5% of all flights), in absolute terms this is a significant number, when you take into account the fact that there are roughly 25,000 domestic flights across the U.S. each day.

Note also that, although most gate blockages are of duration between one and fifteen minutes, a significant number are of longer duration (about 20%). Furthermore, note that for connecting passengers a delay in arrival caused by a gate blockage of even fifteen or twenty minutes may be sufficient to cause a missed connection.

Finally, we note a limitation in our analysis: In the case of extreme weather (e.g., thunderstorms impacting the airport), we may show a lengthy departure delay on an outbound

Station	Frequency of Blockages	Conditional Average Length of Blockage	[1;5]	[8;10]	[11;15]	[16;20]	>20
1	7.29%	10.10	370	399	276	135	160
2	4.17%	8.66	224	215	142	59	51
3	4.19%	8.05	222	203	120	54	34
4	6.84%	9.43	272	278	170	95	96
5	5.70%	11.04	189	199	119	74	112
6	5.36%	9.31	169	180	113	57	48
7	6.48%	9.37	200	184	96	62	63
8	2.89%	6.85	107	82	38	18	9
9	3.01%	7.69	99	82	38	18	9
10	4.43%	8.73	97	70	47	37	19
11	3.11%	7.27	60	54	27	8	8
12	4.83%	9.68	51	50	39	16	19
13	3.45%	8.79	41	28	16	11	11
14	6.63%	17.71	15	8	21	19	30
15	2.57%	12.29	9	8	3	6	8
Total Period	5.10%	9.39	2125	2040	1262	662	681
Percentage of Flights			1.60%	1.54%	0.95%	0.50%	0.51%
Percentage of Blocked Flights			31.39%	30.13%	18.64%	9.78%	10.06%

Table 2.2: Detailed results for period 3

flight that we compute to cause a lengthy gate blockage for the corresponding inbound flight. If ground operations are halted, however, then that inbound flight would have been delayed from reaching the gate even if the gate were unoccupied. In those extreme cases, our analysis may over-estimate the gate blockage.

Table 2.3 summarizes across all stations within a given time period. Observe that the results do not vary much from one time period to another.

Period	Frequency of Blockages	Conditional Average Length of Blockage	[1;5]	[8;10]	[11;15]	[16;20]	>20
Period 1	3.69%	7.69	450	363	193	95	63
Period 2	4.00%	8.28	772	686	357	196	148
Period 3	5.10%	9.39	2125	2040	1262	662	681
Period 4	4.83%	8.86	2192	1960	1196	601	586
All Periods	4.71%	8.92	5539	5049	3008	1554	1478
Percentage of Flights			1.57%	1.43%	0.85%	0.44%	0.42%
Percentage of Blocked Flights			33.31%	30.36%	18.09%	9.35%	8.89%

Table 2.3: Results for all the periods

In Figure 2.4, we show the breakdown by length of all gate blockages, across all fifteen stations for all four time periods. We break down blockages by five minute intervals. The y-axis shows the percent of flights falling into each bucket, relative to the total number of flights flown; each column specifies the absolute number of gate blockages as well as the percent of gate blockages that fall into that bucket. Note that roughly one third of the gate

blockages are more than ten minutes in duration and almost 10% are of more than twenty minutes, corresponding to almost 1500 flights.

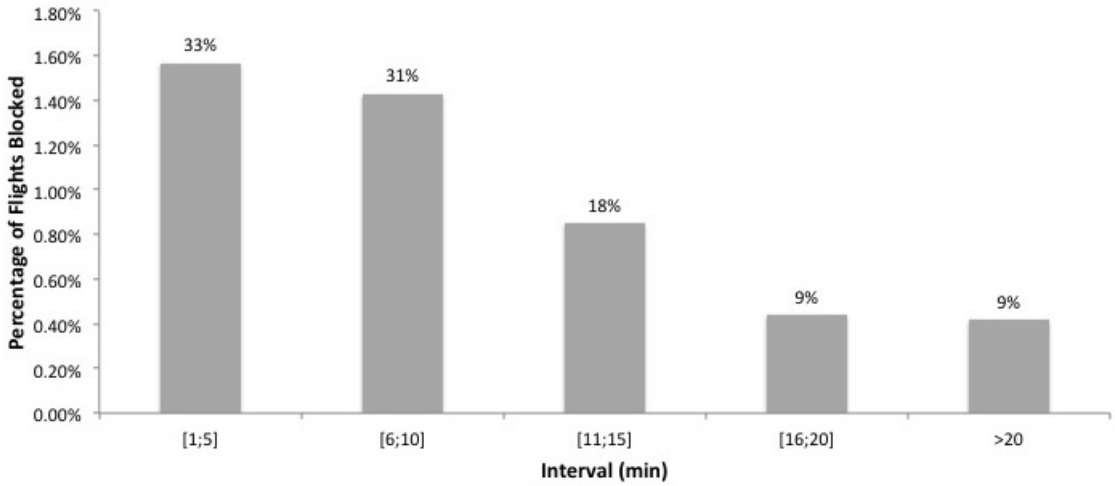


Figure 2.4: Distribution of blockage length over 15 airports and 4 periods of time

In Figure 2.5, we break down the flights by time period. Observe that the percent of overall flights delayed appears to go up slightly between the first two periods and the second two periods; the distribution of gate blockages across their respective lengths remains roughly comparable across all time periods, as shown in Table 2.4.

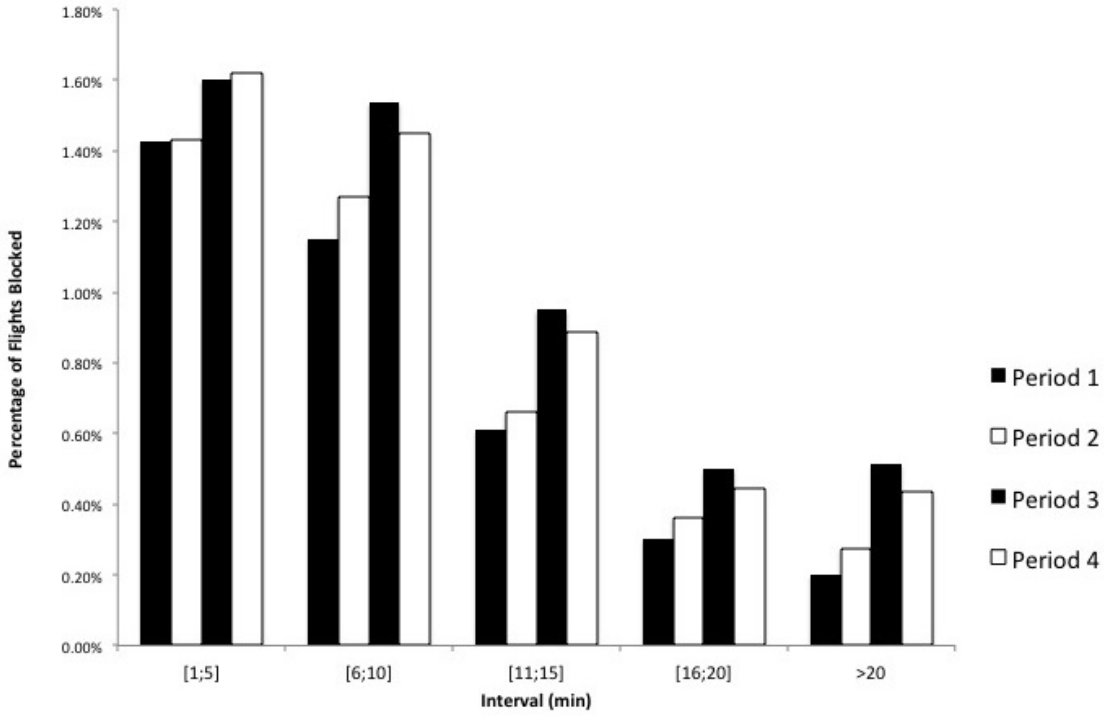


Figure 2.5: Distribution of blockage length over 15 airports for each period of time

Blockage length (min)	[1,5]	[6,10]	[11,15]	[16,20]	>20
Average percentage of blockage in that interval (%)	33	31	18	9	9

Table 2.4: Average distribution of blockage times

Figures 2.6 and 2.7 display the evolution over time of the probability and expected length of blockage for each station and over the four periods. On Figure 2.5, each bar is the percentage of flights blocked for each of the of fifteen airports during each of the four considered periods of time. On Figure 2.7 each bar is the average length of a blockage, conditioned on blockage occuring. Unlike the distribution of blockage length, the proportions of flights blocked in each airport have significant fluctuations over time (Figure 2.6); for instance, the percentage of flights blocked in station 12 is two times larger during Period 1 than the other periods.

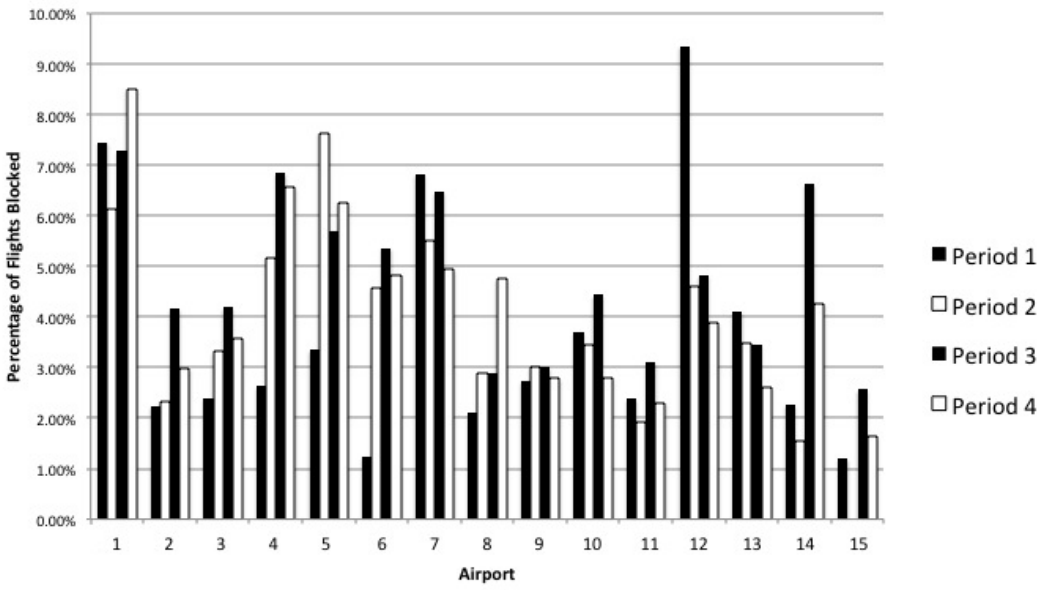


Figure 2.6: Percentage of flights blocked in each station

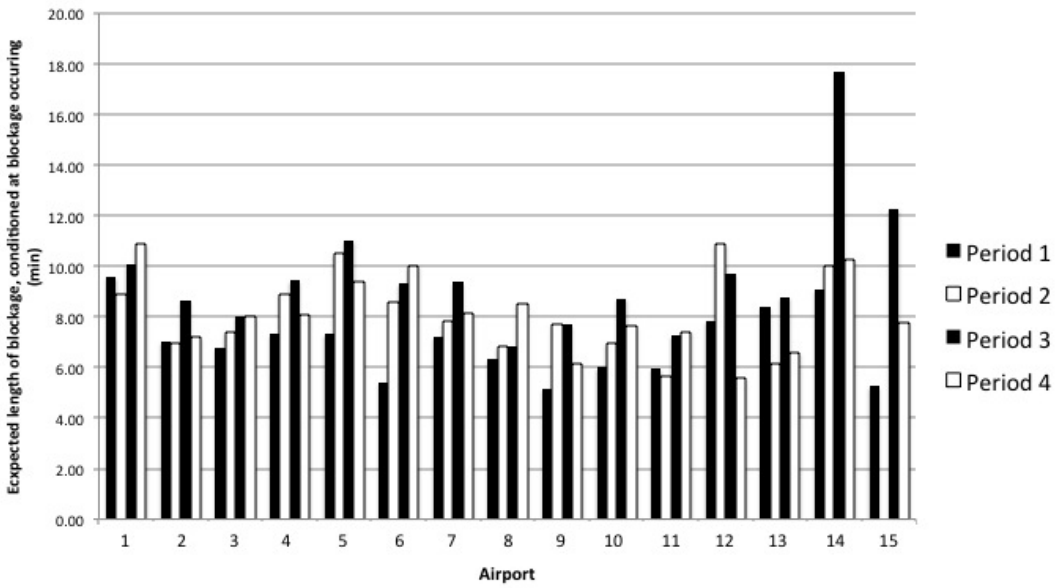


Figure 2.7: Conditional expected length of blockage

Finally, we look for possible correlations between airport characteristics (number of flights, gates, etc.) and the occurrence of gate blockage. For example, when there are more flights per gate, gate turns are tighter, and this suggests a higher likelihood of gate blockage. Figures 2.8 and 2.9 present the probability of gate blockages at each station versus the daily ratio of flights per gate, in Periods 3 and 4 which represent the most flights.

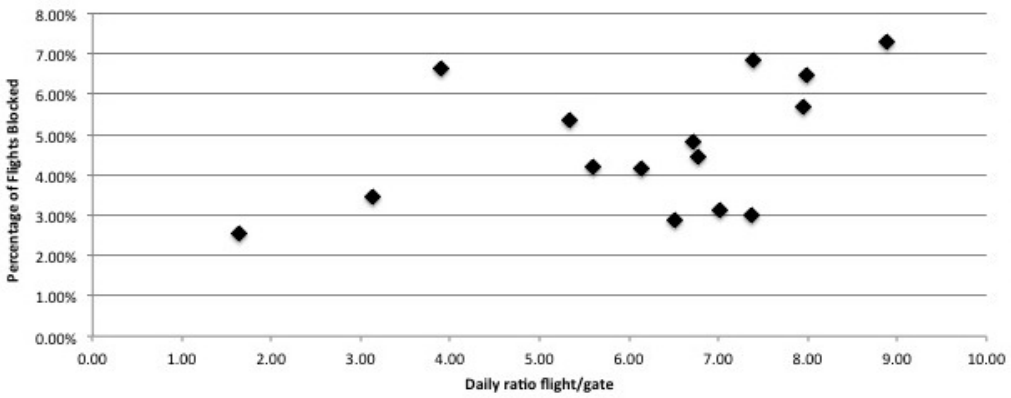


Figure 2.8: Percentage of flights blocked in each station as a function of the daily number of flights per gate in Period 3

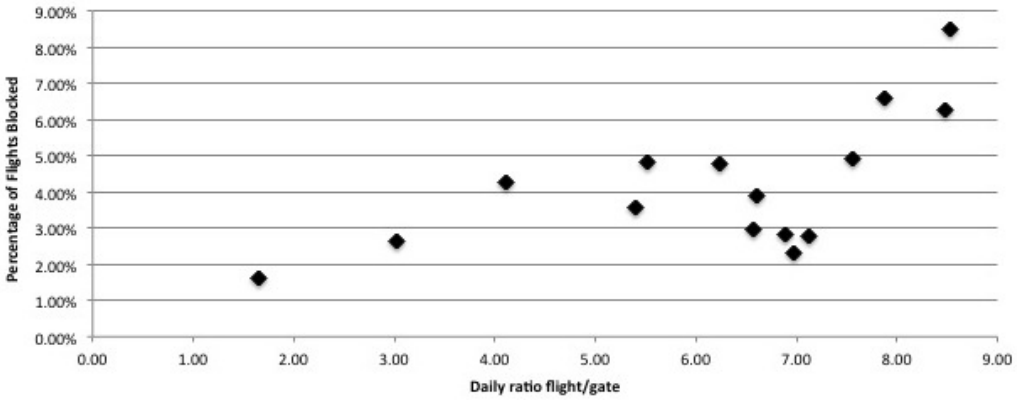


Figure 2.9: Percentage of flights blocked in each station as a function of the daily number of flights per gate in Period 4

We observe that stations at the tails with the lowest flight to gate ratio have the lowest probability of blockage, and similarly the highest ratios have the highest probabilities, which is not surprising. We do not observe a strong correlation in general, however, suggesting that many other factors beyond the amount of buffer in the gate turn impact potential for gate blockage.

This case study allows us to assess the frequency and the duration of gate blockages in a panel of US airports, over different time periods. The key results of this analysis are that (1) gate blockages affect roughly 5% of flights, (2) their duration is smaller than 10 minutes for 60% of them but around 9% of the blockages last more than 20 minutes, (3) the

frequency and the length of gate blockages are highly variable depending on the stations, however they are similar from one period of time to another.

These observations show that gate blockages have a significant impact on daily airline operations, and motivate us to take into account gate blockages when building a gate assignment, which is the objective of the next sections.

2.4 Robust Gate Assignment

Motivated by the analysis presented in the previous section, we have developed a mathematical programming-based approach to the gate assignment problem, with the goal of improving solution robustness by incorporating variability in departure and arrival times. We first consider the case where all aircraft types (and thus all flights) are compatible with all gates; we refer to this as the *homogeneous* case. Then we generalize to the *heterogeneous* case, where certain gates are incompatible with certain aircraft types and thus the corresponding flights.

2.4.1 Robust Homogeneous Gate Assignment

To model the *Robust Homogeneous Gate Assignment Problem* (*RHoGA*), we consider a network flow-based formulation, as is commonly used in airline planning. The key difference in our approach, however, is the perspective: *rather than flowing aircraft through gates or stations, as is commonly seen*, **we flow gates through aircraft turns**. Figures 2.10 through 2.13 represent a portion of a sample network.

Figure 2.10 depicts the nodes in this network. There is one node (S_g) for each gate g representing that gate at the start of the day with a supply $d_g = 1$ and one node (E_g) for each gate g with a demand $d_g = -1$ representing that gate at the end of the day. In addition, there is one pair of nodes for each aircraft turn a without any supply ($d_a = 0$). Note that some turns consist of both an inbound and an outbound flight, while some represent just an outbound flight (where the aircraft would have overnighted at the station the preceding night) and some represent just an inbound flight (where the aircraft is intend to stay overnight at the station). We create an arc with lower and upper bounds of 1 between the inbound and outbound parts of the nodes, which ensures that each aircraft turn is assigned

to exactly one gate.

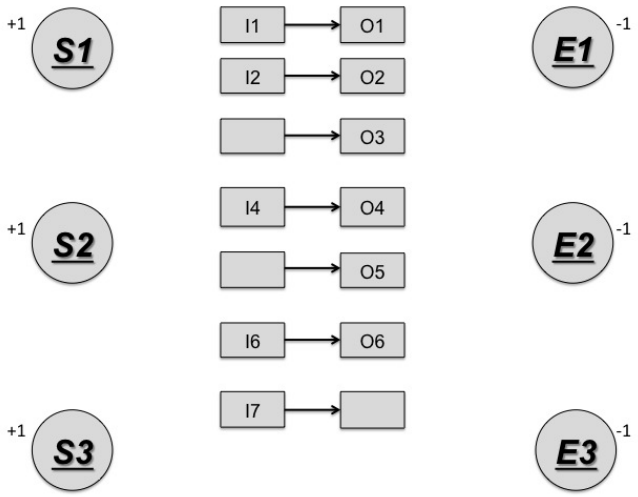


Figure 2.10: Nodes of the network

Figures 2.11, 2.12, and 2.13 depict the arcs in this network. Figure 2.11 shows arcs that originate from the gate start nodes. Generally, there is one arc from each gate to each aircraft turn; flow over this arc corresponds to assigning that turn as the gate's first activity of the day. If a specific flight is pre-determined to be the first flight of the day out of a given gate (for example, when using the model in an operational context, where last night's gate occupants are known), then there would only be one corresponding arc in the network to represent this — for example, in Figure 2.10, the arc from S_2 to O_3 and from S_3 to O_5.

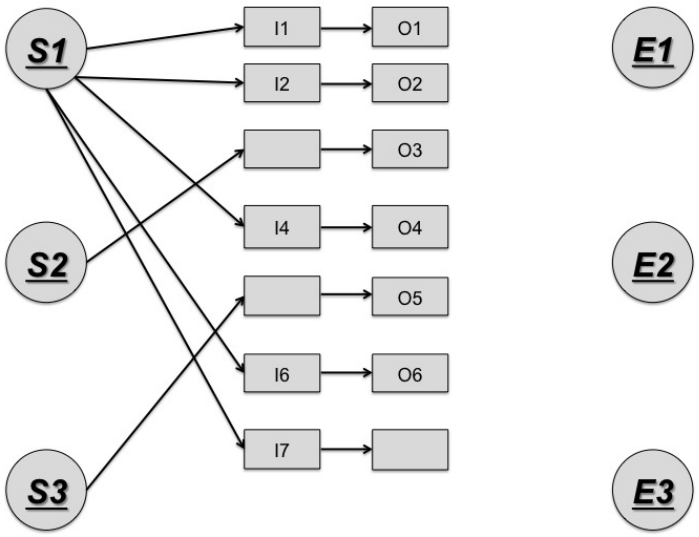


Figure 2.11: Arcs leaving the gate start nodes

Figure 2.12 depicts arcs between aircraft turn nodes. Generally, there is an arc from aircraft turn T_1 to aircraft turn T_2 so long as the departure time O_1 plus the minimum buffer time of the gate is earlier than the arrival time I_2. [Note that this can be relaxed in planning mode to help *determine* minimum buffer gate time – for example, if an outbound flight often leaves early and an inbound flight often arrives late, pairing their respective turns may be desirable, even if they are closer together in time than the system minimum.] If an aircraft turn is an outbound flight only, then it cannot have inbound arcs from other aircraft turns, as it is presumed to be the first flight of the day from a gate. Similarly, if an aircraft turn is an inbound flight only, then it cannot have outbound arcs to other aircraft turn nodes, as it is presumed that the aircraft will remain on the ground overnight.

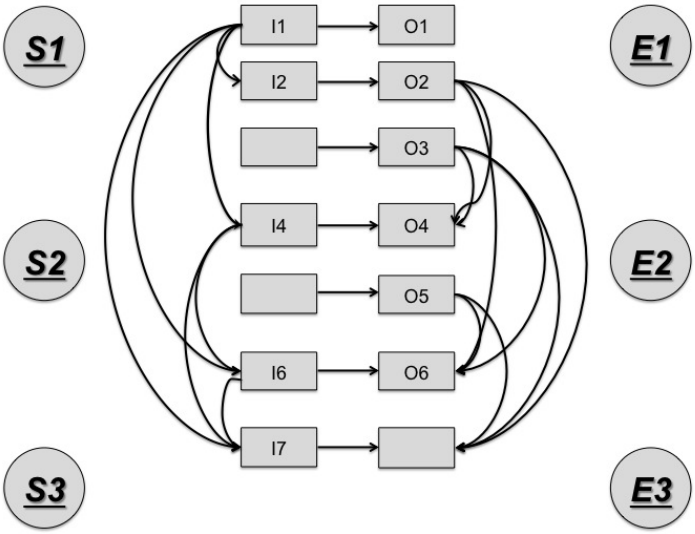


Figure 2.12: Arcs between aircraft turns

Finally, Figure 2.13 depicts arcs into the gate end nodes. There is generally one arc from each aircraft turn node to each gate end node; flow on this arc represents that aircraft turn being the last activity of the day at that gate. If a flight has been pre-assigned to a specific gate to overnight, then that would be the only arc into the gate end node, as illustrated by the arc from I_7 to E_3.

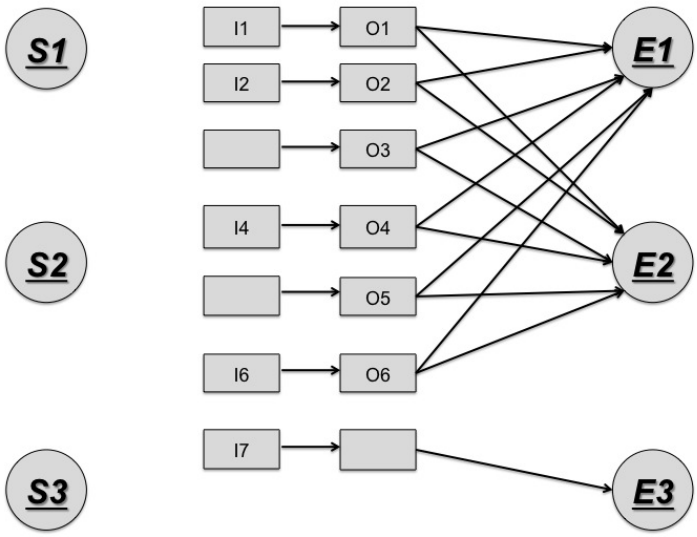


Figure 2.13: Arcs arriving to gate end nodes

[Note that, in theory, we could also include arcs from S_g to E_g, representing the case in which gate g is unused throughout the day. In practice, such an occurrence rarely happens.]

Clearly, a path through this network, starting from gate S_g and ending at E_g, is a valid sequence of activities to be assigned to gate g.

The resulting model is a pure min-cost flow problem and therefore has the following structure:

$$
\begin{aligned}
\min_{x} \quad & \sum_{(a,b)\in A} c_{ab}x_{ab} && && (1.1:\ \text{Objective})\\
\text{subject to} \quad & \sum_{b:(a,b)\in A} x_{ab} - \sum_{b:(b,a)\in A} x_{ba} = d_a && \forall a \in N && (1.2:\ \text{Flow balance})\\
& x_{ab} \geq l_{ab} && \forall (a,b) \in A && (1.3:\ \text{Lower bound})\\
& x_{ab} \leq u_{ab} && \forall (a,b) \in A && (1.4:\ \text{Upper bound})
\end{aligned}
$$

where N and A are the sets of nodes and arcs of the network described above.

The variable x_{ab} represents the flow along arc (a, b). Having a flow of 1 along an arc means that the two adjacents nodes belong to the same gate schedule.

The supply and demand d_a are set according to the description above (1 for the start of the day nodes, -1 for the end of the day nodes and 0 for all the aircraft turns nodes).

The bounds l_{ab} and u_{ab} are set to 0 and 1 on any arc with the exception of the arcs

resulting from the splitting of the aircraft turns nodes whose lower and upper bounds are both 1 in order to ensure that each aircraft turns is assigned to exactly one gate.

The cost per unit of flow along arc (a, b) is set according to the estimated probabilities of blockage and the associated metrics (see Section 5).

Finally notice that we do not require a binary constraint on each variable x_{ab} since the arc incidence matrix is totally uni-modular and therefore we will get an integral optimal solution.

The model above can be applied to the first three objective functions that we have outlined.

For the fourth (objective (W)), we must modify the problem slightly. Specifically, we define a new variable $z \in R$, which represents the maximum expected blockage. We then add one constraint for every aircraft turn arc x_{ab} of the form:

$$z \geq c_{ab} x_{ab}$$

Because we are minimizing z, the constraints effectively impose that $z = \max_{(a,b) \in A} c_{ab} x_{ab}$. Note that in this case we have now violated the total uni-modularity of the matrix and therefore we also have to impose integrality requirements on the flow variables.

2.4.2 Robust Heterogeneous Gate Assignment

In the previous subsection, we assumed that all flights (or, more precisely, all aircraft turns) could be assigned to all gates. In practice this is often not the case. For example, certain gates are not equipped to handle either very large or very small aircraft. Therefore, we generalize *RHoGA* to take this into account, defining the *Robust Heterogeneous Gate Assignment (RHeGA) Problem*.

Specifically, given a set of gates, a set of aircraft turns (as defined in the previous subsection), and the added component of a *compatibility matrix*, which defines for each gate and aircraft turn pairing whether the associated assignment is feasible, the objective is to assign each aircraft turn to a compatible gate, so as to maximize robustness.

We use a similar network structure, where gates rather than aircraft flow through the network, as in *RHoGA*. However, we can no longer model the problem as a pure minimum cost flow problem, because the gates are no longer fully interchangeable, i.e., they cannot be treated as commodities, because not all flights can be assigned to all gates.

Instead, we create one copy of the network *for each gate*, similar to the network from RHoGA (but without splitting the aircraft turn nodes). We remove from this network,

however, all nodes corresponding to aircraft turns that are not compatible with the gate and all associated arcs going into or coming out of those nodes. Each gate-specific sub-network now again captures feasible paths (i.e., sequences of activity).

We then still need to ensure that all aircraft turns get assigned to exactly one gate. To do so, we add one constraint for each aircraft turn ensuring that the flow into the node representing that aircraft turn, across all arcs in the sub-networks for all gates, must equal one. Observe that we have, in the process, violated the pure minimum-cost flow structure. Therefore, we must now add integrality requirements — the flow on all arcs must be non-fractional to ensure feasible paths, i.e., gate-specific sequences of tasks.

The resulting RHeGA model is a network flow problem with side constraints, which can easily be viewed as a multi-commodity flow problem.

$$
\begin{array}{lll}
\min\limits_{x} & \sum\limits_{g\in G}\sum\limits_{(a,b)\in A} c_{ab}x^g_{ab} & (1.1:\ \text{Objective}) \\
\text{subject to} & \sum\limits_{b:(a,b)\in A} x^g_{ab} - \sum\limits_{b:(b,a)\in A} x^g_{ba} = d_a \quad \forall a\in N\ \forall g\in G & (1.2:\ \text{Flow balance}) \\
& x^g_{ab} \geq l_{ab} \quad \forall (a,b)\in A\ \forall g\in G & (1.3:\ \text{Lower bound}) \\
& x^g_{ab} \leq u_{ab} \quad \forall (a,b)\in A\ \forall g\in G & (1.4:\ \text{Upper bound}) \\
& \sum\limits_{g\in G} x^g_{ab} = 1 \quad \forall (a,b)\in A & (1.5:\ \text{Side constraint}) \\
& x^g_{ab}\ \text{binary} \quad \forall (a,b)\in A\ \forall g\in G & (1.6:\ \text{Integrality})
\end{array}
$$

2.5 Computing Coefficients

The two models presented in Section 2.4 require objective coefficients that capture the probability that a gate blockage occurs between two given aircraft turns.

As is the case with virtually all airline planning problems, identifying the appropriate data to populate our models is a non-trivial challenge. Historical data can be used but there may not be adequate data about past events to predict future occurrences. This is particularly true when making decisions about a future schedule for new flights that have not been included in prior schedules. Furthermore, historical data may not accurately reflect the future, with potential system changes having significant impact. Nonetheless, with these caveats in mind, approximations of objective parameters must be made. We do so by using historical data to predict how alternate schedules might have performed. We note

that an important area of future research is to work towards improving these parameter estimates.

To predict the probability of aircraft turn T_1 imposing gate blockage on aircraft turn T_2 (for objective P), and the expected amount of this blockage (for objective X), we rely on historical data. Specifically, we consider all days during which both flights operated from a common scheduling period. We consider only day-specific flight pairs so that the correlation of weather impacts will be incorporated (for example, if T_1 is delayed in departing due to local inclement weather, it is more likely that T_2 will be delayed in arriving as well, and this should be recognized in our approximation).

Consider two possibilities. First suppose that T_1 and T_2 did in fact share a common gate on day d during our historical period. Was there gate blockage on this day and, if so, for how long? We know from the carrier-provided data what time T_1 pushed back from the gate (d_1) and therefore by adding the minimum buffer gate time b when the gate was available $(d_1 + b)$. We also know when T_2 landed (l_2), and we know when T_2 arrived at the gate (a_2). What we do not know, however, is how much of the window from l_2 to a_2 was taxi time and how much (if any) was gate blockage. To try to deconstruct this, we consider the default nominal taxi time $\tilde{T}$ defined in Section 2.3, such that $l_2 + \tilde{T}$ is the *estimated* time of arrival at gate for aircraft turn T_2. Therefore, we assume that the remaining time $T_{2,1}^{blockage} = (d_1 + b - l_2 - \tilde{T})$ was gate blockage if positive and that there wasn't any gate blockage otherwise.

Second, suppose that aircraft turns T_1 and T_2 did *not* share the same gate on day d. We still need to know what would have happened if they had shared a gate since in our model we consider all possible assignments. Since we consider a nominal taxi time our approximation of the arrival time at the gate for an inbound flight does not depend on its gate, therefore we can apply the exact same reasoning as in the first case to obtain an approximation of an eventual gate blockage that could have happened if the two flights had shared the same gate.

Once we know how to compute an estimation of the gate blockage length between any pair of aircraft turns, we just need to loop through the flight data provided by the carrier, compute the estimated gate blockage length for each day on which those two aircraft turns occurred (even if they were not assigned to the same gate) and calculate the cost coefficient relative to this pair for each one of the four objectives described in Section 2.2.

2.6 Computational Experiments

The purpose of our computational experiments is two-fold. First, we want to assess the tractability of our approaches, to assess if they are viable for use in practice, in planning as well as in operational contexts. Second, we want to analyze the potential benefit to be gained by using optimization-based techniques to build gating schedules.

In Section 2.6.1 we focus on the homogeneous problem, where all aircraft types (and thus all flight turns) are compatible with all gates. Section 2.6.2 addresses the heterogeneous problem, where certain flight turns are incompatible with certain gates.

2.6.1 Homogeneous Experiments

For our homogeneous experiments, we focused on the aircraft turn data from one specific date, as provided by the carrier. We considered five stations (2, 3, 4, 5 and 6), which were among the largest in the network.

Using historical data to generate objective coefficients (as described in Section 5), we created four different schedules for each station, optimized under four different objective functions. Specifically, we created:

- $(optP)$: the schedule that minimizes the sum of the probabilities of a gate blockage (i.e., the expected number of gate blockages),
- $(optX)$: the schedule that minimizes the sum of the expected blockage minutes,
- $(optC)$: the schedule that minimizes the sum of the expected connecting passenger blockage minutes,
- $(optW)$: the schedule that minimizes the maximum expected blockage length.

In addition to these four schedules, we also constructed (FIFO), a schedule that assigns flights to gates in a first-in-first-out order.

Figure 2.14 provides the results. Each row corresponds to a different schedule. Each column corresponds to a different objective function. For example, for station 2, the schedule that minimizes the sum of the expected blockage minutes (schedule $optX$) has an objective of 2.071 under the objective W: maximum expected blockage.

For each column, it is of course true that the best value corresponds to the schedule which was optimized relative to that objective. It is interesting to note, however, that the

2	P	X	C	W
optP	0.505	33.714	959.999	11.464
optX	1.242	10.221	196.941	2.071
optC	3.332	85.875	140.019	25.043
optMM	1.888	20.921	635.745	1.25
FIFO	2.507	124.612	3597.291	15.142

5	P	X	C	W
optP	1.198	58.779	1417.905	9.033
optX	1.655	28.772	751.391	4.033
optC	2.89	105.455	433.691	24.566
optMM	3.021	62.758	1607.543	4.033
FIFO	3.707	128.609	3589.717	12.034

3	P	X	C	W
optP	0.535	19.748	535.71	4.866
optX	1.102	11.961	457.372	3.071
optC	2.641	64.605	303.571	18.068
optMM	1.44	21.566	698.03	1.689
FIFO	4.131	136.061	3795.56	15.892

6	P	X	C	W
optP	1.736	90.327	1922.921	13
optX	2.427	57.017	1344.787	8.172
optC	4.098	113.491	622.915	13.851
optMM	4.323	101.23	2033.108	5.407
FIFO	4.345	184.431	4819.237	18.6

4	P	X	C	W
optP	0.14	10.813	256.344	4.655
optX	0.185	3.343	17.994	2.48
optC	1.817	48.549	5.571	25.32
optMM	0.304	5.817	26.43	1.75
FIFO	1.668	82.61	1918.514	8.481

Schedules:
optP: optimal schedule under P objective
optX: optimal schedule under X objective
optC: optimal schedule under C objective
optMM: optimal schedule under MM objective
FIFO: First In First Out schedule

Figure 2.14: Results of the homogeneous model

optimal schedule for the metric X: sum of the expected block time for each turn, performs very well under the other costs – in fact, in almost every case, for any given metric, the $optX$ schedule is second only to the schedule optimized relative to that objective function. Our intuition to support this observation is that the X metric is a good trade-off between the three others objectives. The FIFO schedule is significantly worse than all other schedules for all metrics in almost all cases.

2.6.2 Heterogeneous Range Experiments

We have noted that the homogeneous model is a pure minimum cost flow formulation, which naturally has integrality properties, while the heterogeneous model has side constraints that can induce the need for branching.

To assess the pure impact of the formulation itself on computational performance, we consider the situation of no compatibility restrictions on the aircraft type (all flights can use all gates), and find the optimized schedule under the objective (X), on a specific date

for station 5 using the two models. As expected since we do not have any gate restriction, we find the same optimal objective: 28.772 minutes. The run time for the homogenous model is 1 second and the run time for the heterogeneous model is 60 seconds. It is interesting to note that the difference in structure of the two models results on a much longer computational time for the heterogeneous model. All runs were conducted on a Intel Xeon E31230 computer clocked at 3.20GHz and 8GB DDR3 RAM clocked at 1333MHZ solved with CPLEX version 12.1.

We then seek to understand how the level of incompatibility between gates and aircraft types impacts performance (both for the run time and the optimal objective). To do so, we design the following *heterogeneous range experiment*: We consider 3 different aircraft types: types S,M and L corresponding to small, medium and large aircraft and representing $20\%, 4\%$ and 76% of the aircraft turns. We assume that aircraft of type M can go to any gate, but that aircraft types S and L can be constrained. Under that assumption there are three possibilities for a gate:

- $([S, M])$: it is compatible with only types S and M,
- $([M, L])$: it is compatible with only types M and L,
- $([S, M, L])$: it is compatible with all three aircraft types.

We consider station 2; this airport has 19 gates. The different possible gate constraints are represented in Figure 15. An entry at coordinates (x, y) is equivalent to a scenario in which x gates are $([S, M])$, y gates are $([M, L])$ and the remaining $19 - x - y$ gates are $([S, M, L])$.

For each of these scenarios we run the heterogeneous model on metric X: sum of the expected gate blockage lengths. Note that for some of these points (typically when all gate are restricted for one aircraft type) the problem will be infeasible which corresponds to a infinite cost. However, for the point $(0, 0)$, which represents the situation with no gate constraints at all, we will find the same optimal objective as the one obtained in the homogeneous model. The main purpose of this experiment is to study how the objective increases when we add gate constraints and also to study the impact on computational performance as a function of how constrained the model is.

To do so, we ran the heterogeneous code for each point of Figure 2.15. We represent their optimal objective values in Figure 2.16 and the computational time in Figure 2.17. To compare qualitatively the different values, we use gray markers: lightest circles correspond

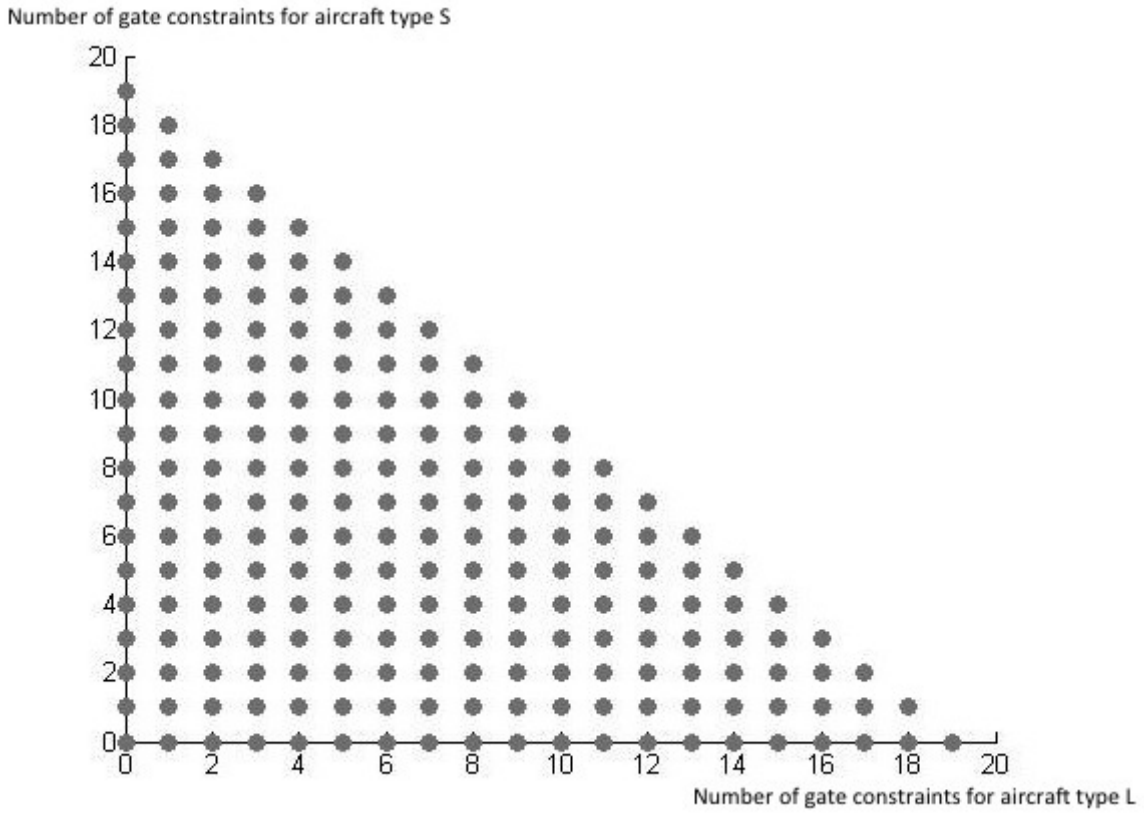


Figure 2.15: Different scenarios considered in the heterogeneous range experiment

to the lowest values and the darkest circles to the highest values. Star-shaped markers are used when the problem is infeasible.

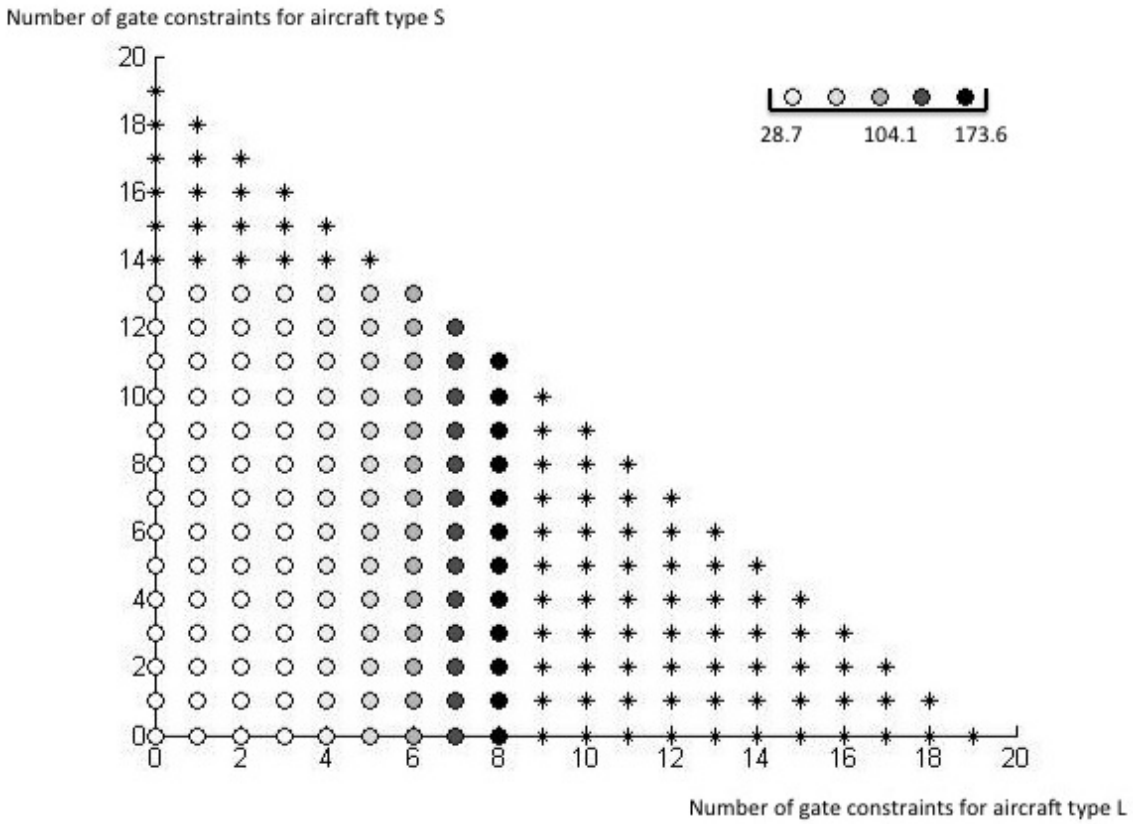


Figure 2.16: Objectives of the heterogeneous range experiment

The computational times (in seconds) are distributed as shown in Table 2.5:

Min	25% Percentile	Median	75% Percentile	Max	Mean
8	17	31	54.5	237	45

Table 2.5: Distribution of the computational times of the heterogeneous range experiment

The fact that the objective does not visibly change when moving in the vertical direction implies that constraining aircraft type S does not significantly impact the total cost. However constraining aircraft type L has a high price, even when aircraft type S is not constrained. This results from the fact that there are roughly four times more aircraft of type L than aircraft of type S.

In Figure 2.16, we can see that constraining the problem typically reduces the computational time and that the lowest run times are obtained when the problem is almost infeasible in the sense that adding one gate constraint would make the problem infeasible. However

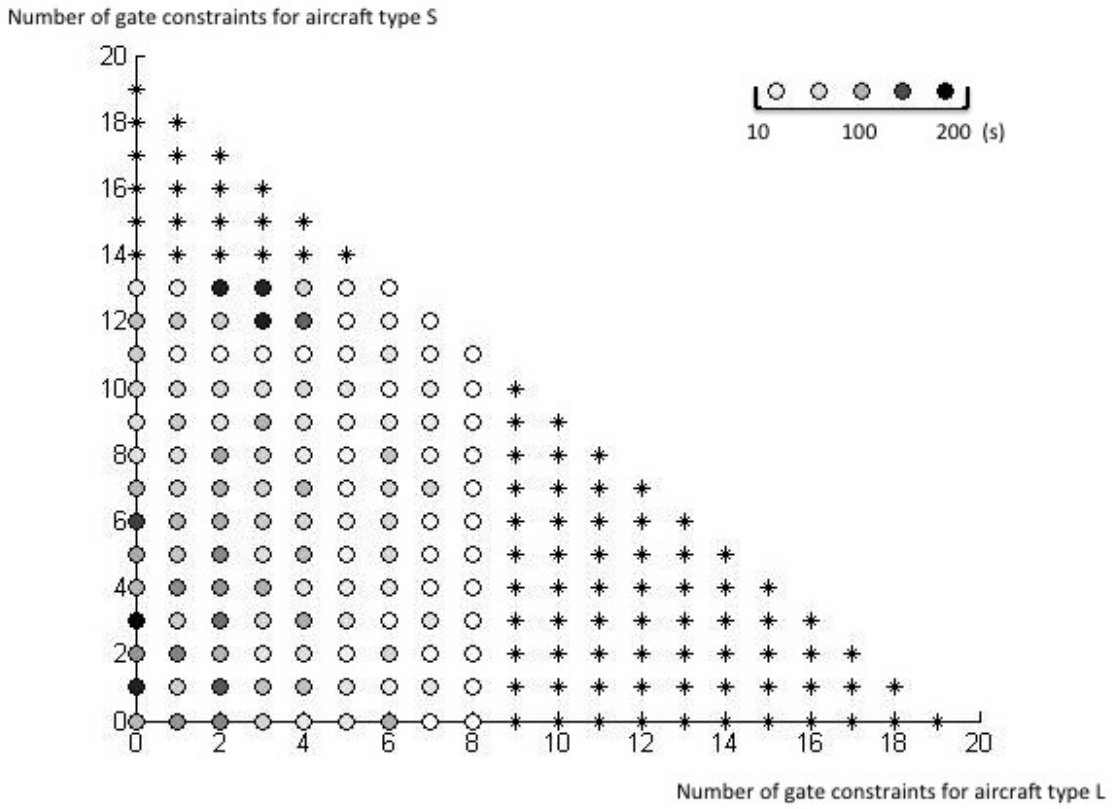


Figure 2.17: Computational times of the heterogeneous range experiment

we notice a zone of higher computational times when aircraft type S is very constrained. Furthermore, the time to prove infeasibility was very short, typically on the order of one second. As a whole, there does not seem to be any significant correlation between level of constraints and run time, with all problems solving quickly.

2.7 Future Research and Conclusions

2.7.1 Model Extensions and Future Research

We present here some ideas for future projects to extend our research.

- Objective based on missed connections:

  In our (C) metric: expected connecting passenger blockage minutes, we use the blockage time associated with connecting passengers as a surrogate for missed connections. This metric has two key limitations:

1. It does not recognize the daily variability in passenger itineraries on any given flight.
2. It treats the impact of blockages in a linear fashion. In fact, the impact is really binary: either the blockage is long enough to induce a missed connection or not.

Ideally it would be better to estimate (and do so more accurately) the expected number of passengers missing their connections due to gate blockage. To do so, we recommend using a stochastic distribution for the connection time depending on influencing factors, such as the origin and destination of the flight and the hour of the day. The expected number of missed connections, given a delay of d minutes, would be the number of connecting passengers multiplied by the cumulative distribution function of the connection time evaluated in d (i.e., the probability that a connection time is less than d minutes).

- Adjacency issues

  An important constraint faced by airlines when building a gate assignment is the adjacency constraint which means that, in certain cases, two given aircraft type cannot be simultaneously at two adjacent gates: the orientation of adjacent gates may make it impossible to fit two wide-bodied aircraft next to each other simultaneously, for instance. In order to take those adjacency issues into account in our model, we would need to add a constraint for each pair of adjacent gates and associated pairs of incompatible (in time and fleet type) aircraft turns to ensure that at most one of the two assignments is made. However this potentially represents a very large number of constraints and so alternative modeling and/or solution techniques might need to be developed.

- Analysis of delay propagation in a multi-station network

  Propagation of delays throughout the day is one of the consequences of gate blockage: a flight delayed due to gate blockage reaches its gate late and is consequently likely to leave the gate late, which increases the risk of generating a new gate blockage at the current station (as well as many other negative system impacts). Interestingly, this departure delay also reduces the chance that the flight will be blocked at its next destination. As such, a more effective gating approach would take into consid-

eration the down stream effects caused by gate blockage-caused delay propagation.

- Improved Estimation of Objective Coefficients

  We conducted our analyses by using historical data to estimate delay probabilities, and then using the probabilities to optimize gating assignments for the same time period. In reality, of course, we could not have the known data for the same time period that we are trying to plan. Therefore, a valuable area of research (not only for the purpose of the robust gating assignment problem but for a wide range of airline planning problems as well) is to better develop probability distributions for future flights.

2.7.2 Conclusion

A study of the current situation on a large sample of U.S. airports for a major domestic carrier shows that gate blockage occurs during 5% of commercial flights, with 2% of all flights being delayed by at least 10 minutes. Consequences of gate blockage such as missed connections and increased costs for airlines make it important to address this problem when building a gate assignment.

We propose network-based models for both the homogeneous and heterogeneous versions of the problem, show that these models are computationally tractable, and demonstrate that, for several different metrics of robustness, they significantly improve performance over a first-in-first-out assignment paradigm.

Not surprisingly, the heterogeneous model is slightly more computationally intensive than the homogeneous model due to the pure minimum-cost-flow structure of the homogeneous model. Nonetheless, for realistic instances the heterogeneous model solves quite quickly in practice, with run times on the order of a few minutes at most.

Our computational experiments show that these models give better results regarding the four tested objectives related to gate blockage than a standard first-in-first-out algorithm. Even if other criteria are taken into account by airlines when building their schedules, our research gives a useful tool which can be used to compare several possible schedules according to gate blockage metrics and will allow airlines to select more robust choices for their gate assignment.

CHAPTER 3

A Stochastic Programming Approach to Reduce Patient Wait Times and Overtime in an Outpatient Infusion Center

3.1 Introduction

The University of Michigan Comprehensive Cancer Center *(UMCCC)* receives over 50,000 patients a year for infusion treatments. Visits have been increasing at a rate of nearly 5% per year and accommodating all patients with a fixed capacity is increasingly challenging. This increase in demand presents a challenge faced by many cancer centers [Erikson et al., 2007]. Consequences include long patient wait times and staff overtime. A key contributor to this is the high variability in infusion duration. Our objective is to take this uncertainty into account when setting patient appointment times to improve the quality of the appointment schedules.

3.1.1 Background and Motivation

Chemotherapy is used either before definitive local therapy (neoadjuvant), after definitive local therapy or to treat metastatic or recurrent cancer. Some patients receive chemotherapy a few times a week while others may be treated less frequently. The unique nature of cancer to each patient requires individualized treatment plans which are developed by the patient and his or her provider [Society,].

Chemotherapy is administered using a variety of delivery methods. These methods include oral, intravenous, biliary tube, intraperitoneal, intrathecal, and intravesical. Chemotherapy is most commonly administered intravenously, in one of two ways. The first is the drip bag method. In this method, the drugs are slowly dripped at a certain rate through an IV bag that is connected to the patient. The second is done by syringe. The drug is pushed

through the syringe and into the patients veins. In either method, all patients first undergo a preparation phase with a nurse, which includes seating the patient in an available infusion chair, making sure the patient has IV access, and administering pre-medications [Itano and Taoka, 2005].

For some patients, infusion is part of a full day process that can include appointments in: (1) phlebotomy to have blood drawn, (2) clinic to see their provider, who uses the blood results to determine if the patient is healthy enough to undergo treatment that day (the blood work may also be used to decide what the treatment should be — e.g., dosage), and (3) infusion to receive their chemotherapy treatment. Other patients may only have (1) or (2) before their infusion, or may bypass both of these altogether. Depending on the visit, any one patient may be at the cancer center ranging anywhere from an hour to the entire day.

About 7% of patients seen on an average day are in the middle of their treatment regimen. These patients come directly to the infusion center in many cases. New patients still require an appointment at the clinic and typically some buffer between appointments with other parts of the cancer center is provided to guarantee with high probability that they will be on time when arriving at the infusion area. The typical infusion process that we consider in this paper has 5 main steps:

1. Arrival: patient arrives at his/her infusion appointment time,
2. Waiting time: patient waits until an infusion chair and a nurse are available,
3. Preparation time: nurse brings the patient to his/her infusion chair and prepares the patient to receive the treatment,
4. Treatment time: chemotherapy drugs are administered via infusion,
5. Discharge: at the end of treatment, the patient is discharged.

In most outpatient infusion centers each nurse is responsible for a pod of three to four chairs; the nurse moves from patient to patient, preparing them to receive the chemotherapy drug, monitoring their infusion process and finally discharging them. Depending on the appointment arrival schedule, and the duration of the infusion, the nurse may be attending to three or four patients at one time. Thus, a large portion of the workload is conducted in parallel. If several patients are waiting, the nurse usually takes care of the patient with the earliest appointment time first. Therefore, in our model we assume a first come first serve policy.

One of the main challenges in scheduling appointment times for chemotherapy patients is the uncertainty in treatment times. In order to measure this variability, we analyzed data from the *UMCCC* collected electronically from August 1, 2014 to November 30, 2014, which represents over 10,000 visits. For each visit, we compared the scheduled appointment length to the actual appointment length. Our results show large variability for all types of appointments we studied, appointments ranging from short (30 minutes) to long (8 hours or more). This represents 18 groups of patients having the same scheduled appointment time, but not necessarily the same treatment protocol. Most groups have at least 400 realizations in our database, the least represented group has 70 samples. We typically observe a wide spread of the actual treatment length, centered around their mean, which is always close to the scheduled appointment length. For all of our computational experiments, we use the distributions obtained from this data set. We present the distribution of actual appointment lengths for patients who have been scheduled for 150 minutes in Figure 3.1 as an example.

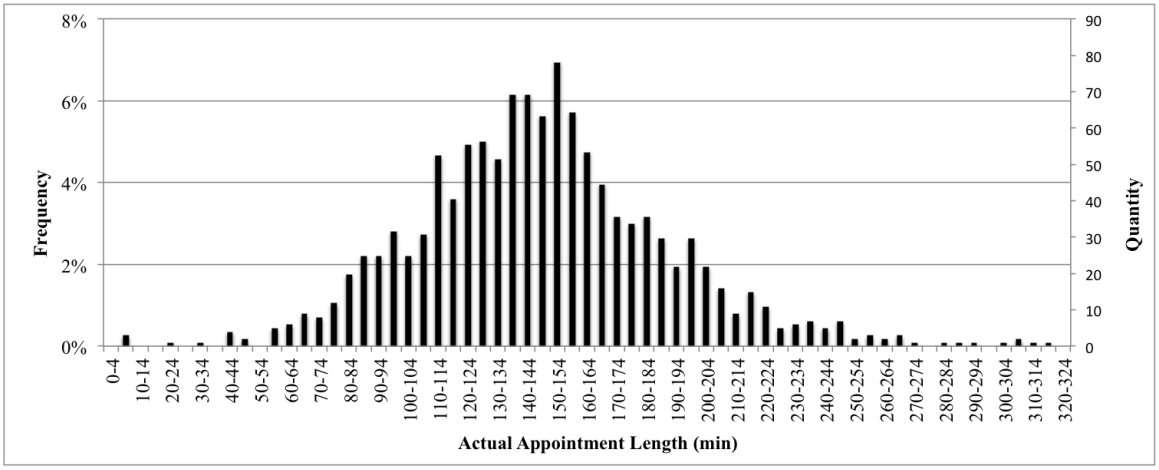


Figure 3.1: 150 Minute Scheduled Appointments

Typical causes of deviation in the actual length of appointments include:

- Early termination of infusion for patients who are not tolerating their treatment
- Complications due to patient adverse reactions
- Last minute change of administered drug which results in a treatment length (shorter or longer) that was not anticipated in the schedule

3.1.2 Appointment Scheduling Process

Patients are scheduled for an appointment dynamically, meaning that when a scheduler has to set a visit date and an appointment time for a patient, the schedule is only partially filled, and more patients may be scheduled at a later time. This sequential decision process is referred to in the literature as an *online scheduling* problem — see chapter 3.5 in the scheduling textbook by Pinedo [Pinedo, 2012] for an introduction to online scheduling or [Erdogan and Denton, 2013] and [Erdogan et al., 2015] for two papers describing online scheduling and applications to patient appointments scheduling.

In this paper, we focus on fine-tuning an initial appointment schedule in a second scheduling phase. We assume that an initial schedule has already been constructed and, shortly before the schedule is to be implemented (e.g., a day or two), this schedule is refined by making small changes to those initial appointment times to create a more robust schedule. This second phase scheduling process is not currently implemented at *UMCCC* but, based on our discussion with collaborators, making changes to patient arrival times is feasible (small timing changes are already frequently made for other purposes) and, as we show, could yield significant improvements to overall service quality.

When refining appointment times it is important to consider that patients have already been notified of their appointment time. Therefore a large deviation from the initial schedule (e.g., an appointment moved from morning to afternoon) might have undesirable impacts on a patient's personal schedule for that day. However, we will demonstrate that minor timing changes in the initial schedule are often enough to significantly improve the quality of the schedule. To guarantee that changes are relatively small, we assume there is no change to the original sequence of appointment times. This means that we create a new schedule with refined appointment times, that conserve the initial service order of patients. We plan to relax this assumption and consider fixed bounds on the appointment times changes in our future research.

3.1.3 Literature Review

Comprehensive overviews of the typical process flow at a cancer center can be found in [Singprasong and Eldabi, 2013] and [Dohse, 2007]. In [Woodall et al., 2013], a discrete event tool is used to predict patient wait times at each step of their visit to a cancer center; the authors then present recommendations on nurse staffing based on results obtained through an optimization model. In [Santibáñez et al., 2012], flow mapping and various operations research techniques, such as a discrete-event simulation model and an optimization-based scheduling tool, are used to significantly reduce the size of the patient

wait list. In order to simplify the scheduling process a common approach is to schedule the phlebotomy and clinic visit one day and the infusion on the following day. Pros and cons of this *"next-day"* model are discussed in [Dobish, 2003]; one major drawback of this approach is that patients have to visit the cancer center twice for each infusion. While this might be acceptable in some cases, this is not the norm in practice, which motivates our approach to improve service quality by simply optimizing appointment times, without changing the process.

Chemotherapy drugs are expensive and a common practice is to only start mixing a dose once the patient is ready in the infusion chair so as to avoid drug waste in case of patient deferral. This potentially results in additional patient wait times and contributes to the uncertainty in total treatment times that we observed in our historical data analysis; this challenge is discussed in various studies such as [Mazier et al., 2010], [Masselink et al., 2012] and [Aboumater et al., 2008].

There is a significant body of literature on appointment scheduling in healthcare systems. See [Gupta and Denton, 2008] for an extensive survey of applications of simulation, queuing, and optimization methods to appointment scheduling. Most appointment scheduling models in healthcare focus on one of the two following stages: the day of visit or the time of the appointment.

Chemotherapy treatment plans typically consist of several visits. In [Turkcan et al., 2012], the problem of scheduling patient visit days to balance the workload is considered. On top of patient wait time and staff overtime, the authors also aim to reduce treatment delays and to maximize staff utilization. In [Sevinc et al., 2013], a two-stage model is developed to optimize days of appointment then the assignment of patient to an infusion chair. This second phase specifically involves a heuristic based on the knapsack problem, which is similar to the *First Chair Available* greedy approach we propose in our work. In [Ahmed et al., 2011] a scheduling template is created to improve and simplify the scheduling process, evaluation through simulation showed a potential to increase patient throughput by over 20%.

Numerous articles deal with appointment time scheduling, generally minimizing a trade-off between patient wait times and total length of operations (or staff overtime or idle time). However only a few articles specifically consider a chemotherapy environment. [Sadki et al., 2011] considers clinic appointment times and bed or infusion chair availability. The authors propose a Lagrangian relaxation-based heuristic to minimize a weighted sum of expected patient wait time and makespan, which is exactly the objective we consider in our work. Simulation is used in [Cayirli et al., 2006] to evaluate various scheduling policies when setting patient appointment times in the context of ambulatory care visits. In

this paper, patients are divided into different groups, which allows schedulers to have more information and better predict the expected length of the visit which is similar to our classification of patients into types, each having a specific treatment times distribution.

Variability in treatment times further complicates the scheduling process. Very few studies consider uncertainty in infusion times in a chemotherapy context. In addition to [Turkcan et al., 2012], already cited above, variability is also considered in a thesis [Tanaka, 2012] where heuristics based on the bin-packing problem are proposed. Scheduling under uncertainty is more frequently applied in the context of surgery and operating room scheduling [Denton et al., 2007], [Denton et al., 2010] and [Min and Yih, 2010].

This paper differs from the existing literature in the following ways:

- Our framework not only consider location availability (infusion chair) but also an external resource (nurse) when scheduling patients to infusion,
- We refine an existing appointment schedule instead of creating one from scratch,
- We propose a novel heuristic algorithm to solve appointment scheduling type of problems under uncertainty.

3.1.4 Contributions and Outline of the Paper

We formulate the schedule refinement problem as a two-stage stochastic integer program. The objective is to minimize a weighted combination of expected patient wait times and expected staff overtime across a large set of scenarios obtained by sampling from patient treatment length distributions. Given the computational complexity required to solve the resulting large-scale mixed integer program exactly, we propose a fast heuristic algorithm, which we evaluate using lower bounds on the optimal solution.

The main contributions of our work are:

1. Studying important dynamics of the process flow at an infusion cancer center to formulate a new *Schedule Refinement Optimization Problem* (SROP) under uncertainty of preparation times and treatment times,
2. Developing an efficient heuristic to quickly obtain good approximations of this challenging problem, and designing methods to compute lower bounds on the optimal objective value to evaluate the quality of the heuristic solutions.
3. Using a parametric approach to generate different Pareto efficient solution schedules to add flexibility and fit the preferences of any cancer center regarding the trade-off between patients waiting times and staff idle time.

4. Drawing managerial insights based on the results from our model. We show that allowing more time between successive appointments in the middle of the day rather than appointments in the morning or late afternoon allows us to significantly reduce expected patient wait time at a very low cost in staff overtime.

The remainder of the chapter is organized as follows: In Section 3.2, we motivate and describe the stochastic programming formulation of *SROP* and show that computing exact solutions requires a prohibitive amount of time.In Section 3.3, we develop a heuristic algorithm to quickly generate solutions which are evaluated through computation of bounds on the optimal objective value. In Section 3.4, we present a case study of an application of our approach at *UMCCC*, and propose recommendations to better schedule patient appointment times. In Section 3.5, we conclude with extensions and ideas for possible future research.

3.2 The Schedule Refinement Optimization Problem

In this section, we formulate *SROP*. We first describe the problem (inputs, decisions, constraints, and objective), then we present a stochastic programming formulation of the problem before analyzing its tractability.

3.2.1 Problem Description

SROP can be modeled as a two-stage stochastic integer program with continuous first stage variables for patient appointment times and binary and continuous second stage variables representing what happens in each scenario, given the appointment times decided in the first stage (which chair each patient goes to, waiting times, times of discharge and total length of operations).

We use the *sample average approximation* framework to model uncertainty, thus, we sample a finite number of scenarios for chemotherapy infusion times. Each scenario consists of a realization of a treatment time for each patient to be scheduled. Those realizations are drawn independently from the distributions of treatment time of each patient type, as described in Section 3.1.2. Extensive studies of this approach can be found in the engineering literature: [Wang and Ahmed, 2008] provides a general overview of the method applied to stochastic linear programs while [Kleywegt et al., 2002] and [Ahmed et al., 2002] specifically study applications to integer programming. In [Mancilla and Storer, 2012] the sample average approximation method is used in the context of appointment scheduling.

In our problem, scenarios are defined as a realization of preparation time and treatment time for each patient. We construct each scenario using the following process: first, we sample from the appointment length distributions to obtain the total time each patient spends in the infusion chair (preparation time with the nurse plus infusion length or treatment time) — note that this does not include wait times, since wait time is an output of our model, not an input. In order to obtain separate values for the preparation and treatment times we use the following procedure:

1. Preparation by the nurse: Time during which a nurse brings a patient to an available infusion chair and prepares the patient to receive his/her drug. Based on expert opinion and our observations at the *UMCCC*, we assume that preparation time follows a uniform distribution between 0 and 30 minutes for all patients. Even though a preparation time of 0 minute seems unrealistic, this range is used to model the variability in preparation time: some patients might come in almost ready to receive their infusion while others require extensive care during the initial set-up. The only exception is the extremely rare event when the actual total length of the patient visit is less than or equal to 30 minutes for which we then assume that the preparation time follows a uniform distribution between 0 and the visit length. For instance, consider the case of a patient who spent a total of 25 minutes in a infusion chair. The preparation time by the nurse must have been less than 25 minutes, so we then assume that this preparation time followed a uniform distribution between 0 and 25 minutes.

2. Treatment: Time during which the patient receives his/her drug before being discharged. We define treatment time as the remaining time: total visit length minus preparation time.

The inputs to our model are: (1) A sequence of patients to be scheduled. Recall that we assume that this sequence has to be preserved (Section 3.1.2), (2) a set of infusion chairs supervised by one nurse and, (3) a list of scenarios, each containing a realization of preparation and treatment times for each patient.

The main decision variables are the appointment times for patients. These appointment times are first stage decisions since they have to be decided before realization of the uncertainty. We assign patients to chair in the second stage. In reality, the assignment decision is dynamic and patients are assigned to a chair one by one, after realization of treatment duration of the previous patients. In our model, the decision is made in each scenario after realization of all treatment times for the day, which is not possible in reality. However this approach is valid since the optimal chair assignment for a given only depends on the

treatment time of the patients before him, as described by the first chair available routine (see proof in the Appendix).

3.2.2 Notation and Stochastic Optimization Formulation

We now formally define all variables and parameters of the model:

Sets:

P: sequence of patients for one day for the set of infusion chairs considered. Patient $p+1$ has to be seen after patient p.

C: set of chairs

Ω: set of scenarios considered

Parameters:

s_p^ω: preparation time of patient p in scenario ω

t_p^ω: infusion length of patient p in scenario ω

m: number of scenarios

$\lambda \in [0,1]$: trade-off parameter in the objective function. λ is the weight assigned to patient wait time, while $1-\lambda$ is the weight associated with the total length of operations.

M: large number

Decision Variables:

- First stage:

  a_p: Appointment time of patient p

- Second stage:

  x_{pc}^ω: 1 if patient p is treated in chair c in scenario ω; 0 otherwise

  w_p^ω: Waiting time of patient p in scenario ω at waiting area

  d_p^ω: Discharge time of patient p in scenario ω

  L^ω: Length of operations in scenario ω

Figure 3.2 illustrates the different time stamps of the visit of a given patient p under a given scenario ω. When this patient p arrives in the waiting room — at his/her appointment time a_p — he/she has to wait for the nurse and an available infusion chair for a waiting duration of w_p^ω, then the patient goes to a chair and is prepared by the nurse for a duration of s_p^ω, then the infusion begins and takes time t_p^ω. When it is completed, the patient is discharged at time d_p^ω.

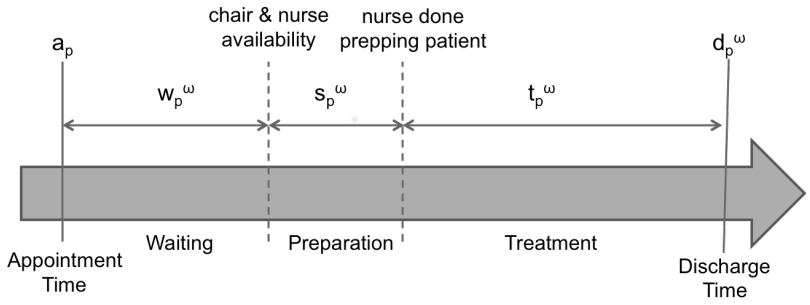


Figure 3.2: Patient Time Line

Formulation:

$(SROP)$

$$\min \frac{1}{m}\left(\lambda \sum_{p\in P}\sum_{\omega\in\Omega} w_p^\omega + (1-\lambda)\sum_{\omega\in\Omega} L^\omega\right) \tag{3.1}$$

Subject to:

$$\sum_{c\in C} x_{pc}^\omega = 1 \qquad \forall p\in P, \omega\in\Omega \qquad (2)$$

$$a_p + w_p^\omega + s_p^\omega + t_p^\omega = d_p^\omega \qquad \forall p\in P, \omega\in\Omega \qquad (3)$$

$$a_{p_j} + w_{p_j}^\omega + M(2 - x_{p_jc}^\omega - x_{p_ic}^\omega) \geq d_{p_i}^\omega \qquad \forall c\in C, p_j > p_i \in P, \omega\in\Omega \qquad (4)$$

$$a_{p+1} + w_{p+1}^\omega \geq a_p + w_p^\omega + s_p^\omega \qquad \forall p\in P - \{n\}, \omega\in\Omega \qquad (5)$$

$$L^\omega \geq d_p^\omega \qquad \forall p\in P, \omega\in\Omega \qquad (6)$$

$$x_{pc}^\omega \in \{0,1\} \qquad \forall c\in C, p\in P, \omega\in\Omega \qquad (7)$$

$$a_p \geq 0 \qquad \forall p\in P \qquad (8)$$

$$w_p^\omega, d_p^\omega \geq 0 \qquad \forall p\in P, \omega\in\Omega \qquad (9)$$

The objective function (1) is to minimize a linear combination of the total expected waiting time and the expected total length of operations. Since scenarios are obtained via uniform sampling on the distributions, they all happen with the same probability $1/m$ and expected values reduce to the average.

Constraint (2) represents the assignment of each patient to exactly one infusion chair in each scenario. (3) defines the value of the discharge time of patient p in scenario ω. Discharge time d_p^ω is equal to the arrival time a_p plus the waiting time w_p^ω plus the preparation length s_p^ω plus the infusion length t_p^ω. Constraint (4) is the "available chair constraint" — A patient can sit in a chair only if every previously sequenced patients *assigned to this chair* has been discharged. Consider a scenario ω and a patient p_j sequenced later than a patient p_i (not necessarily right after):

- If the two patients are not assigned to the same chair then $2 - x^{\omega}_{p_j c} - x^{\omega}_{p_i c} \geq 1$ and the constraint is relaxed as long as the constant M is chosen suitably large.
- If the two patients are assigned to the same chair c in this scenario ($x^{\omega}_{p_i c} = x^{\omega}_{p_j c}$) then the constraint reduces to $a_{p_j} + w^{\omega}_{p_j} \geq d^{\omega}_{p_i}$ which means that patient p_j cannot sit before patient p_i is discharged.

Constraint (5) is the "available nurse constraint" — A patient can sit in a chair if the nurse has finished preparing the previous patient in the sequence, *not necessarily assigned to the same chair*. Recall that we assume that one nurse only is working on this pod of $|C|$ chairs. Consider a patient p, the end of his/her preparation time is defined as $a_p + w^{\omega}_p + s^{\omega}_p$ (appointment time plus wait time plus preparation length). Only then can the nurse prepare the following patient, indexed as patient $p + 1$. Finally, constraint (6) defines the value of the total length of operations in each scenario. Since it has to be minimized, L^{ω} will be set to the maximum discharge time in scenario ω which corresponds to the discharge of the last patient.

3.2.3 Run Time and Computational Performance

In this section, we assess the tractability of $(SROP)$. We measure the sensitivity of computation time to the number of scenarios m. Even though computation time also depends on other parameters such as the number of patients, the number of chairs and the trade-off weight λ, the dependency on m is the most important since the Sample Average Approximation requires us to consider a large number of scenarios to obtain accurate approximations of the solution to the full problem containing all possible scenarios.

We solved instances with 12 patients and 3 chairs, a trade-off parameter $\lambda = 0.3$ (we postpone the discussion of the choice of this value to Section 4) and an suitably large value of $M = 10000$.

For each choice of m we solve 10 instances, each of which is based on the random generation of m scenarios by sampling from the historical data presented in Section 3.1.3. The optimality gap termination criterion is set to the default value of 10^{-6} (larger values of the optimality gap are discussed later). We report the median computational time in Table 3.1, under the "original model" column. We were unable to solve instances with $m > 10$ scenarios in less than an hour. All computational experiments were run using an Intel Xeon E3-1230 quad-core running at 3.20 GHz with hyper-threading and 32 GB of RAM. We used IBM ILOG Optimization Studio (*CPLEX*) 12.6 C++ API software package.

The large value of M parameter in constraint (4) causes linear relaxations of the problem to be very weak. Imagine that one or both of the variables $x^{\omega}_{p_1 c}$ and $x^{\omega}_{p_2 c}$ are fractional

when solving a relaxation of the problem, then the constraint is loose and the waiting time $w^{\omega}_{p_2}$ is not constrained so the actual patient wait time is drastically under-estimated in those relaxations. In order to improve the run time of the model, we studied three areas of improvement:

1. We arbitrarily set the chair assignment of the first 3 patients (when 3 chairs are considered) to break some symmetry of the problem and reduce the number of binary variables: patient 1 to chair 1, patient 2 to chair 2 and patient 3 to chair 3, in all scenarios.

2. We add a set of constraints giving lower bounds on the length of operations in each scenario to tighten the formulation. In scenario ω, patient p occupies an infusion chair for a time equal to $s^{\omega}_p + t^{\omega}_p$ (preparation plus treatment time). Since only 3 chairs are available, the total completion time cannot be less than $\frac{1}{3}\sum_{p\in P} s^{\omega}_p + t^{\omega}_p$. We add the following constraints:

$$L^{\omega} \geq \frac{1}{3}\sum_{p\in P} s^{\omega}_p + t^{\omega}_p \quad \forall \omega \in \Omega$$

3. Finally, we address the issue of weak relaxations due to the big-M parameters. In order to mitigate this problem, one can set M to the smallest possible value that still guarantees the inequality to be valid. Analysis of constraint (3) shows that it is sufficient to have $M = t^{\omega}_{p_i}$ for each patient i and scenario ω.

Table 3.1 contains run times obtained using this improved model. We also notice that a significant part of the computational effort is spent on final branching steps, only yielding minor improvement on the objective value. Closing the duality gap to find the true optimal solution might not be worth the additional computational time in the context of patient scheduling, so we also present computational times obtained when stopping the optimization as soon as an optimality gap of 1% is reached. As expected, the improved formulation outperforms the original model, and run times are even lower when we only look for an approximate solution. However, it appears that run times still increase exponentially with the number of scenarios considered, and even after improvement of the model, solving an instance with as few as 10 scenarios already requires a significant computational effort. Given those preliminary results, it is clear that a direct approach to solving this model is not viable for a large number of scenarios. We thus propose to use the special structure of this scheduling problem to design a heuristic approach.

Table 3.1: Comparison of Run Times (in seconds)

Number of Scenarios	**Original Model**	**Improved Model (10^{-6} Opt. Gap)**	**Improved Model (10^{-2} Opt. Gap)**
1	0.2	0.3	0.2
2	0.7	0.5	0.5
3	2.5	1.6	1.2
4	12.3	3.4	3.0
5	35.4	7.1	6.5
6	73.6	26.3	15.3
7	157.0	76.5	40.2
8	685	153.3	113.2
9	>3600	549.2	457.6
10	>3600	>3600	>3600

3.3 A Fast Heuristic

Our heuristic approach is motivated by the following two facts: First, suppose that the appointment times a_p are known (first stage decisions). Then the model can be solved independently for each scenario since we don't have any first stage decision variables linking the scenarios together. Within a scenario, the problem reduces to assigning patients to chairs to minimize wait time and length of operations, with knowledge of their arrival time. This is an easy problem that can be solved to optimality in linear time by the *First Chair Available* greedy sub-routine presented in Algorithm 1. We call this problem $FCA(A)$ since it depends on a set of appointment times $A = \{a_p : \ p \in P\}$.

Second, suppose that the assignment of patients to chairs x^{ω}_{pc} is known for each scenario. Then by substituting those values in place of the binary variables we reduce $(SROP)$ to a pure linear programming model containing only continuous decision variables, and therefore solvable in polynomial time. We call this reduced problem $LP(X)$ since it depends on a chair assignment $X = \{x^{\omega}_{pc} : \ p \in P, \ c \in C, \ \omega \in \Omega\}$.

The key idea of our heuristic is to start with all patients scheduled at time 0 ($a_p = 0 \ \forall p \in P$), then solve $FCA(A)$ to obtain a chair assignment X. At this point, we alternate

between solving $FCA(A)$ and $LP(X)$ to obtain progressively better solutions; note that this can be done quickly since both sub-problems are easy. Since this heuristic alternates between solving sub-problems, each of which only contains a subset of the original decision variables, we call it the *Fix-Unfix* algorithm.

The heuristic is illustrated in Figure 3.3 and proceeds as follows:

- **Initialization:** Start by setting the appointment times A_0 to 0 for all patients, i.e., they all arrive at the beginning of the day, and construct the first available chair assignment X_0 by solving sub-problem $FCA(A_0)$. Note that we now use subscripts within the A_i and X_i notations (previously referred to as A and X) to denote the current iteration number.

- **Iteration:**

  An iteration i starts at a state (A_i, X_i).

  1. Solve the linear program $LP(X_i)$ to get new appointment times A_{i+1}, which are optimal with respect to chair assignment X_i.
  2. Use those appointment times to create the next first available chair assignment X_{i+1} by solving sub-problem $FCA(A_{i+1})$.

- **Termination criterion:** When we obtain a chair assignment, X_k, that we already visited in a prior iteration, we terminate and return the current pair (A_k, X_k). Note that the algorithm always terminates since there only exists a finite number of chair assignments.

Note also that the current objective value (combination of expected wait times and expected length of operations) can only decrease or stay the same during an iteration. Consider an iteration i. In step 1, we optimize the appointment time so the objective value of the pair (A_{i+1}, X_i) is lower or equal than the one of the pair (A_i, X_i). In step 2, we create the first available chair assignment X_{i+1} which is optimal with respect to the appointment times A_{i+1} so the pair (A_{i+1}, X_{i+1}) has a lower (or equal) objective value than the pair (A_{i+1}, X_i). By transitivity, pair (A_{i+1}, X_{i+1}) is at least as good as pair (A_i, X_i).

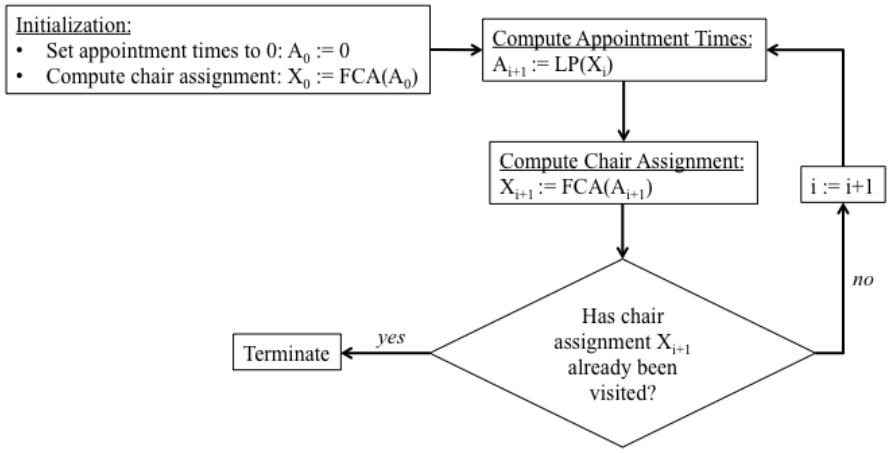


Figure 3.3: Representation of the algorithm

We now present the *First Chair Available* sub-routine that is used to solve problem $FCA(A)$. Given a set of appointment times A, we want to optimally assign patients to chairs in each scenario and output an optimal chair assignment X with respect to the appointment times A. The *First Chair Available* sub-routine sets the value of the decision variables $X = \{x_{pc}^{\omega} : \ p \in P, \ c \in C, \ \omega \in \Omega\}$. Recall that $x_{pc}^{\omega} = 1$ if patient p is assigned to chair c in scenario ω, and 0 otherwise. Since chairs are identical, there is no reason to delay a patient's treatment when a chair becomes available so it is optimal to assign patients in order of their arrival to the first available chair in each scenario. For now, we state this result in Proposition 1 below and we provide a formal proof in the Appendix.

Proposition 1. *The First Chair Available algorithm provides an optimal chair assignment with respect to the objective considered in SROP, for a given set of patient appointment times.*

This subroutine is presented in Algorithm 1.

3.3.1 Computational Performance of the Fix-Unfix Algorithm

We repeat the experiments from Section 3.2.3 to evaluate the computation time required to approximately solve an instance of *SROP* using the *Fix-Unfix* heuristic. For each value of the number of scenarios m, we applied the heuristic algorithm to 10 randomly generated instances (with 12 patients, 3 chairs and $\lambda = 0.3$) and we report the median run time in Figure 3.4. Because the algorithm is much faster than solving directly the Mixed-Integer Programming model, we are able to go much further than instances with 10 scenarios. The

$FCA(A)$: **Data:** Appointment times $A = \{a_p : \ p \in P\}$
Result: Chair assignment $X = \{x^{\omega}_{pc} : \ p \in P, \ c \in C, \ \omega \in \Omega\}$
for *scenario* $\omega \in \Omega$ **do**
  Initialize available time of chairs to 0: $T_{avail}(c) = 0 \ \forall c \in C$;
  for *patient* $p \in P$ *(in order of arrival)* **do**
    Find a chair $c_0 \in C$ with smallest available time $T_{avail}(c_0)$:
     $c_0 = argmin\{T_{avail}(c) : \ c \in C\}$;
    Set $x^{\omega}_{pc_0} = 1$ and $x^{\omega}_{pc} = 0 \ \forall c \neq c_0$;
    Update available time of chair c_0: $T_{avail}(c_0) = T_{avail}(c_0) + s^{\omega}_p + t^{\omega}_p$;
  end
end
return Chair assignment X;

Algorithm 1: First Chair Assignment Algorithm

dependency of run times on the number of scenarios is roughly linear, and solving large instances of the problem can be done in only a few seconds. The number of iterations in each run of the algorithm appeared to be independent of the number of scenarios and was always between 4 and 7. Note that the run times are substantially faster than solving the extensive form of the stochastic program (3.1). Our heuristic approach, applied to instances containing 1000 scenarios, ran in just a few seconds.

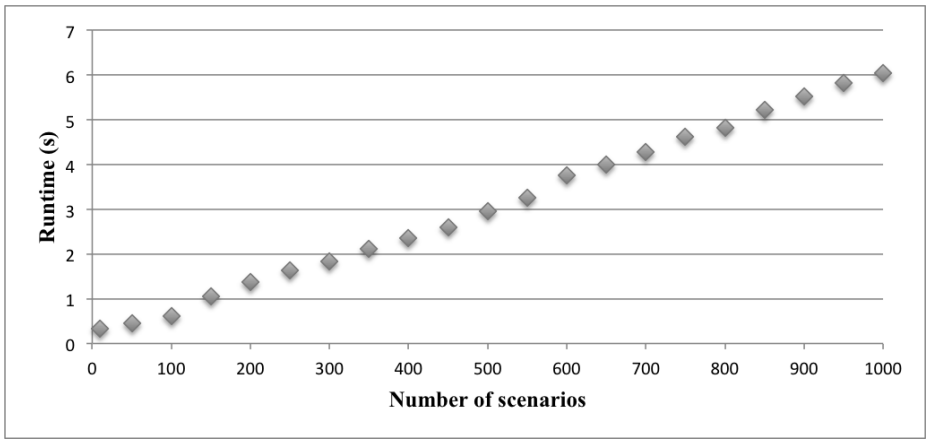


Figure 3.4: Run Times with Heuristic

3.3.2 Computation of Lower Bounds on the Optimal Solution

The *Fix-Unfix* algorithm has two main features: (1) the objective value can only decrease throughout the iterations and (2) it is guaranteed to terminate. However it does not have a guarantee of global optimality (see the Appendix for a counter-example). Therefore, in this section, we describe a way to obtain lower bounds on the optimal objective value of *SROP*,

which can then be used to evaluate the quality of the solutions returned by the *Fix-Unfix* heuristic.

The *SROP* problem can be written as:

$$\min \left\{ \frac{1}{m} \sum_{\omega \in \Omega} f_\omega(a) \mid \text{ s.t. } a \in A \right\} \quad (7)$$

where a is the vector of appointment times, $f_\omega(a)$ is the weighted combination of wait times and total length of operation obtained in scenario ω when using schedule a, and A is the set of valid schedules respecting the sequence of patients, that is:

$$A = \{a \in R^n, a_{p+1} \geq a_p\}$$

To obtain lower bounds on (7), we first consider the case where the variable a (patient appointment times) can take different values in different scenarios. In other words, we relax the *non-anticipativity* constraints. Obviously, the resulting solutions are not practical since, in reality, there is no way to know in advance the realization of preparation and treatment times (i.e., the scenario ω). However, allowing variable a to vary by ω effectively partitions the problem into independent sub-problems containing only one scenario which can be solved much faster than the entire formulation. This approach yields the well-known *wait-and-see* lower bound, see chapter 4 of [Birge and Louveaux, 2011b]. This bound is valid since we are solving a relaxation of the original problem.

Clearly, this will not yield a strong lower bound in this case. It is straightforward to see that the waiting time part of the objective will always be 0: in each scenario, since we know how long each treatment is, we can schedule such that patients never have to wait. The total length of operations is also minimized since there is no idle time in each scenario.

Treating each scenario independently yields a weak lower bound, but motivates an intermediate approach which we call *partial relaxation of the non-anticipativity constraints*. It consists of partitioning the problem into groups of scenarios and allowing the first stage decision variables a to take a different value for each group.

We choose a group size m' and we randomly create a partition of the set of scenarios Ω in groups $G_1, \ldots, G_k$ each of size m' where $k = m/m'$. In the case where m/m' is not integer, we simply create some groups with size $m' - 1$ such that the groups form a valid partition of Ω:

$$G_1 \cup \ldots \cup G_k = \Omega \text{ and } G_i \cap G_j = \emptyset \; \forall i \neq j$$

We define the new lower bound as the sum of optimal objective values of the sub-problems defined by the groups G_i, as follows:

$$\frac{1}{m} \sum_{1 \leq i \leq k} \min \sum_{\omega \in G_i} f_\omega(a_{G_i}) \text{ s.t. } a_{G_i} \in A$$

Computing this bound necessitates solving k instances of ***SROP***, each with approximately m/k scenarios, which is faster than solving the original instance of ***SROP*** which contains m scenarios. There is a trade-off between the quality of the bound and the computational effort needed to compute it: if k is close to m we can expect a tight bound on the objective but the computational time to obtain it will be similar to the time needed to solve the original problem. On the contrary, low values of k leads to weaker bounds, but are easier to compute.

3.3.3 Comparison of Heuristic Objective to Lower Bounds Values

Comparison of Lower Bound Strength: We now compare the lower bounds obtained for different values of the group size, for an instance with 12 patients, 3 chairs and for a trade-off parameter $\lambda = 0.3$ and 100 scenarios. We also compute the lower bound obtained when solving the continuous relaxation of the ***SROP*** formulation obtained by relaxing the integrality of the chair assignment variables. Table 3.2 contains the objective value of the feasible solution returned by the ***Fix-Unfix*** algorithm, and for each bounding method, the value of the lower bound, the computational time and the heuristic-to-bound gap, defined as: $\frac{\text{heuristic objective} - \text{bound}}{\text{heuristic objective}}$, which is an upper bound on the performance ratio of the heuristic.

As expected, the continuous relaxation of the objective provides a weak lower bound (see Section 3.3.2). The bounds based on the relaxation of the non-anticipativity constraints lead to tighter values of the heuristic-to-bound gap, at the expense of a higher computational effort as the group size increases. With a group size of 6, we reach a heuristic-to-bound gap of roughly 3%.

Table 3.2: Comparison of lower bounds on one instance of *SROP* with 100 scenarios

		Objective Value or Bound	Run Time (s)	Heuristic-to-Bound Gap
Heuristic		529.4	1.0	N/A
SROP Relaxation		404.6	1.2	23.6%
Partial Scenario Decomposition	**Group Size: 1**	463.0	31	12.5%
	Group Size: 2	489.1	28	7.6%
	Group Size: 3	499.0	53	5.8%
	Group Size: 4	505.7	97.2	4.5%
	Group Size: 5	510.7	250.6	3.5%
	Group Size: 6	514.3	1051	2.9%

Sensitivity of lower bounds: Since we are ultimately interested in solving *SROP* for different values of the trade-off parameter λ, we now study the sensitivity of these bounds with parameter λ. We use the partial relaxation approach with group size 6 to evaluate the performance ratio of the heuristic for different values of the trade-off parameter λ. Results are presented in Table 3.3.

- When λ is 0, it is clearly optimal to schedule all patients at time 0, which achieves the minimal length of operations in each scenario. The optimal objective value is simply the average of those minimal length of operations over all scenarios. In this case the heuristic is optimal since it schedules all patients at time 0 in its initialization step and terminates immediately. The lower bound approach also finds the optimal objective value since it schedules all patients at time 0 for all groups of scenario, therefore reaching the same objective value. This explains the value of 0% for the heuristic-to-bound gap.

- When λ is 1, the problem is to minimize patient wait time, ignoring the total length of operations. Since it is always possible to have 0 wait time by scheduling patient far apart from one another, the optimal objective value is 0. In this case the heuristic again finds the optimal solution and the bound is tight.

- For other values of λ the partial relaxation approach with group size 6 allows us to obtain good performance ratios for the *Fix-Unfix* heuristic, typically less than 5% for most values of λ, the worst case being 11.2% for $\lambda = 0.9$. Although the relative heuristic-to-bound gap increases with λ it is not representative of the absolute performance of the heuristic since 1 minute of extra wait time represents a much higher

percentage when λ is 0.9 than when it is 0.1. Also note that a 10% heuristic-to-bound gap may not mean that the heuristic solution is far from optimal. The heuristic-to-bound gap is a worst-case scenario, and that the true difference between the heuristic and optimal objective values may be smaller (and in fact could even be zero).

Table 3.3: Optimality Gap for different values of the trade-off parameter λ

Parameter λ	0	0.1	0.2	0.3	0.4	0.5
Heuristic-to-Bound Gap	0%	1.9%	2.2%	2.9%	3.8%	4.1
Objective Value From Heuristic	649	623	574	517	455	388
Parameter λ	0.6	0.7	0.8	0.9	1	
Heuristic-to-Bound Gap	4.7%	5.3%	6.9%	11.2%	0%	
Objective Value From Heuristic	317	242	165	87	0	

Performance under a general distribution class: Finally, we study the performance of the heuristic for a random set of test instances under the normal distribution for patient treatment times, as an alternative to the distributions based on real patient visits that we have been using so far. We chose the normal distribution because it is commonly used in practice and because it provides a reasonable fit to empirical data (See Figure 3.1). We ran the following experiment: In each trial, we solved an instance of *SROP* with the *Fix-Unfix* heuristic and computed a bound using the partial decomposition approach with a group size of 6, with a trade-off parameter λ equal to 0.3. To create an instance, we first generated a mean and a standard deviation for the infusion length of each patient, assumed to follow a normal distribution with those parameters. The means are sampled form a uniform distribution between 0 and 600 minutes, while the standard deviations are sampled form a uniform distribution between 0 and 100 minutes. For this experiment, standard deviations are generated independently from the mean of their distribution. This represents the fact short or long treatment protocols might be equally likely to have low or high variability. We then created 100 scenarios by sampling as we did in previous experiments, making sure that, if a negative value is sampled, we set it to 0. Note that we assumed that the preparation time still follows a uniform distribution between 0 and 30 minutes. We recorded the heuristic-to-bound gap obtained in 100 trials. To speed up the process of computing the bounds, we only solve the mixed-integer sub-problems to a 1% optimality gap and returned the best current lower bound. Over these 100 trials, the minimum heuristic-to-bound

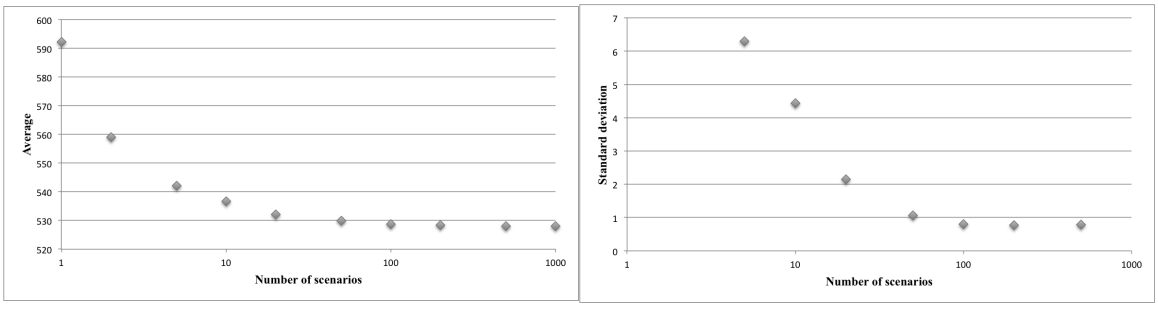
gap was 2.2%, the maximum was 5.4% and the average was 3.4%. This suggests that the heuristic performs well for a broad range of test instances.

3.4 Case Study: Application of *SROP*

The goal of this section is to compare the characteristics and the quality of the schedules obtained with the *Fix-Unfix* algorithm to other traditional scheduling approaches used at outpatient infusion centers, such as *UMCCC*.

3.4.1 Study of Sample Size

When solving exactly a Sample Average Approximation version of a stochastic optimization problem, the theory indicates that the optimal objective value converges exponentially fast to the optimal objective value of the full problem as the number of scenarios increases [Wang and Ahmed, 2008],[Kleywegt et al., 2002]. However, we are only computing an approximation to the sample problem, not an optimal solution, and we do not seek to develop definitive theory on convergence relative to our heuristic, but rather to anecdotally explore the impact of number of scenarios on solution quality for our test problem instance. We therefore explore the impact of the number of scenarios on the objective value of the solution for a given sequence of patients. We performed the following experiment: For each value of m (the number of scenarios included in the approximation), we generated and solved 100 instances (again with 12 patients, 3 chairs and $\lambda = 0.3$) using the *Fix-Unfix* algorithm. For each of these solutions, i.e., sets of appointment times, we then evaluated them via a much larger simulation, i.e., considering 10,000 scenarios.



(a) Average of Simulated Objectives (b) Standard Deviation of Simulated Objectives

Figure 3.5: Average and Standard Deviation of Simulated Objectives of Solutions from Heuristic

We report the average and standard deviation of the 100 simulated objectives on Figure 3.5a and 3.5b, using a logarithmic scale on the x-axis for clarity. From the figures, we observed that both the average and standard deviation decreased roughly exponentially up to a certain point, then reached a plateau after 100 scenarios. This suggests that considering 100 scenarios is sufficient to achieve accurate solutions. Therefore, for the remainder of this section, we use 100 scenarios.

3.4.2 Evaluating the Benefits of Schedule Refinement

We begin by selecting a value of the trade-off parameter λ. Rather than using a single arbitrary value of λ, we generate refined schedules with different values of λ. Recall that values of λ close to 1 put more emphasis on patient wait times while values of λ close to 0 favor total length of operations. For each candidate value of λ, we solved *SROP* with 100 scenarios.

Figure 3.6 illustrates the performance of the schedules obtained with λ varying from 0.05 to 0.75 by increments of 0.05. As expected, we observe a natural trade-off between total length of operations and total wait time. Intuitively, schedules achieving low patient wait times have more built-in buffer between successive patients which might cause unused resources and ultimately a higher total length of operations. On the contrary, reaching a lower total length of operations necessitates scheduling shorter times between appointments in order to maximize resource utilization, which leads to longer wait times.

Now, we compare refined schedules to the initial schedule obtained when scheduling patients according to their scheduled appointment length found in the data set presented in Section 3.1.1. We observe, still on Figure 3.6, that a few optimized schedules ($\lambda = 0.25, 0.30, 0.53, 0.40$), strictly dominate the initial schedule with respect to the two considered metrics. In particular, using the *Fix-Unfix* algorithm on this instance would reduce the total expected patient wait time by more than an hour without increasing the total length of operations, or reduce the expected duration of operations by more than 30 minutes for a similar level of waiting times. Another benefit of our approach compared to the existing approach is that generating a candidate set of schedules allows more flexibility in allowing the infusion center to enforce their preferences when picking one schedule for the day.

For comparison purposes, we also include in Figure 3.6 the schedules obtained when using a simple scheduling rule that has been previously proposed and referred to as "job hedging" [Gul et al., 2011]. The heuristic computes the mean of appointment duration, then schedules the duration of the appointment for a time equal to the mean adjusted by a

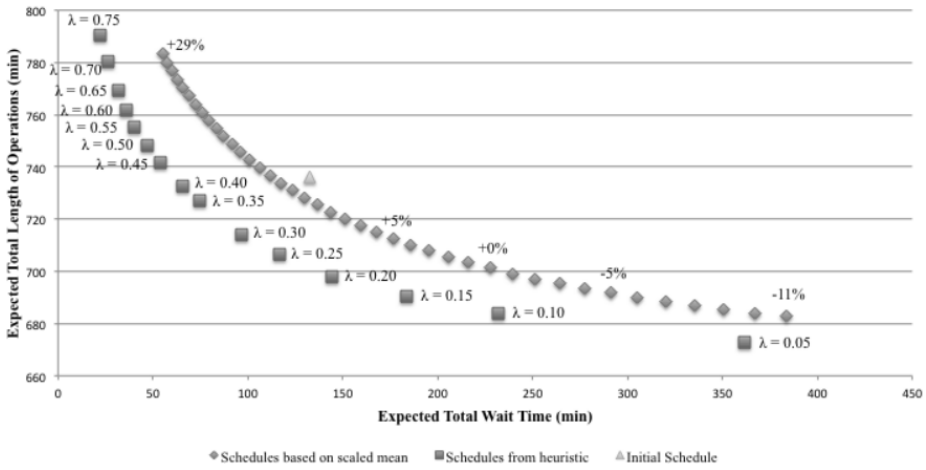


Figure 3.6: Comparison of initial schedule and schedules from optimization in a Length of Operations / Wait Time chart

scale factor. A scale factor of 0 corresponds to scheduling according to the mean, while scale factor of 10% (respectively -10%) corresponds to scheduling 10% over (respectively 10% under) the mean. We observe that the optimization-based schedules from the *Fix-Unfix* algorithm also dominate the schedules obtained with this simpler scheduling rule.

Analysis of a Specific Schedule: We now study in depth a schedule obtained when using the *Fix-Unfix* algorithm with a specific value of the trade-off parameter λ. Looking at Figure 3.6 we pick $\lambda = 0.3$ because this is one of the schedules that dominates the initial schedule. We compare the performance of this schedule to the initial schedule.

First, we observe in Table 3.4 that in the refined schedule, no patient is scheduled more than one hour apart from his/her original appointment time. The largest change in appointment time is 51 minutes in this case. Moreover, half of the patients have only minor changes, lower than 10 minutes.

Second, we compare the expected total length of operations and the expected wait time per patient as well as its standard deviation for each patient across the scenarios considered. The expected total length of operations is 736 minutes with the initial schedule and 711 minutes for the refined schedule. This means that using the refined schedule would lead in average to a reduction of the total length of operations of 25 minutes, which could decrease staff overtime or allow for adding in an additional patient to the day's schedule.

The total expected wait time is 134 minutes for the initial schedule and 102 minutes with the refined schedule. Table 3.5 contains the average and the standard deviation of wait time for each patient in both schedules. We note that the refined schedule decreases wait

Table 3.4: Appointment times in initial and refined schedules

Patient	1	2	3	4	5	6	7	8	9	10	11	12
Initial Scheduled Appointment Length (min)	180	30	150	300	180	60	90	420	60	150	90	90
Appointment Times in Initial Schedule (min)	0	15	30	45	180	195	255	345	360	375	420	510
Appointment Times in Refined Schedule (min)	0	14	25	52	143	203	264	294	363	400	445	536
Change in Appointment Time (min)	0	-1	-5	7	-37	8	9	-51	3	25	25	26

times of patients with long wait times in the initial schedule (patients 4,6,7,9,10,11 and 12). However patients with the lowest wait times in the initial schedule tend to wait more in the refined schedule (patients 2,3,5 and 8). As a result, wait times in the refined schedule are (1) lower overall, and (2) better distributed between patients. We observe a similar trend for the standard deviations: reduction (resp. increase) of the variability for patients with a large (resp. low) standard deviation in the initial schedule. In the context of waiting time, a moderate variability for all patients is arguably better than no variability for half of the patients and a high variability for the other half.

For this value of the trade-off parameter λ, the refined schedule not only outperforms the initial schedule for the two metrics that are considered in the optimization model (expected total length of operations and expected total wait time), but also has some additional desirable features, such as wait times being more fairly distributed across patients and more consistency in term of variability of the wait for each patient.

Table 3.5: Patient wait times in initial and refined schedules

Patient	1	2	3	4	5	6	7	8	9	10	11	12
Initial Scheduled Appointment Length (min)	180	30	150	300	180	60	90	420	60	150	90	90
Expected Wait Time in Initial Schedule (min)	0	3.8	3.2	9.6	2.2	17.2	14.7	3.0	11.8	22.3	26.7	19.4
Expected Wait Time in Refined Schedule (min)	0	4.4	5.8	5.1	11.3	10.8	9.2	16.4	6.9	8.1	13.8	10.4
Standard Deviation of Wait Time in Initial Schedule (min)	0	5.0	5.3	14.9	8.9	25.6	24.7	11.2	19.6	27.7	38.9	32.9
Standard Deviation of Wait Time in Refined Schedule (min)	0	5.4	6.9	13.3	19.1	22.8	19.7	24.5	18.3	20.1	30.4	24.4

3.5 Conclusions and Future Research

Scheduling patient appointment times for chemotherapy infusion is a challenging task, largely due to the uncertainty in treatment times. In this paper, we formulated a two-stage stochastic integer program to refine appointment times of a pre-existing schedule, with the goal of simultaneously improving two important performance measures: expected patient wait times and expected total length of operations.

Solving exactly this large-scale mixed-integer model for realistic instances requires a prohibitive computational time, motivating us to design a heuristic algorithm exploiting the structure of the problem. This *Fix-Unfix* algorithm alternates between optimizing the first stage decisions and the second stage decision variables, which can be done very quickly. We then described several ways to compute lower bounds on the objective of the original problem. Comparing the objective value of solutions from the *Fix-Unfix* algorithm leads to a performance ratio of the heuristic of about 3%. The schedules obtained with the heuristic significantly outperformed the initial baseline schedule as well as schedules created with a simpler scheduling rule. Under our assumptions, and for the considered data set, using the *Fix-Unfix* algorithm could allow a reduction of the total daily expected patient wait time by more than an hour for the same total length of operations. We have shown that with limited changes to the schedule, that in turn can have minimal impact on patients prior to their appointment, it is possible to improve both the patient experience (via reduced wait

times) and the clinic performance (via reduced nurse overtime).

Implementing the *Fix-Unfix* heuristic would allow infusion centers to quickly generate different feasible appointment times schedules corresponding to different trade-offs between patient wait times and total length of operations, making it possible to then pick a schedule according to their preferences. We also observed that in the schedules created by our heuristic, the mean and variance of waiting times are more fairly distributed among patients.

We now discuss the limitations of our work and propose three areas of future research:

Patient sequencing: Our approach is to refine a pre-existing schedule and we assume that the sequence of patients is fixed and cannot be changed. This assumption is not only useful in making sure that the refined schedule stays close to the initial schedule but also simplifies the problem. We showed that *SROP* can be solved approximately very quickly and yields significant improvement over the initial schedule. However, our approach does not consider the patient sequencing optimization problem. Finding optimal sequences could yield some benefits such as: (1) it is possible that some changes in the sequence will not cause too large perturbations in the refined appointment times but still lead to a better schedule and (2) gaining insight as to what makes a good patient sequence could help in designing scheduling templates that the schedulers could use when building the initial schedule (e.g., longest treatment time first, shortest variance first etc...).

Additional assumptions: As described in Section 3.1.2, getting chemotherapy infusion can be a complicated process, involving other resources than a nurse and an infusion chair: a previous visit with a clinician or at phlebotomy might be required, then a pharmacist has to prepare and deliver the drug to be administered, and finally a nurse has to discharge the patient upon completion of the treatment. Our approach is based on a simplified model which does not take these steps in consideration and might underestimate delays and overlook some interactions between various areas of the cancer center. Creating a more realistic model is certainly possible, even though computation times might increase as a result.

Efficient scheduling rules: Although we demonstrated that an optimization-based approach could significantly reduce patient wait time and staff overtime, it is not easy to implement in a hospital setting. Thus, there is value in designing simple scheduling rules and guidelines that, although having less impact, are easier to implement. The experiment presented on Section 3.3 — schedules based on scaling the mean with the same constant factor throughout the day — does not yield much improvement over the base case schedule. This suggests that allowing variable appointment lengths throughout the day might be a crucial component of a schedule robustness. For instance, even if all patients within a day had the same distribution, the scheduled appointment time should vary throughout

the day. A study of a one dimensional version of the problem with only one infusion chair shows that optimal schedules allocate more time to patients in the middle of the day (see Appendix for more details). Specifically, extra time in the middle of the day reduces the propagation of delays that might occur in the morning, while shorter appointment times late in the day can reduce the likelihood of expensive overtime. Engineering similar scheduling guidelines for the multi-dimensional case with several infusion chairs has the potential to improve significantly the base-case schedule without adding the complexity of more elaborate optimization-based algorithms.

3.6 Appendix

3.6.1 Proof of Proposition 1:

Consider a fixed set of appointment times $A = \{a_p, p \in P\}$. We use an exchange argument to transform an optimal schedule into the First Chair Available schedule without changing its objective value. Since the exchange argument only involves two chairs, we consider an example with two infusion chairs for simplicity but the proof can be easily generalized to any number of chairs, doing exchanges on two chairs at a time.

If X is a chair assignment we refer to the chair patient p has been assigned to by $c_X(p) \in \{1, 2\}$. Suppose that there exists an optimal chair assignment X_{opt} that is different than the First Chair Available assignment X_{FCA} (otherwise, X_{FCA} is optimal). Let i be the index of the first patient that has a different assignment in X_{opt} than in X_{FCA}. Without loss of generality we assume that $c_{X_{opt}}(i) = 1$ and $c_{X_{FCA}}(i) = 2$. Similarly, let k be the index of the first patient sequenced after patient i that is assigned to chair 2 in X_{opt} — that is, $c_{X_{opt}}(k) = 2$. A schematic representation of the situation is presented on Figure 3.7, where X_{new} denotes the assignment after the exchange.

X_{new} is obtained by swapping chairs 1 and 2 after patient i in X_{opt}. X_{new} is formally defined as:

$$c_{X_{new}}(p) = \begin{cases} c_{X_{opt}}(p), & \text{if } p < i \text{ (case 1)} \\ 2, & \text{if } p \geq i \text{ and } c_{X_{opt}}(p) = 1 \text{ (case 2)} \\ 1, & \text{if } p \geq k \text{ and } c_{X_{opt}}(p) = 2 \text{ (case 3)} \end{cases}$$

Now we show that the objective of chair assignment X_{new} is at least as good the objec-

tive of X_{opt}. To do so we argue that the start time of the infusion of each patient is earlier (or the same) in X_{new} than in X_{opt}:

- Patients sequenced before patient i have not been moved so their start time is the same.
- Patient i and the following patients assigned to Chair 1 in X_{opt} have been moved to Chair 2. Since, by assumption, Chair 2 is the first available chair for patient i, he/she can start earlier (or at the same time) in X_{new} than in X_{old}.
- Patient k and the following patients assigned to Chair 2 in X_{opt} have been moved to Chair 1. Because patients are treated in order of a predefined sequence, and patient k is sequenced after patient i, he or she always starts after patient i. In X_{new}, patient k occupies the spot formerly occupied by patient i and therefore can start as early as patient i starts in X_{opt}, which allows him/her to start earlier (or at the same time) than in X_{opt}.

Therefore, in the new assignment, each patient begins the infusion no later than in the optimal assignment we started with, and the objective value of the new assignment cannot be higher than the optimal objective. Consequently, the new assignment is also optimal and is identical to the first chair available assignment (at least) up to patient i (included). This process can be iterated to achieve the first chair available assignment. □

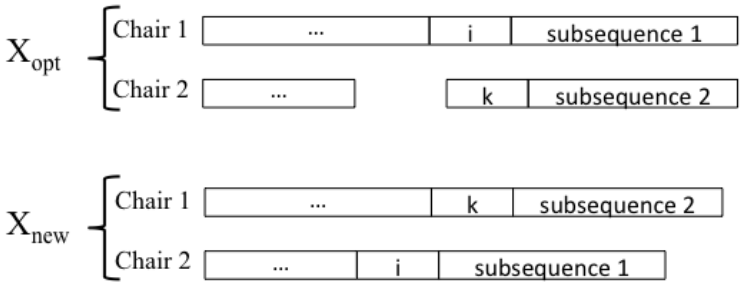


Figure 3.7: Assignments before and after the exchange

3.6.2 Example where the heuristic is not optimal

The following instance depicts an example where the *Fix-Unfix* heuristic presented in Section 3.3.3 is not optimal. For simplicity, we consider 4 patients, 2 chairs, no preparation

time by the nurse and two scenarios only. The intuition behind our example is to engineer values of treatment times in each scenarios so that the heuristic generates a set of appointment times and the associated chair assignment that are not optimal but cannot be improved by only changing one set of variable. In such a case, the *Fix-Unfix* algorithm will fail to find an optimal solution. Table 3.6 contains the treatment times for each patient in each scenario, counted in arbitrary units.

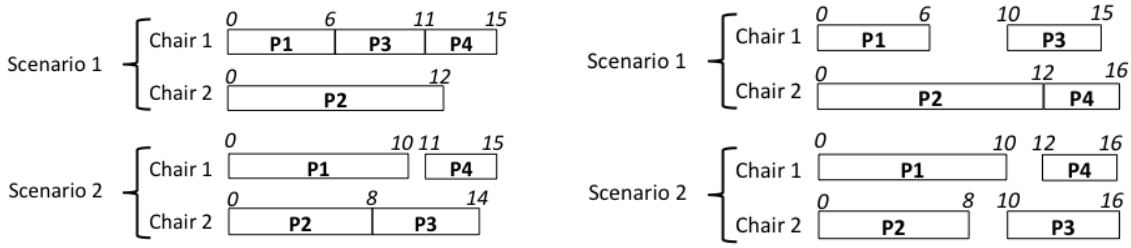
Table 3.6: Distribution of treatment times

	Scenario 1	Scenario 2
Patient 1	6	10
Patient 2	12	8
Patient 3	5	6
Patient 4	4	4

We then solve this instance using the *Fix-Unfix* algorithm and the mixed-integer programming formulation *SROP* and compare the results obtained with different values of the trade-off parameter λ. When λ is between 0 and 0.5 and between 0.67 and 1, the heuristic approach returns an optimal solution. However, when λ is between 0.51 and 0.66, the schedules obtained with the heuristic are not optimal. For example, consider the two schedules obtained when λ equals 0.6 (see Figures 3.8a and 3.8b):

- In the heuristic schedule, Patients 3 arrives at time 6 but only starts his treatment at time 8 in scenario 2. The expected total wait time is 1, while the expected total length of operations is 15. The weighted objective is: $0.6 * 1 + (1 - 0.6) * 15 = 6.6$.
- In the optimal schedule, there is no waiting time and the expected total length of operations is 16. Therefore the total weighted objective is $(1 - 0.6) * 16 = 6.4$.

We now describe how the heuristic creates the schedule and give an explanation as to why it cannot find an optimal solution. Recall that in the initialization of the heuristic all appointment times are set to 0. Then patients are assigned to chairs in order of their sequence. Then appointment times are optimized with respect to the current chair assignment. At this point, the heuristic obtains the schedule from Figure 3.8a. The next step of the heuristic is to fix the appointment times and try to come up with a better chair assignment. The optimal chair assignment is only one switch away: patient 4 would have to be moved to chair 2 in



(a) Schedule returned by the *Fix-Unfix* algorithm (b) Optimal Schedule

Figure 3.8: An instance where the heuristic is not optimal

scenario 1. However, doing this without changing the objective would add 1 unit of wait time for patient 4. Since the greedy approach does not allow this, the algorithm terminates.

3.6.3 Study of the single chair *SROP*

In an attempt to understand the structure of optimal solutions (which is more complicated than scheduling everyone with the same scale factor), we propose to look instead at a simpler version of the problem with only one chair, no nurse and homogeneous i.i.d. patients, each having an appointment length following a uniform distribution between 0 and 100 minutes. To solve this problem we modify the linear programming formulation of *SROP* by removing the chair assignment variables as well as the constraint that enforced waiting time for the nurse. The reduced model is:

$$\min \frac{\lambda}{m} \sum_{p \in P} \sum_{\omega \in \Omega} w_p^\omega + \frac{(1-\lambda)}{m} \sum_{\omega \in \Omega} L^\omega \tag{3.1}$$

Subject to:

$$a_p + w_p^\omega + s_p^\omega + t_p^\omega = d_p^\omega \quad \forall p \in P, \omega \in \Omega \tag{3.2}$$

$$a_{p_2} + w_{p_2}^\omega \geq d_{p_1}^\omega \quad \forall p_2 > p_1 \in P, \omega \in \Omega \tag{3.3}$$

$$L^\omega \geq d_p^\omega \quad \forall p \in P, \omega \in \Omega \tag{3.4}$$

$$a_p \geq 0 \quad \forall p \in P \tag{3.5}$$

$$w_p^\omega, d_p^\omega \geq 0 \quad \forall p \in P, \omega \in \Omega \tag{3.6}$$

- (3.1): Minimize a linear combination of the total expected waiting time and the expected end of the day.

- (3.2): Value of the discharge time of patient p in scenario ω.
- (3.3): Available chair constraint.
- (3.4): Definition on the length of operations in each scenario.
- $(3.5 - 3.6)$: Non-negativity restriction for variables.

This model is a pure linear program containing only continuous variables and therefore can be solve very quickly. Note that it is very similar to the linear program $LP(X)$ solved in the appointment time optimization phase of the *Fix-Unfix* algorithm. We solve the model for an instance with 10 patients and 10,000 scenarios for different values of the trade-off parameter λ. We present the results in Figure 3.9.

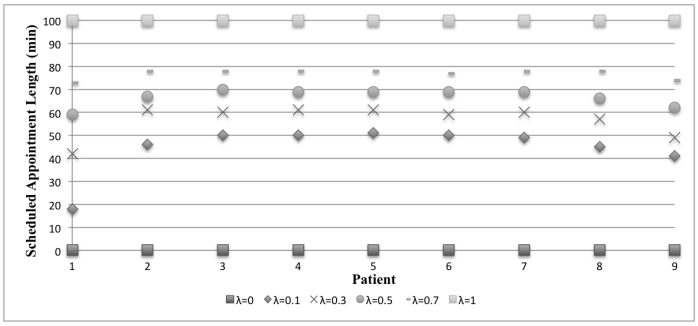


Figure 3.9: Scheduled appointment length for different values of trade-off parameter λ for the 1 chair problem

As expected, patients are always scheduled for their minimum possible appointment length when $\lambda = 0$ since we wish to minimize the total length of operations in this case. Similarly, patients are always scheduled for 100 minutes when $\lambda = 1$ since we minimize patient wait times.

For the non trivial cases ($0 < \lambda < 1$), we observe that the scheduled length is not constant across patients. Instead, schedules allow more time for patients in the middle of the day. These bell-shape curves are characteristic of scheduling problems involving a trade-off between makespan (or idle time) and waiting time. Intuitively, this trend can be explained by several factors:

- Early in the day, we only have limited uncertainty and very little propagating delays, so long wait times are unlikely and tighter appointments can be scheduled to maximize resource utilization.

- In the middle of the day, we may observe propagating delays from the morning infusions and scheduling longer appointments is an effective ways to recover.
- Towards the end of the day, even though we have high uncertainty resulting from the accumulating variability of all previous appointments, delays will not affect a lot of patients, since only a few of them are to be scheduled. Therefore it is optimal to take more risks and schedule tighter appointments, which will reduce the makespan.

CHAPTER 4

Scheduling Downloads During a Small Satellite Mission under Uncertainty

4.1 Introduction

4.1.1 The Satellite Downlink Scheduling Problem

Small satellites missions are a very efficient way of collecting data from space. Small satellites range from 750 kilograms to less than 1 kilogram and can be added at low cost to the launch of bigger satellites. That is why these missions have a shorter development and benefit to scientists as well as to students who can be easily involved in these projects [Baker and Worden, 2008].

We consider the problem of scheduling and managing the download of data from collecting satellites to receiving ground stations. A typical mission consists of a satellite in orbit around Earth collecting data and downloading them to several ground stations. The satellite uses solar energy to generate power and consumes it to stay in orbit and to communicate with the ground stations. Since downloading data has an energy cost, scheduling these transfers under resource constraints is a crucial part of the mission efficiency.

We address the dynamics of collecting, storing, using, and spilling both data and energy in designing an optimal downlink schedule over the planning horizon. A satellite can only communicate with a ground station when it is close enough, so as long as the orbit of the satellite is known, the planning horizon can be divided in n intervals corresponding to different download opportunities.

In each interval we have the opportunity to schedule a download and to choose how much data to download, as long as the satellite has enough data and energy in its buffers at that time to do so. During a download, the satellite transmits a scheduled amount of data

to the ground station. Due to inefficiencies in the transmission, the ground station typically only receive a fraction of the data sent from the satellite. The deterministic version of this scheduling problem has been studied in the case of a single satellite [Spangelo et al., 2015] and in the case of multiple satellites [Castaing, 2014].

4.1.2 Uncertainty in Ground Station Availability

One limitation of the deterministic models solving the download scheduling problem is that they assume that all parameters are known with certainty, which, in the context of space operations is not realistic, even for short-term missions. It might happen, for instance, that a ground station involved in a scheduled downlink with the satellite is unavailable due to a technical failure or because it is receiving a download from another satellite at the same time. Those failures and conflicts may not be known before the start of the mission and might compromise the scheduling decisions that have been made. In order to address this issue we focus our analysis on uncertain availability of ground stations.

We consider independent Bernoulli random variables for each interval equal to 0 with probability p_i if the ground station is unavailable and 1 otherwise. The probabilities p_i of these random variables will be fixed and considered as parameters, which can be estimated from analysis of historical data.

We introduce the set $S = [0, \ldots, 2^n - 1]$ of scenarios each defined as a list of possible availability for the ground stations across the planning horizon. The size of S is 2^n since there are two possible outcomes in each of the n intervals (ground station is available or not). We can represent this as a matrix $X \in R^{n,|S|}$ whose coefficient x_{is} is 1 if ground station i is available in scenario s and 0 otherwise. Each column of X represents a scenario.

Our main decision is to schedule downlinks for each interval. We introduce two important notions to handle the case where a downlink is scheduled but the ground station is not available:

1. The *ping* capability: if the satellite is equipped with this capability, it can first send a short message to the ground station, wait and listen to a reply from the ground station and start transmitting data only if the ground station gave the instruction, if the ground station is not available and does not reply back to the satellite ping, the satellite does not transmit any data, saving energy for future downloads. If the satellite is not equipped with the ping capability, all scheduled downlinks result in the satellite sending data (and consuming energy) regardless of the availability of the ground station to receive.

2. The *on-board scheduling* capability: if equipped, this capability allows the satellite to dynamically re-schedule the downlink plan after a failed downlink — we refer to this as *recourse*. Otherwise, the schedule is computed once at the start of the planning horizon in a ground station and then uploaded onto the satellite and is not modified throughout the mission.

Most small satellites are not equipped with either of those capabilities. However, the technology required to build these kinds of features on satellites is well known but has a high impact on cost of production and weight which is a crucial parameter for small satellites. Our goal is to decide whether or not the abilities to detect ground station unavailability and dynamically schedule the downlink plan lead to a significant gain in the total download and could justify the cost of developing more advanced small satellites.

4.2 Stochastic Optimization Approach

In order to evaluate the benefits of adding the ping and on-board scheduling capabilities, we develop four stochastic optimization models (no capability, ping only, on-board scheduling only, ping and on-board scheduling) and a deterministic model in which all ground stations are available. The goals of this section are to present each formulation, describe the relationships between them and ultimately prove that there is no benefit in having only the ping or the on-board scheduling capability. We postpone computational experiments to the next section where we will discuss the increase in total expected download when the satellite is equipped with both features.

4.2.1 Notation

Sets and Subsets

- I is the set of time intervals. We define one interval every time the satellite comes in range with a ground station.
- S is the set of scenarios. Each scenario corresponds to a binary vector of availability of the ground station of each interval. As an example, for three grounds stations we would define three intervals and the scenario $s = [0, 1, 1]$ would mean that the first ground station is not available to receive data and that the second and third ground stations are available.

Parameters

- η_i is the efficiency (fraction of downloaded data successfully received by the ground station) during interval i.
- ϕ_i is the data rate associated with downloading during interval i, measured in bits/second.
- α_i is the energy cost associated with downloading data during interval i, measured in joules/bit.
- e_{min}, e_{max}, d_{min} and d_{max} are the minimum and maximum allowable amounts of energy and data to be stored in the buffer, measured in joules and bits, respectively. The minimum amounts are typically set to 0 but in some situations it might make sense to require that the satellite always keep a certain amount of energy on board to process basic operations.
- e_{start} and d_{start} are the amounts of energy and data stored in the buffers at the beginning of the planning horizon, measured in joules and bits, respectively.
- δ_i^e and δ_i^d are the amounts of energy and data that are acquired by the satellite during interval i, measured in joules and bits, respectively. Energy is typically collected using solar panels and data can be gathered in multiple ways, using cameras or sensors.
- x_{is}: is 1 if the ground station from interval i is available in scenario s, and 0 otherwise.

Variables

- $q_i \geq 0$ (resp. $q_{is} \geq 0$) is the amount of data transmitted during interval i (resp. during interval i in scenario s), measured in bits.
- $e_{is} \geq 0$ and $d_{si} \geq 0$ are the amounts of energy and data available at the beginning of interval i in scenario s, measured in joules and bits, respectively.
- $h_{is}^e \geq 0$ and $h_{is}^d \geq 0$ are the amounts of excess energy and data spilled throughout interval i, in scenario s in the case where the energy or data buffers are full, measured in joules and bits, respectively.

4.2.2 Deterministic model

In this first model, we consider the simplest case where ground stations are always available.

$$
\begin{aligned}
P_0(\eta) = \max_{q,e,d} \quad & \sum_{i \in I} \eta_i q_i & & (0.1)\\
\text{subject to} \quad & q_i \leq \Delta t_i \phi_i & \forall i \in I \quad & (0.2)\\
& d_{i+1} = d_i + \delta_i^d - q_i - h_i^d & \forall i \in I \quad & (0.3)\\
& e_{i+1} = e_i + \delta_i^e - \alpha_i q_i - h_i^e & \forall i \in I \quad & (0.4)\\
& d_{i,} \leq d_{max} & \forall i \in I \quad & (0.5)\\
& e_{i,} \leq e_{max} & \forall i \in I \quad & (0.6)\\
& d_{1,} = d_{start} & & (0.7)\\
& e_{1,} = e_{start} & & (0.8)\\
& q, d, e, h^d, h^e \geq 0 & & (0.9)
\end{aligned}
$$

- (0.1): the objective is to maximize the total download received on the ground over the planning horizon.
- (0.2): the amount of data downloaded is smaller than the download speed multiplied by the length of the interval.
- (0.3) and (0.4): data and energy dynamics.
- (0.5) and (0.6): capacity of the storage for data and energy.
- (0.7) and (0.8): amount of data and energy available at the beginning of the mission.
- (0.9): non-negativity variable restrictions.

4.2.3 Basic satellite, no ping or on-board scheduling

Problem description: The satellite has no way to know if a ground station is available or not. If a downlink has been scheduled, the satellite consumes energy associated with it even if the ground station is not listening. We also assume that the data that was scheduled to be transmitted is lost since the satellite will not try to download it at a later that. Note that in this case, we do not consider recourse (ability to change the download plan during the mission). Therefore, only one schedule is built for the mission and the download variable

q only depends on interval i but not on scenario s.

Main Decision: q_i is how much to download during interval i.

$$
\begin{aligned}
P_1(\eta) = \max_{q,e,d} \quad & \sum_{i\in I}\sum_{s\in S} p_s \eta_i q_i x_{is} && && (1.1)\\
\text{subject to} \quad & q_i \le \Delta t_i \phi_i && \forall i \in I && (1.2)\\
& d_{i+1,s} = d_{i,s} + \delta_i^d - q_i - h_{is}^d && \forall i \in I\ \forall s \in S && (1.3)\\
& e_{i+1,s} = e_{i,s} + \delta_i^e - \alpha_i q_i - h_{is}^e && \forall i \in I\ \forall s \in S && (1.4)\\
& d_{i,s} \le d_{max} && \forall i \in I\ \forall s \in S && (1.5)\\
& e_{i,s} \le e_{max} && \forall i \in I\ \forall s \in S && (1.6)\\
& d_{1,s} = d_{start} && \forall s \in S && (1.7)\\
& e_{1,s} = e_{start} && \forall s \in S && (1.8)\\
& q, d, e, h^d, h^e \ge 0 && && (1.9)
\end{aligned}
$$

Observations: This model is very similar to the deterministic model $P_0(eta)$. The objective function is now the expected total download received on the ground with respect to the availability random variable x_{is}. Note that in (3) and (4) (data and energy dynamics), how much data or energy is collected and how much data is downloaded does not depend on scenario s since energy and data are used regardless of the ground station availability. Therefore the quantity of data and energy in the buffer and spilled will take the same values across all scenarios:

(3) implies that $d_{i,s} = d_i$ and $h_{i,s}^d = h_i^d \ \ \forall i, s$

(4) implies that$e_{i,s} = e_i$ and $h_{i,s}^e = h_i^e \ \ \forall i, s$

So the model reduces to a deterministic model with adjusted efficiency:

$$\tilde{\eta_i} = \eta_i \sum_{s\in S} p_s x_{is} = \eta_i E[x_i] = p_i \eta_i$$

$$
\begin{aligned}
P_1(\eta) = P_0(\tilde{\eta}) = \max_{q,e,d} \quad & \sum_{i \in I} \tilde{\eta}_i q_i & & (1.10)\\
\text{subject to} \quad & q_i \leq \Delta t_i \phi_i & \forall i \in I \quad & (1.11)\\
& d_{i+1} = d_i + \delta_i^d - q_i - h_i^d & \forall i \in I \quad & (1.12)\\
& e_{i+1} = e_i + \delta_i^e - \alpha_i q_i - h_i^e & \forall i \in I \quad & (1.13)\\
& d_{i,} \leq d_{max} & \forall i \in I \quad & (1.14)\\
& e_{i,} \leq e_{max} & \forall i \in I \quad & (1.15)\\
& d_{1,} = d_{start} & & (1.16)\\
& e_{1,} = e_{start} & & (1.16)\\
& q, d, e, h^d, h^e \geq 0 & & (1.18)
\end{aligned}
$$

4.2.4 Partially equipped satellite: ping capability only

Problem description: The satellite knows if a ground station is available or not before sending data. If a downlink has been scheduled and the ground station is not available, the satellites does not send data and avoid wasting its energy. We still do not consider recourse so only one schedule is built for the mission and the download variable q still only depends on interval i but not on scenario s.

Main Decision: q_i is how much to download during interval i.

$$
\begin{aligned}
P_2(\eta) = \max_{q,e,d} \quad & \sum_{i \in I} \sum_{s \in S} p_s \eta_i q_i x_{is} & & (2.1)\\
\text{subject to} \quad & q_i \leq \Delta t_i \phi_i & \forall i \in I \quad & (2.2)\\
& d_{i+1,s} = d_{i,s} + \delta_i^d - q_i x_{is} - h_{is}^d & \forall i \in I \; \forall s \in S \quad & (2.3)\\
& e_{i+1,s} = e_{i,s} + \delta_i^e - \alpha_i q_i x_{is} - h_{is}^e & \forall i \in I \; \forall s \in S \quad & (2.4)\\
& d_{i,s} \leq d_{max} & \forall i \in I \; \forall s \in S \quad & (2.5)\\
& e_{i,s} \leq e_{max} & \forall i \in I \; \forall s \in S \quad & (2.6)\\
& d_{1,s} = d_{start} & \forall s \in S \quad & (2.7)\\
& e_{1,s} = e_{start} & \forall s \in S \quad & (2.8)\\
& q, d, e, h^d, h^e \geq 0 & & (2.9)
\end{aligned}
$$

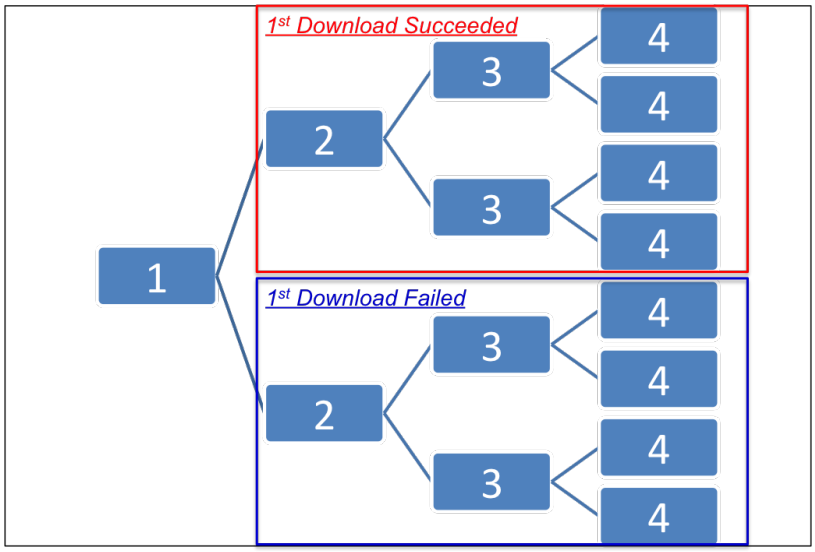


Figure 4.1: Sub-problems after the first download opportunity

Observations: The only difference with the stochastic model with no capability $P_1(\eta)$ is the random variable x_{is} in constraints (2.3) and (2.4): if the ground station of interval i is not available in scenario s, then $x_{is} = 0$ and the energy and data loss term is canceled out which allows the satellite to save its energy and data for future opportunity. However recourse is not allowed and the downlink plan is independent of the scenarios so the schedule does not have the flexibility to take advantage of the energy saved in some scenarios.

The optimization problem can be simply viewed has:

$$P_2(\eta) = \max_{q} \sum_{i \geq 1} \sum_{s \in S} p_s \eta_i q_i x_{is}$$

subject to: q feasible

Let q_1 be the amount of data scheduled to be sent in interval 1, the problem can be rewritten following Figure 4.1 as:

$$P_2(\eta) = \max_{q} \; p_1(\eta_1 q_1 + \sum_{i \geq 2} \sum_{s \in S} p_s \eta_i q_i x_{is}) + (1 - p_1) \sum_{i \geq 2} \sum_{s \in S} p_s \eta_i q_i x_{is}$$

subject to: q feasible after using data and energy during download at interval 1

Which is the same as:

$$P_2(\eta, e_{start}, d_{start}) = \max_{q} p_1 q_1 + \sum_{i \geq 2} \sum_{s \in S} p_s \eta_i q_i x_{is}$$

subject to: q feasible after using data and energy during download at interval 1

Which inductively proves that it is equivalent to solve a model where the satellite consumes data and energy regardless of the availability of the ground station. In this case, the ping capability is useless and this model is equivalent to the model with no capabilities, that is $P_2(\eta) = P_1(\eta) = P_0(\tilde{\eta})$. And this model is equivalent to a deterministic model.

4.2.5 Partially equipped satellite: on-board scheduling only

In this case, we consider recourse, which means that the schedule can be dynamically changed after each interval. However, the satellite has no way to know if a ground station is available and send data regardless. So at the end of an interval, the quantity of energy and data on satellite is independent of the scenario and the on-board satellite scheduler will make the same decisions regardless of the availability of the ground station during the previous interval. Therefore the recourse capability is useless in this case and the model is equivalent to the base case $P_1(\eta)$.

4.2.6 Fully equipped satellite: ping and on-board scheduling available

We now consider a satellite that only downloads when a ground station is available and can dynamically reschedule the downlink plan after the realization of each interval.

A common feature of stochastic optimization models with recourse is the non-anticipativity constraint [Ruszczyński, 1997]: since, in practice, the on-board scheduler doesn't have information about what is going to happen in future intervals, we need to enforce that, if two scenarios share the same path for the first N periods, their schedules are similar for the first $N + 1$ periods. This can be done by adding non-anticipativity constraints to the model. We create a matrix A whose coefficient a_{s_1,s_2} is the first period such that the path followed by scenarios s_1 and s_2 is different. Consider the scenario tree obtained for four periods in Figure 4.2, in which each node is labeled (i, s) and the each arc is labeled 0 (if the ground station is not available) or 1 (if the ground station is available). Note that according to that

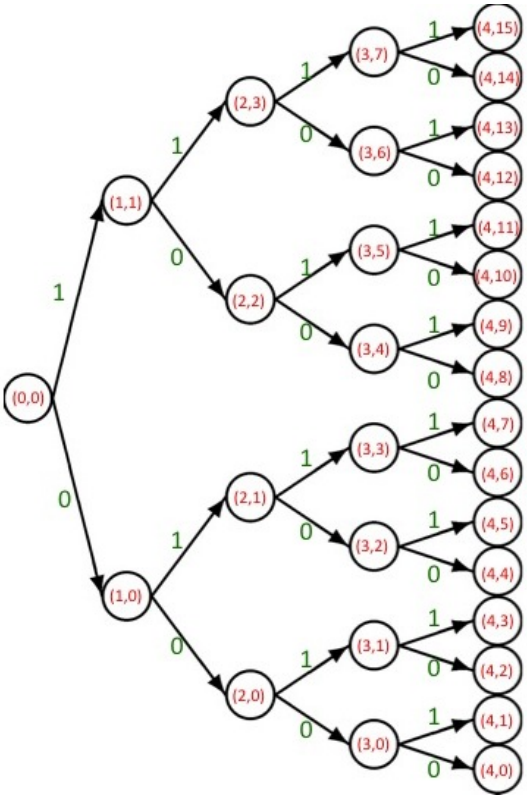


Figure 4.2: Binary Scenario Tree

design, the index of a scenario s is exactly equivalent to the base 2 number formed by the availability of the ground stations in that scenario, taken in reverse order — for example scenario 11 corresponds to the availability 1 in period 1, 0 in period 2, 1 in period 3 and 1 in period 4 and $1 * 2^1 + 1 * 2^0 + 0 * 2^2 + 1 * 2^3 = 11$. This makes it very easy to label every node in the tree, even for a large number of periods.

The non-anticipativity constraints simply ensure that downloads in scenarios s_1 and s_2 share the same decision up to A_{s_1,s_2}:

$$q_{i,s_1} = q_{i,s_2} \; \forall (s_1, s_2) \in S^2 \; \forall i \in [1, A_{s_1,s_2}]$$

$$P_4(\eta, e_{start}, d_{start}) = \max_{q,e,d} \quad \sum_{i\in I}\sum_{s\in S} p_s \eta_i q_{is} x_{is} \tag{4.1}$$

subject to

$$q_{is} \leq \Delta t_i \phi_i \qquad \forall i \in I\ \forall s \in S \tag{4.2}$$

$$d_{i+1,s} = d_i + \delta_i^d - q_{is}x_{is} - h_{is}^d \qquad \forall i \in I\ \forall s \in S \tag{4.3}$$

$$e_{i+1,s} = e_i + \delta_i^e - \alpha_i q_{is} x_{is} - h_{is}^e \qquad \forall i \in I\ \forall s \in S \tag{4.4}$$

$$d_{i,s} \leq d_{max} \qquad \forall i \in I\ \forall s \in S \tag{4.5}$$

$$e_{i,s} \leq e_{max} \qquad \forall i \in I\ \forall s \in S \tag{4.6}$$

$$d_{1,s} = d_{start} \qquad \forall s \in S \tag{4.7}$$

$$e_{1,s} = e_{start} \qquad \forall s \in S \tag{4.8}$$

$$q_{is_1} = q_{is_2} \qquad \forall (s_1, s_2) \in S^2\ \forall i \in [1, A_{s_1,s_2}] \tag{4.9}$$

$$q, d, e, h^d, h^e \geq 0 \tag{4.10}$$

Observations: The main difference of this model with the previous ones is that the decision variable q now depends not only on interval i but also on scenario s which allows recourse, in the limitation of the non-anticipativity constraint (4.9). This leads to an explosion of the state space called *curse of dimensionality* [Pereira and Pinto, 1991] where the number of variables and constraints grows exponentially with the length of the planning horizon.

4.3 Computational experiments

In this section, we run two computational experiments to study the run time and the solution quality achieved by a basic satellite (model P_1) and a fully equipped satellite with ping and on-board scheduling capabilities (model P_4). We do not explicitly consider the cases where the satellite has only the ping or only the on-board scheduling capability since we showed in the previous section that they were equivalent to having no capability at all. Therefore two models we are focusing on are:

- The case of a basic satellite that follows a plan computed using the deterministic model presented in Section 4.2.3.
- The case of a fully equipped satellite that is able to dynamically schedule its download depending on the successive realization of the ground stations availability. (Section 4.2.6)

Throughout these experiments we randomly generate data sets using the following parameters:

4.3.1 Parameters Value:

We use $N(\mu, \sigma)^+$ to denote the normal distribution with mean μ and standard deviation σ truncated to positive values.

- The duration of interval i (minutes): Δt_i follows a truncated normal distribution $N(10, 4)^+$
- Data rate associated with downloading during interval i (bits/sec): ϕ_i follows a truncated normal distribution $N(5, 25)^+$
- Energy cost associated with downloading during interval i (Joules/bit): α_i follows a truncated normal distribution $N(1.6, 0.8)^+$
- The energy collected during interval i (Joules): δ_i^e follows a truncated normal distribution $N(40, 20)^+$
- The data collected during interval i (bits): δ_i^d follows a truncated normal distribution $N(25, 12.5)^+$

For simplicity, we assume that all ground stations have an efficiency η of 1, that the satellite starts with a full buffer of energy and empty buffer of data at the beginning of the horizon and that the capacities of these buffers are not limited.

4.3.2 Computational Complexity:

We begin by comparing the time necessary to solve the two models.

In the case of the basic satellite, time is not an issue: the run time of the model P_1 was linearly dependent with the number of interval and was taking less than a minute, even for instances with 100 intervals. This is not surprising since we showed that P_1 was equivalent to a deterministic model P_0 with adjusted efficiency, which has a number of variables and constraints growing as n.

In the case of the fully equipped satellite, solving the stochastic optimization P_4 was more challenging. We solved a series of instances with increasing number of intervals (identical to number of ground stations) and reported the run time on Figure 4.3. We observe that the run time increases exponentially with the number of intervals in this case.

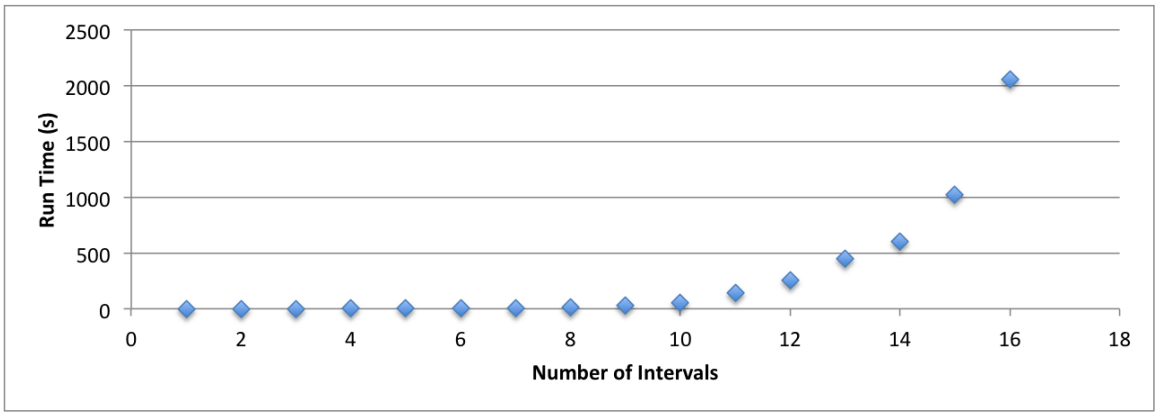


Figure 4.3: Run time of the stochastic optimization model P_4

This makes sense since the number of variables and constraints in P_4 grows as $O(2^n)$. Solving an instance with 16 ground stations took more than 30 minutes. This dependency greatly limits the size of the problem that we can solve using this approach. We discuss ways to mitigate this in the conclusion and future work section of this chapter.

4.3.3 Comparison of Performance in Expected Total Download:

We now compare the objective values obtained using the two models P_1 and $P4$. To do so, we randomly generated 50 instances of the problem with 10 grounds stations for different values of the ground station availability probability, solved using both models and reported the average objective value (total expected download) across all instances. We also compare the results obtained by the two models to two other scheduling paradigms:

- A greedy heuristic that simply consists in always downloading as much as possible while making sure to not exceed the available amount of time and energy.
- A schedule created under perfect information which means that we optimally schedule downloads for each scenario independently (i.e., knowing in advance which ground stations will be available). This value is computed by relaxing the non-anticipativity constraints (4.9) in model P_4. This could never be achieved in practice but this provides an upper bound on the objective of P_4.

Results are presented on Figure 4.4. We first notice, as expected, that the expected amount of data downloaded decreased when ground stations are less likely to be available. The maximum objectives are reached when all ground stations are always available (availability probability is 1) and all objectives drop to 0 when ground stations are never

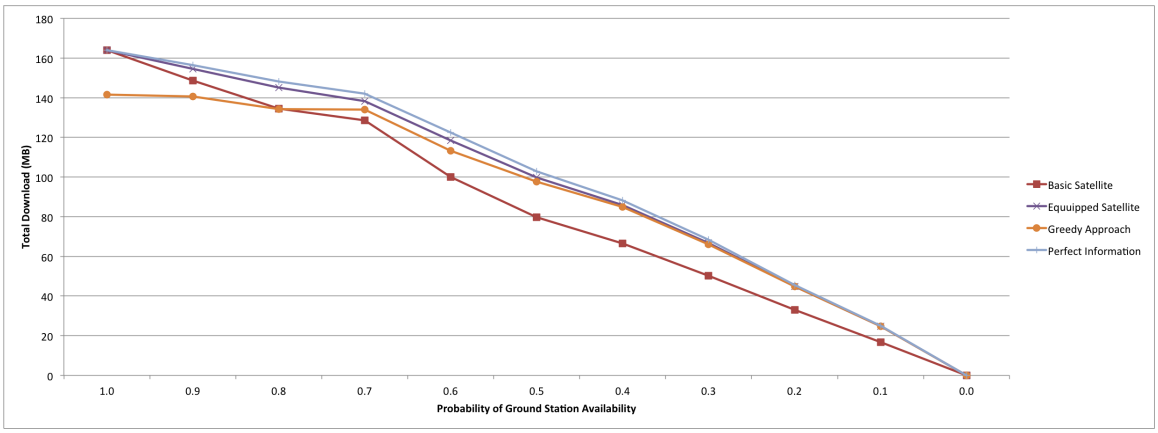


Figure 4.4: Comparison of performance for different scheduling strategies

available (availability probability is 0). Then we notice that the objective obtained under perfect information is always higher than the objective reached by the fully equipped satellites which, in turn, dominates the performance of the basic satellite. This makes sense since the optimization models used to compute these objectives are relaxations of each other. Note that, when the availability probability is 1, these 3 objectives are equal. This is explained by the fact that, in this case, ground stations are always available and the problem reduces to the deterministic case for which all models are equivalent to P_0. Finally we look at the behavior of the greedy heuristic approach. Its objective is dominated by the fully equipped satellite since the latter achieves the optimal value of the problem of dynamically schedule downloads (case where recourse is allowed) and the greedy solution is feasible for this problem. We observe a more interesting behavior when we compare the greedy heuristic to the basic satellite (i.e., greedy approach when recourse is allowed versus optimal solution without recourse). When ground stations are often available (probability greater than 0.8) the basic satellites performs better. Intuitively this is reasonable since we know that the basic satellite performs optimally when all ground stations are available. However, for lower availability probabilities, we see that the greedy heuristic outperforms the basic satellite. This is because the greedy approach has the ability to take different actions in different scenarios while the basic satellite follows a single download plan. For instance, if a few ground stations in a row are not available, a satellite that uses the greedy approach will have saved energy and will download a lot more during the next opportunity and this scenario is more likely to happen when the availability probability is low.

4.4 Conclusion

In this project, we studied the problem of scheduling downloads during a small satellite mission under uncertainty of availability of the receiving ground stations. We considered the difference between satellites having or not two different options: (1) the ability to check if a ground station is available or not before downloading data, which allows to not waste energy if the ground station is not listening and (2) the ability to dynamically adapt its scheduled plan during the mission when ground station failures occur. We showed that having only one of these capabilities did not allow better performance, since the underlying optimization models are all equivalent. However, if a satellite is fully equipped with both options, we observed that higher total expected downloads can be achieved. This comes at the cost of a much higher computational complexity. The methodologies developed in this work can be applied to other satellite designs and used to estimate the trade-off between complexity of satellites and expected profitability. The key limitation of the stochastic optimization approach used to solve the download scheduling problem in the case of a fully equipped satellite is the computational complexity due to the explosion of the state space when we increase the length of the planning horizon. We started developing techniques to mitigate this issue and solve bigger size problems. Such techniques include reducing the numbers of variables by transforming the scenario based model into a node based model where we only define a variable for each node of the scenario tree (Figure 4.2) and decomposing the problem in smaller pieces by dynamically dividing the mission horizon in independent time periods in which download choices do not impact future decisions (rolling horizon approach).

CHAPTER 5

Recovery Under Uncertainty in Airline Operations

5.1 Introduction

Airline companies operate at very high cost (crews, aircraft, fuel...) with tight margins and maximizing the efficiency of their system is extremely important. Carriers typically invest a lot of work to improve efficiency at several levels such as revenue management, long term strategic planning and day to day operations. A key performance indicator is the amount of delays in the system. Delays are caused by many different factors such as bad weather, mechanical problems, gate blockage or variability in flight times. These delays have a wide range of negative effects. From a customer perspective, delays globally decrease satisfaction and might cause missed connections which means that passengers need to be re-accommodated. From an operational standpoint, delays mean wasted time and resources (aircraft or crews) that could be used for a different flight, additional congestion at airports which increases the risk of accidents, extra fuel burnt for idling aircraft etc...

Because carriers design schedules to maximize efficiency, aircraft and crews are typically supposed to operate several flights each day (for domestic lines), with tight windows in between. This causes delays to propagate throughout the system. A delayed flight means that subsequent flights operated by this aircraft and crew have a high probability of being delayed as well. It is not uncommon to see a single perturbation in the morning impact flights over the entire network for the whole day.

In order to recover from perturbations, carriers use a variety of techniques which include cancellations, gate re-assignments, aircraft or crew swaps, diversions, or ground delay programs. More details about these mechanisms can be found in a survey about recovery strategies [Filar et al., 2001]. These decisions are typically made solely based on

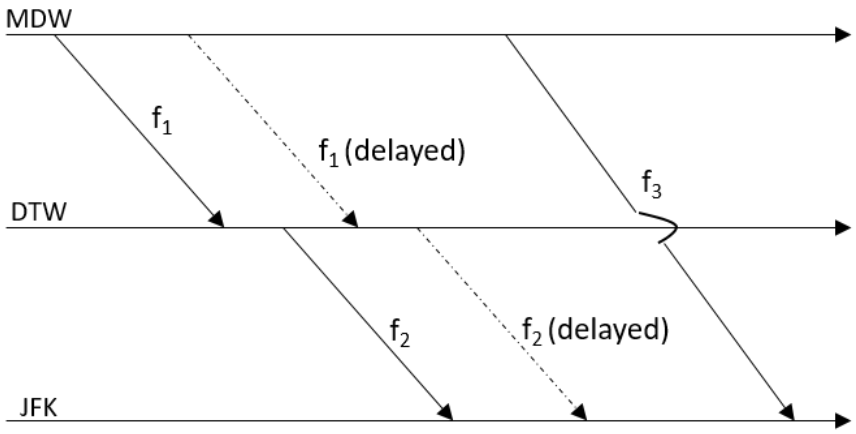


Figure 5.1: Motivating Example

the current state of the system, estimating how long flights will be delayed. However, it is possible that other perturbations will occur later in the day, forcing schedulers to reevaluate their plans.

Consider the example (see Figure 5.1) of a storm hitting the Chicago's Midway airport (MDW) in the morning. Departures of outbound flights and arrival of inbound flights will be delayed for some time until operations go back to normal, say in the afternoon. Suppose that a flight (f_1) from MDW to the Detroit airport (DTW) is scheduled to leave at 9:00AM and to land 90 minutes later. Also suppose that a large proportion of the passengers on this flight have booked a connecting flight from DTW to JFK (f_2), New York after a 30 minute lay-over in Detroit. Now, let's say that the first leg MDW-DTW is delayed by an hour. It is reasonable to assume that these passengers will miss their connection to New York so the airline then decides to book them on a direct flight from MDW to JFK (f_3) in the afternoon later that day, which will cause its own scheduling challenges, especially if that flight was already almost full. The passengers who missed their connection are now scheduled to arrive to New York at 7PM. The MDW-DTW flight departs at 10:00 AM and arrives an hour late. However, by then, the storm moved to Detroit and all flights there are delayed by an hour. The passengers could have made their connection and arrive to New York only an hour late instead of at night. This simple example shows that, when a good forecasting of delays is available, it is possible to make better decisions and to reduce the overall amount of delay in the system.

Airline recovery is a difficult problem that is a common theme in the aviation oper-

ations research literature [Rosenberger et al., 2003], [Eggenberg et al., 2010]. Predicting perturbations and future delays is also a challenging problem that has been less studied [Tu et al., 2008], [Xu et al., 2008]. In this paper we lay fundamental ground work to facilitate researchers to implement and compare different recovery strategies. We also discuss ways to account for uncertainty in the recovery decision making. Our main goal is to develop a flexible simulation framework that will model daily flights for a specific carrier, to introduce random perturbations in the system and to measure the impact of delays as well as the effectiveness of a given recovery strategy. In addition, we also introduce the notion of future uncertainty and describe approaches to take it into account in the recovery decision process. In section 2 we propose a methodology to generate complete data for a daily flight schedule based on publicly available data from the Bureau of Transportation Statistics. In section 3 we describe our simulation tool. In section 4, we discuss computational experiments and results. In section 5, we discuss future research, approaches to estimate correlation between delays and ideas to account for future perturbations when planning recovery.

5.2 Generating a Schedule

Since our goal is to simulate recovery and delay propagation throughout an airline network, we need to access data containing relevant information such as, original schedule, daily itinerary for each aircraft and crew, scheduled flight time etc. Airline data is well known to be highly proprietary and multi-dimensional (flights, crews, passengers, etc) and it is often a challenge for researchers to get access to comprehensive real data sets that cover their specific needs. One of the key contributions of this work is to enable the community to develop and test new algorithms by designing a methodology to create realistic data sets.

We decided to limit our scope to one day and one carrier. Studying only one day makes sense since fairly few domestic flights are scheduled at night which means that delays tend to not propagate from one day onto the next. It is true, however, that in some rare cases disruptions can impact the network for several days, for instance an aircraft might need to undergo maintenance at a specific airport so canceling the last flight before a scheduled maintenance might lead to changes in the next day schedule. In these instances, recovery strategies are different than the ones used for day-to-day operations so are considered out of scope for this project. Additionally, we choose to neglect interactions between different airline companies and focus on a single carrier's network. Most majors carriers operate their own schedule and very rarely share aircraft or crews with other airlines in case of

disruptions.

Instead of randomly creating a schedule from scratch we start from information publicly available on the Bureau of Transportation Statistics (BTS) website [BTS, a] to obtain a realistic schedule. Given a specific day and carrier, we can get the following historical data:

- Tail Number
- Origin
- Destination
- Scheduled departure time
- Scheduled arrival time

These fields give us a good idea of what the schedule was for that day. However we need additional information since our simulation model aims to not only consider flights but also aircraft, crews and passengers. We are specifically interested in:

- The type and capacity of each aircraft
- The number of passengers on each flight
- The number of passengers in transit on each flight and their connecting flights
- The crew flying each flight

We developed and implemented a method to randomly generate values for these additional fields. This data generation algorithm has 3 steps: aircraft data, passenger data and crew data.

5.2.1 Aircraft Data

In order to determine the capacity of each aircraft in our schedule (and in turn estimate how many passengers were on the flight), we use tail numbers to find the type of each aircraft (e.g., A320, 737...). An aircraft tail number is the equivalent of a car's license plate: it is a unique identifier of a specific aircraft. There are various websites that contain information about how tail numbers are assigned to each aircraft but each carrier follows its own procedure (which might change over time). This makes it difficult to write code to convert a tail number to an aircraft type. Fortunately, the Federal Aviation Administration (FAA) has an

online registry containing the aircraft type and tail numbers of all US commercial aircraft [FAA,] where we can directly look up specific tail numbers. To avoid checking all of them manually, we plan to write a Python script to do it automatically.

Once we have the aircraft type, we can find the capacity of the aircraft. The FAA website does not give the size of each aircraft, since the number of seats depends on the operational decisions made by each carrier (e.g., seat configurations, size of first class...). This information can be found on each individual carrier's website, for example the Delta fleet is detailed at [Del,]

5.2.2 Passenger Data

5.2.2.1 Number of Passengers on Each Flight

Getting passenger data is necessary to estimate important performance metrics such as the total number of passenger delay minutes or the number of passenger missed connections throughout the day. This data is highly proprietary so we use the average load factor to relate the amount of passengers per flight to the capacity of its aircraft. The passenger load factor is a metric commonly used in the aviation world. It is typically defined as the number of passenger-kilometers divided by the seat-kilometers available. Load factors of major airlines are available on the BTS website at [BTS, b]. For instance, the average load factor for domestic Delta flights in 2015 was 85%. Depending on how specific we want to be, we could use a single average number for the entire network or specific load factors for each market (origin - destination pair) to model the fact the some flights are typically more full than others.

5.2.2.2 Connecting Passengers

The vast majority of passengers book itineraries that only have one or two flights. For simplicity, we assume that this is the case for all passengers in our model. Therefore, we can divide passengers from each flight f in 3 groups (see Figure 5.2):

- Group 1: Direct passengers who have only flight f in their itinerary
- Group 2: Passengers who had another flight before flight f and stop at flight f's destination

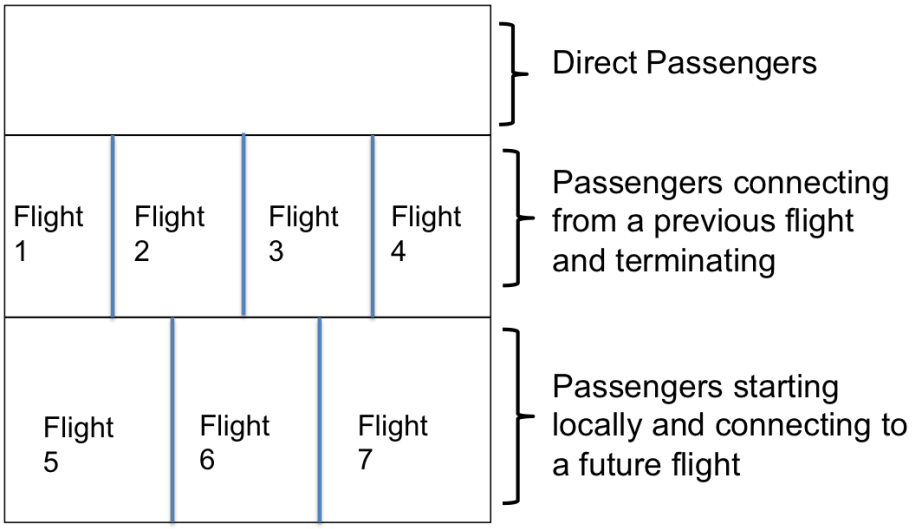


Figure 5.2: Passenger diagram for flight f

- Group 3: Passengers who started their itinerary with flight f and have another flight after

Since we already generated the total number of passengers for each flight, we need to distribute these passengers among the three groups we just described. We are going to this sequentially for each flight. The main difficulty is that groups 2 and 3 of different flights are not independent; suppose that 10 passengers have an itinerary containing flight f_1 as a first leg and flight f_2 as a second leg. Then group 3 of flight f_1 must contain at least 10 passengers and group 2 of flight f_2 must contain at least 10 passengers. We propose an algorithm that does this while making sure to not exceed the total number of passengers that has been previously assigned to each flight. The key idea is to assume to that some percentage of the passengers of each flight have a second leg (group 3), assign them to group 2 of other flights for their second leg if possible (that group 2 might be already full of passengers coming from a different first leg) and finally assign all remaining passengers to be direct (group 1).

The first step is to assign a proportion x of passengers of each flight to be starting locally and connecting to a future flight (group 3). Note that this allocation is a starting point and might change during the algorithm, in the case where there is no available feasible flight to connect to or that they already have too many incoming passengers. Also note that parameter x is identical for all flights but could easily be randomized or depend on flights, time of day, aircraft size etc.

The second step is to go through the list of flights and sequentially create connections while making sure to not exceed the capacity of group 2 of the second leg flight, which is (1-x) times the number of passengers for that flight. Lets consider the case of a flight f: we create a list of potential candidate next flights f_i that satisfy the following conditions:

- f_i origin is equal to the destination of f
- f_i scheduled departure time is between 30 minutes and 90 minutes after the scheduled departure time of flight f.
- f_i is not going back to the origin of f
- the distance of the 2 legs is no more than 3 times the distance of a direct flight. (We compute straight line distances between airports using latitudes and longitudes found on [lat,].)

We then randomly divide passengers from flight f who have a second flight in their itinerary (group 3) between the n candidate flights f_i, $(1 \leq i \leq n)$ making sure that we don't exceed the remaining number of unassigned seats in group 2 of each flight f_i (passengers coming from a previous flight).

Once we did this for all flights, we go through the list of flights one more time and count the number of passengers from group 2 and group 3 and assign the rest to group 1, which means that they will have only one leg in their itinerary.

5.2.3 Crew Data

Similar to aircraft, crews can also contribute to delay propagation. If a flight operated by a given crew arrives late then the departure of the next flight of the crew is likely to be delayed as well. This is especially important when a crew does not stay on the same aircraft throughout the day. Consider a crew flying a flight f_1 on an aircraft A and then a flight f_2 on an aircraft B, while a different crew uses aircraft A to operate flight f_3. If flight f_1 is late enough then flights f_2 and f_3 will both have a delayed departure. In this case both crew and aircraft propagate the delays in two different flights.

To model this in our simulation tool, we need to access crew data, i.e., which crew operated each flight. This information is not publicly available online so we developed our own method to generate crew assignments. For simplicity, we only consider cockpit crews and we assume that cabin crew stay with the same cockpit crew for a day. One simple way to do create a crew assignment is to assign one separate crew to each tail number for the

day. However this would lead to violate some basic Department of Transportation (DOT) constraints (namely maximum flight and duty time per day) as well as overlooking the fact that real schedules have crew swaps (i.e., crew switching tail number in the middle of the day). To mitigate these two limitations, we follow a more advanced procedure in which we consider one crew at a time, start that crew on a random flight at the beginning of the day and then randomly decide if the crew stays on the same aircraft for the next flight or if we have a swap and the crew changes aircraft. We add flights to the crew's schedule until we reach the maximum daily flight or duty time defined as the time between the crew's first departure and the crew's last arrival for the day. We then pick a new crew and repeat until all flights have been assigned to a crew.

Parameters:
CrewSwapProbability: $[0, 1]$
MaxFlightTime ≥ 0
MaxDutyTime ≥ 0

Crew Assignment Algorithm:

1. crewID=0
2. Pick a random tail number from the schedule and find the earliest possible departure with that tail that does not have a crew yet. Assign flight to crewID, update total flight and duty times for this crewID and remove the flight from the list.
3. Randomly decide if the crew will stay with this tail or not based on the CrewSwap-Probability.
4. If (no swap): Find the next available flight with the current tail number and check if it feasible to assign it to the current crew (i.e., does not violate the max flight and duty times.
    - If (feasible): assign flight to crewID, update total flight and duty times for this crewID and remove the flight from the list.
    - If (not feasible): try to assign the crew to a flight on a different tail number using step 5.
5. If (swap or could not keep crew with the same tail number): Generate list of candidates for crew swap (see below)

- If (candidate list is not empty): randomly pick a candidate flight, assign it to crewID, update total flight and duty times and remove the flight from the list.
- If (candidate list empty): no flights with a different tail number are available for the crew, try to assign the crew to the next flight on their current tail number following step 3.

6. If could not assign crewID to anything, start a new crew (crewID++). Otherwise return to beginning of loop to assign the next flight to the crew.

7. Stop when the list of flights is empty.

For a given flight f, the list of candidates for crew swap contains all flights f_i that satisfies the following conditions:

- f_i origin is equal to the destination of f
- f_i scheduled departure time is between 30 minutes and 90 minutes after the scheduled arrival time of flight f.
- f_i and f have the same aircraft type
- Assigning flight f_i to the crew does not violate the max flight and duty times.

5.3 Simulation Tool

Starting with basic information about a daily schedule obtained from BTS, we generated a realization of a complete schedule. Now we wish to develop a simulation tool allowing us to introduce delays and see how the system behaves under various recovery strategies. We begin this section by defining delays, then we describe two different simulation scenarios: one without any recovery mechanism and one allowing aircraft swaps. Many other scenarios, such as crew swaps and cancellations, could be implemented in the future.

5.3.1 Delays

We distinguish three different types of delay:

1. Primary Delay: delay that we introduce in the system as an exogenous perturbation. Primary delays impact the earliest possible departure time of the flight and could be caused by weather, a mechanical problem, a crew or gate agents being late for work, etc.

2. In-flight Delay: random perturbation of the flight time that we introduced to simulate randomness in taxi and flight. We aggregate the variability in taxi in and out and during flight and only generate one random delay for the flight, not three (one for taxi out, one for in-flight, and one for taxi in). Note that this "delay" could be negative, meaning that the flight took less time than it was scheduled for.

3. Secondary Delay: delay resulting from propagation. This type of delays occurs when the aircraft or crew operating a flight are not available at the scheduled departure time because they are delayed from a previous flight.

Example:

Consider a flight with a scheduled departure at 9:00AM, estimated taxi in and out of 10 minutes each, an estimated flight duration of 90 minutes and a schedule arrival at 11:00AM.

- Case 1: We do not introduce primary or in-flight delay and the aircraft and crew are on time. The flight leaves on time and we have taxi_out = 10min, flight=90min, taxi_in=10min so the flight arrives at 10:50AM, i.e., 10 minutes early.

- Case 2: We introduce a 20 minute primary delay, no in-flight delay and the aircraft and crew are on time. Then the flight leaves at 9:20AM and arrives at 12:10AM, i.e., 10 minutes late.

- Case 3: We introduce a 20 minute primary delay, a 10 minute in-flight delay and the aircraft and crew are on time. Then the flight leaves at 9:20AM and arrives at 12:20PM, i.e., 20 minutes late.

- Case 4: We introduce a 20 minute primary delay, a 10 minute in-flight delay and assume that the aircraft is 30 minutes late, and the crew is 15 minutes late from a previous flight. Then the flight earliest possible departure is 9:20AM because (primary_delay=20min). At this time, the crew is ready but the aircraft is not ready until 9:30AM so we have a secondary delay of 10 minutes and the flight leaves at 9:30AM. Then we have taxi_out=10min, flight=100min, taxi_in=10 so the flight arrives at 11:30AM i.e., 30 minutes late. This final 30 minute delay is the result of 20 minutes of primary delay, 10 minutes of secondary delay and 10 minutes of in flight delay (that is 40 minutes) and a 10 minute buffer in the schedule.

5.3.2 No Recovery Strategy: Delay Propagation Model

The first scenario that we consider is a simple delay propagation model: we do not take any action when perturbations occur. When a flight is delayed, it will leave as soon as possible on its scheduled route and arrive late at its destination. Passengers will then potentially miss their connecting flight if they have one. We do not model passenger re-accommodation in this simulation tool, we simply report the number of missed connections as part of the list of output metrics. We consider flights one after another, in order of scheduled departure, generate delays and compute their actual departure and arrival time, we then accordingly delay subsequent flights using this crew or aircraft. An important concept that we use in this algorithm is the aircraft (resp. crew) turn time which is defined as the time that an aircraft (resp. a crew) needs between an arrival and its next departure to complete operations such as unboarding and boarding of passengers and luggages as well as potential refueling for the aircraft and taking a break and going through various check lists and flight plans for the crew. These turn times can typically take up to 30 minutes.

The simulation scheme follows the procedure below.

Initialization:

First, we randomly draw a realization of primary and in-flight delays (see section 2) and we initialize all secondary delays to 0. Then we create 3 data structures:

- A sorted list F containing all flights based on their earliest possible departure time defined as (scheduled_departure + primary_delay + secondary_delay).
- A list of aircraft (tail numbers) with their ready time and their current location, each aircraft ready time is initialized to 0 and their current location is set to the origin of their first flight of the day.
- A list of crews with their ready time and their current location, initialized the same way as the list of aircraft.

Propagation:

While F is not empty,

1. Consider first remaining flight in the queue F (earliest possible departure) say f_0
2. Set current time t = earliest possible departure of f_0
3. Check if aircraft and crew are ready (and in the right location) at that time

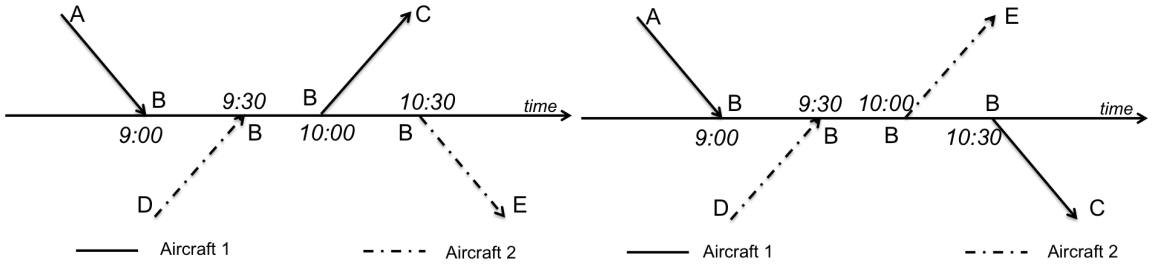
- If (aircraft and crew are ready) calculate arrival time, update aircraft and crew locations to be f_0 destination, and their ready time to arrival time + min turn time. Remove f_0 from F and return to step 1.
- If (aircraft location is not the current origin): the aircraft assigned to flight f_0 is still operating a previous flight and has not yet arrived at the origin airport. In this case, we know that earliest departure of flight f_0 will be pushed back by at least the time needed for the aircraft to turn so we add the minimum aircraft turn time to secondary delay. Return to step 1.
- If (crew location is not the current origin): similar to the previous case, the crew is still operating a previous flight so we add the minimum crew turn time to secondary delay. Return to step 1.
- If (aircraft is not ready): the aircraft is at the correct airport but is still turning, we simply add the slack to secondary delay (aircraft ready time earliest possible start time). Return to step 1.
- If (crew is not ready): the crew is at the correct airport but is still turning, we simply add slack to secondary delay (crew ready time - earliest possible start time). Return to step 1.

5.3.3 A Simple Recovery Strategy: Aircraft Swaps Model

We now introduce an elementary recovery strategy: the aircraft swap. When two aircraft (say aircraft 1 and aircraft 2) are on the ground at the same time in a given station of the network, it is possible to decide that aircraft 2 will be used to operate the flight that was originally scheduled to aircraft 1 and vice versa. This is called an aircraft swap [Jarrah et al., 1993],[Gopalan and Talluri, 1998a], [Aktürk et al., 2014] and it is used in several situations. Sometime this mechanism is used to proactively change the schedule before the day even starts to ensure that a specific aircraft ends at a given station in order to undergo a scheduled maintenance [Lapp and Cohn, 2012]. Some other time, aircraft swaps happen opportunistically during the day to reduce delays.

Consider the example given in Figure 5.3 where we consider 4 flights between 5 airports A, B, C, D and E. In this example, aircraft 1 and 2 are swapped for flights B to E and B to C. This would be useful in the maintenance situation if aircraft 2 had to finish the day at airport E which has the necessary staff and equipment to provide maintenance. Such a swap could also be useful in the case of delays. Suppose that aircraft 1 has a one hour delay on its first leg (A to B). Aircraft 1 lands at airport B at 10:00 instead of 9:00 and its

second leg (B to C) will be delayed as well since it takes time to turn the aircraft. However, aircraft 2 will be ready earlier since it landed at 9:30 so it could operate the flight from B to C while aircraft 1 will be used for the B to E flight. In this scenario, we would not have any second leg delay if we swap aircraft.



(a) Original lines of flight

(b) Swapped lines of flight

Figure 5.3: Example of swapped flights between two aircraft

Aircraft swaps can be advantageous but are limited by several factors. The main constraint is aircraft capacity, we need to make sure that both aircraft have enough capacity to handle passengers in their second leg. Another factor to consider is cockpit and cabin crews: are they staying with their respective scheduled flights or are they swapped as well to stay with their aircraft? The first case might not be possible if the aircraft are different and the crews are not trained to operate both aircraft types. The second case might lead to scheduling issues because crews now potentially end up in different cities at the end of the day. In the context of this project, we assume that crews always stay with their aircraft — so not necessarily to their scheduled flights. Another significant concern is maintenance, the aircraft have changed their lines of flight and so they will end up overnight at different locations, which is a problem in case of scheduled maintenance, as well as incurring different flight miles and numbers of take-offs and landings. Finally, swapping aircraft on the fly is an operational challenge since it means that they have to be towed to a different gate or that passengers need to be directed to a different gate. However the prospect of reducing delays is, in some cases, deemed worth the cost and airline companies decide to swap aircraft. One way to mitigate these limitations is to only allow aircraft swaps between identical aircraft (i.e., same aircraft type), which we assume to be true in this paper.

We enhance the delay propagation model discussed in Section 5.3.2 by adding the option to swap aircraft when it can reduce delay. We follow the same forward procedure, looking at one flight at a time. Consider a flight from A to B that is scheduled to arrive at B at time t_1. If this flight is delayed and the aircraft has a second leg from B to C scheduled to depart at time t_2 later in the day, we look for a candidate aircraft to swap. In this

simulation model we are interested in reducing the total number of delay minute so we only consider as candidates flights scheduled to leave from B after time t_2, otherwise the combined delay for both flight would increase. We also make sure that candidates have the same aircraft type (see compatibility issues describe in the previous paragraph). Since we assume in this variation that crews stay with their aircraft when swaps happen, we then check that swapping would not lead to a violation of their maximum flight and duty time for the crews assuming no subsequent delays (e.g., we do not allow a swap for a three hour flight if a crew only has two hours of flight remaining on their daily clock). If we find a candidate aircraft that meets all of these conditions, we then choose the aircraft with the earliest scheduled arrival at airport B and perform the swap.

5.4 Computational Experiments

In this section we run three different computational experiments to explore the effect of primary delays and other factors on delay propagation. For of all of these computational experiments we assume an average distance-based load factor of 85% (see section 5.2.2.1) as an approximation of the average percentage of occupied seats on each flight and randomly generate the number of passengers by multiplying the capacity of each aircraft by a random number following a normal distribution with average 85% and standard deviation 7.5% (truncated between 60% and 100%). We also set the amount of in-flight delay to follow an arbitrary normal distribution with mean 0 and standard deviation 10: N(0,10) so as to introducing minimal in-flight perturbation since we are focusing on the impact of primary delays on the system.

5.4.1 Effect of Primary Delays

We first study the impact that primary delays have on the system using the Delay Propagation simulation model. we are interested in exploring the relationship between primary and secondary delays. The primary delays will follow a normal distribution with mean 0 and a variable standard deviation but negative values will be set to 0 since we want to consider actual (positive) delays. This means that, on average, half of the flights will not have a primary delay but could be ultimately delayed nonetheless because of secondary (propagated) or in-flight delays. For each value of the primary delay's standard deviation we run the delay propagation model of our simulation tool for 50 trials and report average output metrics across the system. We are specifically interested in primary and secondary delays,

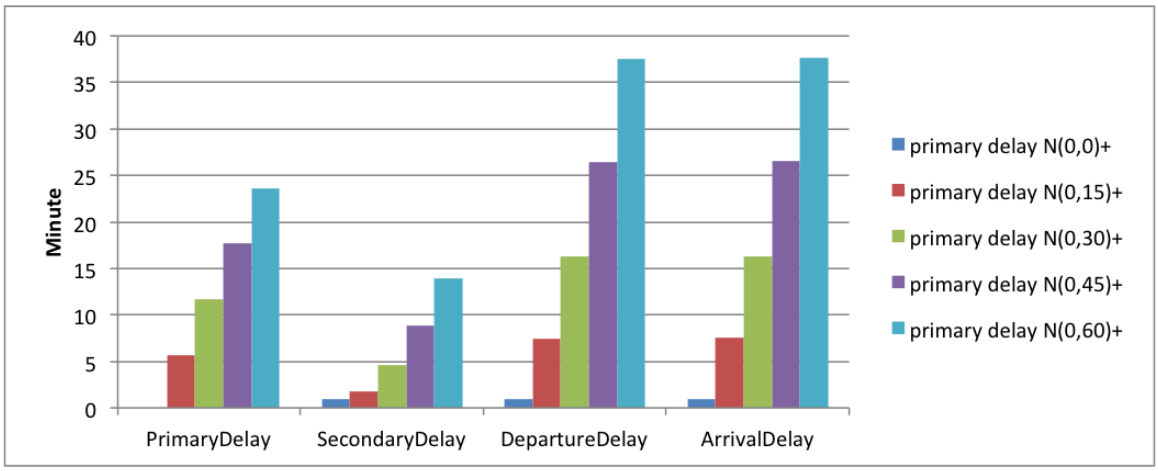


Figure 5.4: Time Based Output Metrics

departure and arrival delays as well as the percentage of flights we arrived 15 minutes or more after their scheduled arrival, the percentage of connecting passengers who missed their connection and the percentage of crews that had to fly pas their maximum flight or duty times. The results are presented on Figures 5.4 and 5.5. As expected, all the metrics increase as we introduce more primary delays in the system with the exception of the percentage of crews who flew past their maximum flight time. This is because the amount of flight time for a crew is independent of primary delays which, by definition, happen on the ground. Flight time is only impacted by in-flight delays which are kept very low in this simulation.

It is interesting to observe that secondary delays occur which means that primary delays propagate in the system. However there is less delay due to propagation than delay introduced via primary delays. this shows that the system is able to absorb some of the delays due to 2 mechanisms: (1) the schedule have built-in buffer in between flight (e.g., an aircraft has 45 minutes scheduled between two successive flights but only need 30 minutes to turn) and to a lesser extent (2) flights are sometimes faster than scheduled, which is modeled in our simulation by a negative in-flight delay, and are able to make up in the air some of their departure delays.

5.4.2 Effect of Complexity in the Schedule: Crew Swaps

Now, we use our simulation tool to study the relationship between complexity of the schedule and delay propagation. We specifically look at how crews are assigned to flights of a daily schedule can impact delay propagation. Consider the simplest case where a crew

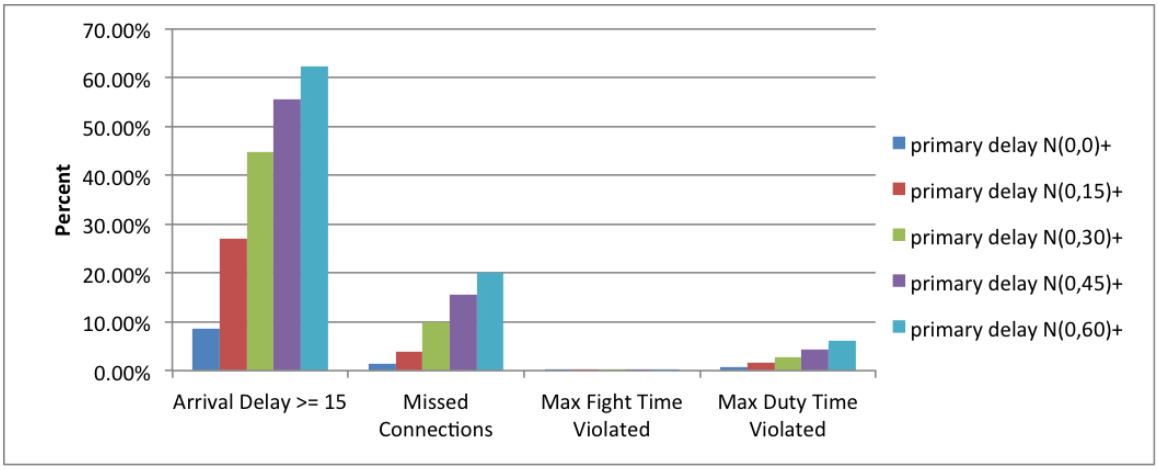
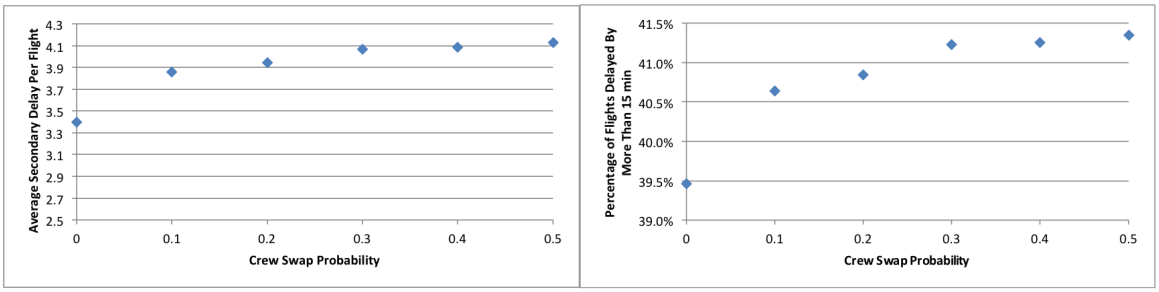


Figure 5.5: Percentage Based Output Metrics

stays with a single aircraft for their entire shift. If one of their flights is late, the delay potentially propagate to the rest of this crew's schedule for the day. Now, consider the case of where the crew changes aircraft in the middle of the day. A delay in a flight before this crew swap can potentially impact two sets of flights: the remaining flights on the crew's schedule but also the subsequent flights using the delayed aircraft. Intuitively, a schedule with a lot of crew swaps is more vulnerable to delay propagation.

Recall from section 5.2.3 that we model crew swaps with a crew swap probability: after assigning a crew to a flight we randomly decide if the crew will carry on the same aircraft or change based on that probability. In this experiment, we change the value of this parameter and run the simulation tool 50 times for each value using the Delay Propagation model. We then report the average secondary (propagated) delay per flight and the percentage of flights that have an arrival delay of more than 15 minutes. Results are presented on Figure 5.6. It appears that schedules generated without crew swaps have a significantly lower amount of propagated delays. We also note that, among the schedules that have crew swaps (crew swap probability > 0), increasing the crew swap probability leads to slightly more delays. We observe a plateau because, after a certain point, there is very little opportunity for additional crew swaps in the schedule , so even if we increase the crew swap probability, the resulting actual schedule will be very similar.



(a) Average secondary delay (minutes) (b) Percent of flights delayed by more than 15 min

Figure 5.6: Effect of crew swap probability on delay propagation

5.4.3 Effect of Adding a Recovery Mechanism: Aircraft Swap

Finally, we explore the potential benefits of using aircraft swaps when facing delays. We run the Aircraft Swap Simulation Model and compare its performance with the Delay Propagation Model on the same input schedule after 20 trials. Table 5.1 contains the results obtained with both simulation models. Out of 2598 flights, the Aircraft Swap Model created 112 swaps on average because it identified an opportunity to reduce delays. This resulted in about one less minute of delay per flight. At the flight level this does not seem like much, but if we consider the savings at the network level, this represents almost 2600 minutes of delay saved for a single day. As a consequence, the percentage of flights delayed by more than 15 minutes and the percentage of connecting passengers who missed their connection is slightly lower when aircraft swaps are allowed. However, one drawback of introducing aircraft swap is that we roughly have twice as many violation of the maximum duty time per crew constraints. When creating a schedule, each crew has a scheduled total duty time that fits that constraint, however, when dynamically making aircraft swaps during the day we modify these crew schedules and potentially introduce violations. As explained in Section 5.3.3, we attempt to reduce this issue when looking for candidate by only considering crews that have enough time remaining on their clock to operate the rest of the swapped line of flight assuming no future delays. The problem is that it is often the case that delays occur which result in violation of both crews maximum duty time constraints. Better predicting delays could help making more informed decisions when swapping aircraft and crews and limit the number of violation caused by the swaps.

Table 5.1: Comparison of Delay Propagation And Aircraft Swap models

Simulation model	**Number of flights**	**Number of swaps**	**Average primary delay**	**Average secondary delay**
Delay Propagation	2598	0	11.84	3.78
Aircraft Swap	2598	112	11.84	2.74
Simulation Model	**Average Arrival Delay**	**Arrival Delay >= 15**	**Missed Connections**	**Max Duty Violated**
Delay Propagation	15.6	40%	10.0%	1.77%
Aircraft Swap	14.6	38%	9.7%	3.89%

5.5 Incorporating Downstream Disruptions: Future Research

The work described in this paper is the beginning of a larger project in which we would like to account for possible future perturbations during the recovery decision making process. As illustrated by the motivating example of Figure 5.1, knowledge of potential future delays could influence how we respond to earlier delays. In order to capture and evaluate this potential, we see two main avenues of future research. Firstly, better understanding the correlation between delays at different points in time and different airports which would ultimately lead to getting estimates of the probability of future disruptions in the network based on the current state of the system. And secondly, developing recovery strategies that use this additional information to be more effective.

5.5.1 Correlation Between Delays

So far, our simulation tool uses independent and identically distributed truncated normal distributions to generate the primary delays that we introduce in the system. However, looking at historical delays show that we can have a better idea of future perturbations by looking at the current state of the system than by simply looking at averages.

We run a simple experiment to evaluate the potential relationship between delays at two given airports. We specifically study delays at Detroit airport (DTW) given knowledge of delays at Atlanta airport (ATL). For the purpose of this experiment, we look at buckets of one hour and define a perturbation as an hour that has 20% or more of flights with a departure delay of 15 minutes or more. We looked at year 2013 and reported the percentage of days that had a perturbation for the corresponding time slot at DTW on Table 5.2. The second line of the table shows the proportion of days with a perturbation at DTW for each

Table 5.2: Percenatage of days with a perturbation at DTW by hour

Time	7-8	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18
Unconditional	14.5%	11.6%	31.6%	20.5%	41.6%	26.7%	34.9%	44.0%	38.8%	58.2%	46.2%
Conditional on ATL perturbation											
t = 0	38.9%	44.4%	47.3%	47.1%	55.7%	50.8%	56.5%	62.3%	57.0%	68.2%	59.6%
t = -1		47.2%	63.0%	45.3%	52.9%	49.8%	62.7%	65.3%	56.4%	70.3%	60.5%
t = -2			63.9%	70.4%	51.4%	52.9%	59.8%	72.9%	59.7%	68.8%	63.2%
t = -3				66.7%	59.3%	52.7%	62.7%	66.7%	68.9%	73.4%	62.0%

hour across the year. For instance, 14.5% of the days had 20% or more outbound flights delayed by more than 15 minutes from 7AM to 8AM compared to 58.2% of the time for the 4-5PM bucket. We also note that the likelihood of having a perturbation increases throughout the day. This is a typical observation when studying airlines performance: more delays are expected in the afternoon than in the morning. The main reason is that each day starts fresh, with typically all aircraft being available on time. Then, as the day progresses, delays occur and start propagating until the end of the day.

We then do the same analysis for flights leaving Atlanta and report the conditional probabilities of having a perturbation in DTW given that ATL had a perturbation at the same time or 1, 2 or 3 hours before. Let's look at the 10-11AM bucket: without any additional information we can only say that the probability of having a perturbation at DTW is 20.5%. However, if we know that ATL has a perturbation for that 10-11 bucket, then the probability of having a perturbation in DTW jumps to 47.1%. Now, if we observe a perturbation at ATL for the bucket 9-10 AM then we know that the probability of having a perturbation in DTW one hour later is 45.3%.

Generating this kind of tables can give us insight on the amount of correlation between delays at different airports and help better predicting future delays based on the current state of the network.

5.5.2 Recovery Under Uncertainty

Once we are able to estimate the probability of having future delays based on the current state of the system, we can use this information to make better recovery decisions. A simple enhancement of the aircraft swap model would be to include the expected primary delay of the candidate flight that we are considering swapping with. This will prevent swapping with a flight that has a high risk of being delayed as well. This could also help better estimating passenger missed connections (recall example form Figure 5.1) to prevent making unnecessary swaps when the entire line of flight is delayed.

Another important feature will be to generate primary delays based on historical distri-

bution instead of using iid. distributions. This will allow us to model complex interactions in the network such as correlations in delays. For instance, bad weather in Chicago might often be followed by bad weather in Detroit a few hours later which would cause similar primary delays.

As part of future research, we would also like to develop other recovery mechanisms, such as cancellation or ground delay program and compare key performance indicator metrics with the baseline Delay Propagation Model.

CHAPTER 6

Conclusion

In this dissertation, we studied four scheduling problems found in the industry. In each of these four problems, we analyzed how uncertainty in parameters impacts the quality and reliability of the schedules and proposed one or several approaches to generate better solutions.

In Chapter 2, we considered the problem of assigning flights to gates to reduce the impact of gate blockage, which occur because of variability in the system: flights leaving/arriving early or late from/to their destination gate. Analysis of historical data showed that about 5% of US domestic flights experience a gate blockage. We develop a network flow optimization model to create gate assignments with a lower expected amount and duration of gate blockage. The schedules obtained from this approach were significantly outperforming a schedule created using a first in first out paradigm. This project showed how considering uncertainty when making scheduling decisions can help mitigating the negative effects of variability during operations. Ideas for future research include enhancing our model with adjacency constraints and developing other techniques to estimate the probability of gate blockage between two flights.

In Chapter 3, we explicitly considered recourse decisions in a patient scheduling setting. We developed an optimization model to fine tune patient appointment times at a chemotherapy infusion center as well as dynamically assign patient to an infusion chair as uncertainty in treatment times is realized. This allows to account for delay propagation throughout the day, which was ignored in Chapter 2. Our objective was to minimize a linear combination of expected patient wait time and expected total length of operations in a two-stage optimization model containing one scheduling phase to set patient appointment times and one recourse phase to assign resource to patients throughout the day. The next step for this project would be to include patient sequencing to further improve the quality

of the appointment schedules.

Chapter 4 considered the problem of scheduling downloads during a small satellite mission. The main difference with chapter 3 is that this problem is dynamic by nature and involve a multi-stage decision tree, instead of a simpler two-stage approach. The mission planning horizon was discretized in time intervals, each representing an opportunity to download to a ground station. When making decisions on how much data to download in each interval, we do not know for sure whether the ground station is available to receive data or not. We modeled and compared the performance of two different types of satellite in term of the expected amount of data they could download during the mission. We showed the value of being able to dynamically adjust the download schedule during the mission, using the knowledge of the realization of uncertainty in previous intervals. One of the main challenges of this approach is the computational complexity required to solve the large scale optimization model. We could address this as part of future research by developing equivalent formulations and approximation algorithms.

In Chapter 5, we studied the concept of recovery in the context of delay propagation in an airline network. The difference between recovery and recourse is that the cost of recourse is directly accounted for in the scheduling phase while recovery decisions are dynamically made during the day. In this project, we designed a data generation scheme and a simulation framework to compare different recovery strategies such as swapping aircraft between two flights. We introduced primary delays in the system and compared the amount of delays that is propagated throughout the day by aircraft and crews. This simulation framework is a good fit to design and implement other recovery strategies in the future. We would also like to model the correlation of delays between different airport to better represent the interactions that impact delay propagation.

These four projects all present examples of uncertainty in real-world scheduling problems. We showed that modeling this variability can improve the quality of the schedules generated but usually increases the complexity of the problem. The stochastic counterpart of a deterministic optimization model is typically much harder to solve. We described several tractable approaches to solve these problems such as exploiting a network flow structure in Chapter 2 and the Fix-Unfix heuristic of Chapter 3. In the application to aerospace of Chapter 4, we showed that the computational complexity of the model grew exponentially with the length of the planning horizon. This curse of dimensionality is a very common challenge in multi-stage stochastic optimization models. We plan to study a rolling horizon

method to simplify the problem in future research.

Another challenge of introducing uncertainty into deterministic scheduling problems lies in the definition of objective functions that could take many different forms. In the gate assignment project, we used historical data to compute the probability of conflict between any two pairs of flights. We then directly used these values in the objective. We took a wildly different approach in Chapter 3 and 4 by defining scenarios and optimizing expected outcomes across all these scenarios. While this scenario-based approach allows to model more of the granularity of the problem such as recourse decisions in each individual scenario, it typically leads to large scale models that are much harder to solve. The trade-off between designing a model that is close to reality but still solvable in a reasonable time is at the center of the scheduling under uncertainty research topic.

BIBLIOGRAPHY

[lat,] http://openflights.org/data.html. [Online; accessed 10 December 2016].

[FAA,] http://registry.faa.gov/aircraftinquiry/NNum_Inquiry.aspx. [Online; accessed 10 December 2016].

[BTS, a] https://www.bts.gov/. [Online; accessed 10 December 2016].

[Del,] http://www.delta.com/content/www/en_US/about-delta/corporate-information/aircraft-fleet.html. [Online; accessed 10 December 2016].

[BTS, b] http://www.transtats.bts.gov/Data_Elements.aspx?Data=5. [Online; accessed 10 December 2016].

[Abdelghany et al., 2004] Abdelghany, A., Ekollu, G., Narasimhan, R., and Abdelghany, K. (2004). A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research*, 127(1-4):309–331.

[Aboumater et al., 2008] Aboumater, H., Winner, L. E., Davis, R. O., Trovitch, P. B., Berg, M. M., Violette, K. M., Messersmith, W. A., Maylor, K. K., and Lehmann, C. U. (2008). No time to waste: decreasing patient wait times for chemotherapy administration using automated prioritization in an oncology pharmacy system. *The American journal of managed care*, 14(5):309–316.

[AhmadBeygi et al., 2008] AhmadBeygi, S., Cohn, A., Guan, Y., and Belobaba, P. (2008). Analysis of the potential for delay propagation in passenger airline networks. *Journal of Air Transport Management*, 14(5):221–236.

[Ahmadbeygi et al., 2010] Ahmadbeygi, S., Cohn, A., and Lapp, M. (2010). Decreasing airline delay propagation by re-allocating scheduled slack. *IIE Transactions*, 42(7):478–489.

[Ahmed et al., 2002] Ahmed, S., Shapiro, A., and Shapiro, E. (2002). The sample average approximation method for stochastic programs with integer recourse. *Submitted for publication*.

[Ahmed et al., 2011] Ahmed, Z., ElMekkawy, T., and Bates, S. (2011). Developing an efficient scheduling template of a chemotherapy treatment unit: A case study. *The Australasian medical journal*, 4(10):575.

[Aktürk et al., 2014] Aktürk, M. S., Atamtürk, A., and Gürel, S. (2014). Aircraft rescheduling with cruise speed control. *Operations Research*, 62(4):829–845.

[Baker and Worden, 2008] Baker, D. N. and Worden, S. P. (2008). The large benefits of small-satellite missions. *EOS, Transactions American Geophysical Union*, 89(33):301–302.

[Barnhart et al., 2003] Barnhart, C., Cohn, A., Johnson, E., Klabjan, D., Nemhauser, G., and Vance, P. (2003). Airline crew scheduling. *Chapter Airline Crew SchedulingSpringer, New York*, pages 517–560.

[Barnhart and Talluri, 1997] Barnhart, C. and Talluri, K. T. (1997). Airline operations research. *Design and Operations of Civil and Environmental Engineering Systems*, pages 435–469.

[Beatty et al., 1999] Beatty, R., Hsu, R., Berry, L., and Rome, J. (1999). Preliminary evaluation of flight delay propagation through an airline schedule. *Air Traffic Control Quarterly*, 7(4):259–270.

[Bertsimas and Tsitsiklis, 1997] Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA.

[Birge and Louveaux, 2011a] Birge, J. R. and Louveaux, F. (2011a). *Introduction to stochastic programming*. Springer Science & Business Media.

[Birge and Louveaux, 2011b] Birge, J. R. and Louveaux, F. (2011b). *Introduction to stochastic programming*. Springer Science & Business Media.

[Borndörfer et al., 2010] Borndörfer, R., Dovica, I., Nowak, I., and Schickinger, T. (2010). Robust tail assignment. In *Proceedings of the fiftieth annual symposium of AGIFORS*.

[Burke et al., 2010] Burke, E. K., De Causmaecker, P., De Maere, G., Mulder, J., Paelinck, M., and Vanden Berghe, G. (2010). A multi-objective approach for robust airline scheduling. *Computers & Operations Research*, 37(5):822–832.

[Burke, 1956] Burke, P. J. (1956). The output of a queuing system. *Operations research*, 4(6):699–704.

[Castaing, 2014] Castaing, J. (2014). Scheduling downloads for multi-satellite, multi-ground station missions.

[Cayirli et al., 2006] Cayirli, T., Veral, E., and Rosen, H. (2006). Designing appointment scheduling systems for ambulatory care services. *Health Care Management Science*, 9(1):47–58.

[Cheng et al., 2012] Cheng, C.-H., Ho, S. C., and Kwan, C.-L. (2012). The use of meta-heuristics for airport gate assignment. *Expert Systems with Applications*.

[Coffman et al., 1980] Coffman, E. G., So, K., Hofri, M., and Yao, A. (1980). A stochastic model of bin-packing. *Information and Control*, 44(2):105–115.

[Cohn and Lapp, 2010] Cohn, A. and Lapp, M. (2010). Airline resource scheduling. *Wiley Encyclopedia of Operations Research and Management Science*.

[Denton et al., 2007] Denton, B., Viapiano, J., and Vogl, A. (2007). Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health care management science*, 10(1):13–24.

[Denton et al., 2010] Denton, B. T., Miller, A. J., Balasubramanian, H. J., and Huschka, T. R. (2010). Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations research*, 58(4-part-1):802–816.

[Ding et al., 2005] Ding, H., Lim, A., Rodrigues, B., and Zhu, Y. (2005). The over-constrained airport gate assignment problem. *computers & operations research*, 32(7):1867–1880.

[Dobish, 2003] Dobish, R. (2003). Next-day chemotherapy scheduling: a multidisciplinary approach to solving workload issues in a tertiary oncology center. *Journal of Oncology Pharmacy Practice*, 9(1):37–42.

[Dohse, 2007] Dohse, L. K. (2007). Nine ways to improve efficiency in an ambulatory infusion center. *Community Oncology*, 4(1):33–34.

[Eggenberg et al., 2010] Eggenberg, N., Salani, M., and Bierlaire, M. (2010). Constraint-specific recovery network for solving airline recovery problems. *Computers & operations research*, 37(6):1014–1026.

[Erdogan and Denton, 2013] Erdogan, S. A. and Denton, B. (2013). Dynamic appointment scheduling of a stochastic server with uncertain demand. *INFORMS Journal on Computing*, 25(1):116–132.

[Erdogan et al., 2015] Erdogan, S. A., Gose, A., and Denton, B. T. (2015). On-line appointment sequencing and scheduling. *IIE Transactions*, (just-accepted):00–00.

[Erikson et al., 2007] Erikson, C., Salsberg, E., Forte, G., Bruinooge, S., and Goldstein, M. (2007). Future supply and demand for oncologists: challenges to assuring access to oncology services. *Journal of Oncology Practice*, 3(2):79–86.

[Filar et al., 2001] Filar, J. A., Manyem, P., and White, K. (2001). How airlines and airports recover from schedule perturbations: a survey. *Annals of Operations Research*, 108(1-4):315–333.

[Fleurquin et al., 2013] Fleurquin, P., Ramasco, J. J., and Eguiluz, V. M. (2013). Systemic delay propagation in the us airport network. *Scientific reports*, 3.

[Genç et al., 2012] Genç, H. M., Erol, O. K., Eksin, İ., Berber, M. F., and Güleryüz, B. O. (2012). A stochastic neighborhood search approach for airport gate assignment problem. *Expert Systems with Applications*, 39(1):316–327.

[Gopalan and Talluri, 1998a] Gopalan, R. and Talluri, K. T. (1998a). The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271.

[Gopalan and Talluri, 1998b] Gopalan, R. and Talluri, K. T. (1998b). Mathematical models in airline schedule planning: A survey. *Annals of Operations Research*, 76:155–185.

[Gul et al., 2011] Gul, S., Denton, B. T., Fowler, J. W., and Huschka, T. (2011). Bi-criteria scheduling of surgical services for an outpatient procedure center. *Production and Operations management*, 20(3):406–417.

[Gupta and Denton, 2008] Gupta, D. and Denton, B. (2008). Appointment scheduling in health care: Challenges and opportunities. *IIE transactions*, 40(9):800–819.

[Itano and Taoka, 2005] Itano, J. and Taoka, K. N. (2005). *Core curriculum for oncology nursing*.

[Jarrah et al., 1993] Jarrah, A. I., Yu, G., Krishnamurthy, N., and Rakshit, A. (1993). A decision support framework for airline flight cancellations and delays. *Transportation Science*, 27(3):266–280.

[Kim and Feron, 2011] Kim, S. H. and Feron, E. (2011). Robust gate assignment. *AIAA Guidance, Navigation, and Control Conference*.

[Klabjan et al., 2001] Klabjan, D., Johnson, E. L., Nemhauser, G. L., Gelman, E., and Ramaswamy, S. (2001). Airline crew scheduling with regularity. *Transportation Science*, 35(4):359–374.

[Kleywegt et al., 2002] Kleywegt, A. J., Shapiro, A., and Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

[Kohl et al., 2007] Kohl, N., Larsen, A., Larsen, J., Ross, A., and Tiourine, S. (2007). Airline disruption management—perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162.

[Lan et al., 2006] Lan, S., Clarke, J.-P., and Barnhart, C. (2006). Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28.

[Lapp, 2012] Lapp, M. (2012). *Methods for Improving Robustness and Recovery in Aviation Planning*. PhD thesis, The University of Michigan.

[Lapp et al., 2008] Lapp, M., AhmadBeygi, S., Cohn, A., and Tsimhoni, O. (2008). A recursion-based approach to simulating airline schedule robustness. In *Proceedings of the 40th Conference on Winter Simulation*, pages 2661–2667. Winter Simulation Conference.

[Lapp and Cohn, 2012] Lapp, M. and Cohn, A. (2012). Modifying lines-of-flight in the planning process for improved maintenance robustness. *Computers & Operations Research*, 39(9):2051–2062.

[Lettovskỳ et al., 2000] Lettovskỳ, L., Johnson, E. L., and Nemhauser, G. L. (2000). Airline crew recovery. *Transportation Science*, 34(4):337–348.

[Li, 2008] Li, C. (2008). Airport gate assignment: New model and implementation. *arXiv preprint arXiv:0811.1618*.

[Lim and Wang, 2005] Lim, A. and Wang, F. (2005). Robust airport gate assignment. In *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*, pages 8–pp. IEEE.

[Mancilla and Storer, 2012] Mancilla, C. and Storer, R. (2012). A sample average approximation approach to stochastic appointment sequencing and scheduling. *IIE Transactions*, 44(8):655–670.

[Masselink et al., 2012] Masselink, I. H., van der Mijden, T. L., Litvak, N., and Vanberkel, P. T. (2012). Preparation of chemotherapy drugs: Planning policy for reduced waiting times. *Omega*, 40(2):181–187.

[Mazier et al., 2010] Mazier, A., Billaut, J.-C., and Tournamille, J.-F. (2010). Scheduling preparation of doses for a chemotherapy service. *Annals of Operations Research*, 178(1):145–154.

[Min and Yih, 2010] Min, D. and Yih, Y. (2010). Scheduling elective surgery under uncertainty and downstream capacity constraints. *European Journal of Operational Research*, 206(3):642–652.

[Pereira and Pinto, 1991] Pereira, M. V. and Pinto, L. M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375.

[Petkov and Maranas, 1997] Petkov, S. B. and Maranas, C. D. (1997). Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty. *Industrial & engineering chemistry research*, 36(11):4864–4881.

[Pinedo, 2012] Pinedo, M. L. (2012). *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media.

[Rosenberger et al., 2003] Rosenberger, J. M., Johnson, E. L., and Nemhauser, G. L. (2003). Rerouting aircraft for airline recovery. *Transportation Science*, 37(4):408–421.

[Rosenberger et al., 2004] Rosenberger, J. M., Johnson, E. L., and Nemhauser, G. L. (2004). A robust fleet-assignment model with hub isolation and short cycles. *Transportation Science*, 38(3):357–368.

[Ruszczyński, 1997] Ruszczyński, A. (1997). Decomposition methods in stochastic programming. *Mathematical programming*, 79(1-3):333–353.

[Sadki et al., 2011] Sadki, A., Xie, X., and Chauvin, F. (2011). Appointment scheduling of oncology outpatients. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 513–518. IEEE.

[Santibáñez et al., 2012] Santibáñez, P., Aristizabal, R., Puterman, M. L., Chow, V. S., Huang, W., Kollmannsberger, C., Nordin, T., Runzer, N., and Tyldesley, S. (2012). Operations research methods improve chemotherapy patient appointment scheduling. *Joint Commission Journal on Quality and Patient Safety*, 38(12):541–541.

[Schaefer et al., 2005] Schaefer, A. J., Johnson, E. L., Kleywegt, A. J., and Nemhauser, G. L. (2005). Airline crew scheduling under uncertainty. *Transportation Science*, 39(3):340–348.

[Şeker and Noyan, 2012] Şeker, M. and Noyan, N. (2012). Stochastic optimization models for the airport gate assignment problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(2):438–459.

[Sevinc et al., 2013] Sevinc, S., Sanli, U. A., and Goker, E. (2013). Algorithms for scheduling of chemotherapy plans. *Computers in biology and medicine*, 43(12):2103–2109.

[Singprasong and Eldabi, 2013] Singprasong, R. and Eldabi, T. (2013). An integrated methodology for process improvement and delivery system visualization at a multidisciplinary cancer center. *Journal for Healthcare Quality*, 35(2):24–32.

[Society,] Society, A. C. How does chemotherapy work? 2014. http://blog.dana-farber.org/insight/2014/11/how-does-chemotherapy-work/. [Online; accessed 22-April-2015].

[Spangelo et al., 2015] Spangelo, S., Cutler, J., Gilson, K., and Cohn, A. (2015). Optimization-based scheduling for the single-satellite, multi-ground station communication problem. *Computers & Operations Research*, 57:1–16.

[Sriram and Haghani, 2003] Sriram, C. and Haghani, A. (2003). An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A: Policy and Practice*, 37(1):29–48.

[Tanaka, 2012] Tanaka, T. (2012). *Infusion chair scheduling algorithms based on bin-packing heuristics.* ProQuest/UMI.

[Teodorovic and Stojkovic, 1995] Teodorovic, D. and Stojkovic, G. (1995). Model to reduce airline schedule disturbances. *Journal of Transportation Engineering*, 121(4):324–331.

[Thengvall et al., 2000] Thengvall, B. G., Bard, J. F., and Yu, G. (2000). Balancing user preferences for aircraft schedule recovery during irregular operations. *Iie Transactions*, 32(3):181–193.

[Tu et al., 2008] Tu, Y., Ball, M. O., and Jank, W. S. (2008). Estimating flight departure delay distributionsa statistical approach with long-term trend and short-term pattern. *Journal of the American Statistical Association*, 103(481):112–125.

[Turkcan et al., 2012] Turkcan, A., Zeng, B., and Lawley, M. (2012). Chemotherapy operations planning and scheduling. *IIE Transactions on Healthcare Systems Engineering*, 2(1):31–49.

[Wang and Ahmed, 2008] Wang, W. and Ahmed, S. (2008). Sample average approximation of expected value constrained stochastic programs. *Operations Research Letters*, 36(5):515–519.

[Woodall et al., 2013] Woodall, J. C., Gosselin, T., Boswell, A., Murr, M., and Denton, B. T. (2013). Improving patient access to chemotherapy treatment at duke cancer institute. *Interfaces*, 43(5):449–461.

[Xu et al., 2008] Xu, N., Sherry, L., and Laskey, K. (2008). Multifactor model for predicting delays at us airports. *Transportation Research Record: Journal of the Transportation Research Board*, (2052):62–71.

[Yan and Yang, 1996] Yan, S. and Yang, D.-H. (1996). A decision support framework for handling schedule perturbation. *Transportation Research Part B: Methodological*, 30(6):405–419.

[Yen and Birge, 2006] Yen, J. W. and Birge, J. R. (2006). A stochastic programming approach to the airline crew scheduling problem. *Transportation Science*, 40(1):3–14.