Methods for Improving Robustness and Recovery in Aviation Planning

by

Marcial Lapp

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
2012

Doctoral Committee:

Associate Professor Amy Cohn, Chair
Professor Mark Daskin
Professor Edwin Romeijn
Professor Thomas Schriber
Sergey Shebalov, Sabre Inc.

ACKNOWLEDGEMENTS

This thesis is the culmination of my career at the University of Michigan, which would not have been possible without the help and encouragement from some very important people.

First and foremost, I would like to acknowledge, Amy Cohn, my advisor, mentor and friend for the past five years in the Ph.D. program in the Industrial & Operations Engineering department. I recall our first meeting in 2007, at which we discussed a programming project on simulating airline delay propagation. It was this research endeavor that sparked my interest in the airline industry. Amy worked tirelessly with me, providing guidance and ideas to further my research goals.

Another person that contributed to my success in the program is Shervin Ahmadbeygi. Shervin invited me to join as a research assistant on his delay-propagation project and provided me with an introduction to large-scale airline optimization models. Without his explanations, patience, inspiration and guidance, I would not be where I am today.

I would also like to thank my committee: Prof. Edwin Romeijn, Prof. Mark Daskin, Prof. Tom Schriber and Dr. Sergey Shebalov. Their patience and positive feedback has been invaluable in developing the ideas in this thesis. In addition, I would like to specifically acknowledge Dr. Sergey Shebalov, who participated in weekly conference calls and inspired the third chapter of this dissertation. I would

also like to acknowledge the friendly folks at Southwest Airlines, including Lonny Hurwitz and Alex Heinold, who helped to formalize the ideas in the second part of this thesis.

There are also quite a few graduate students inside and outside the Industrial & Operations Engineering department that have made the graduate student experience all worthwhile. Thanks go to my roommates for a number of years: Brendan See and T. Jeff Fleszar. In addition, I would like to acknowledge Jonathan Helm, Gregory King, Katrina Appell, Hoda Parvin and Fang Dang for making life as a graduate student quite pleasant. Finally, I would like to acknowledge the person that I spent countless hours with, my office-mate Arleigh Waring. To this day, I wonder how she was able to put up with me.

In addition to my research, I have also been teaching a number of courses at the University of Michigan. During my junior year as an undergraduate, it was Mary-Lou Dorf who offered me a position as a teaching assistant in her EECS class. It was through this appointment that I discovered my passion for teaching. I must thank James Holloway for bringing me in as a teaching assistant for ENGR 101, and Mark Daskin for having the faith to place me in front a class of 150 undergraduate students in IOE 202. Finally, I would not have been able to teach these courses without the help of Tina Blay and Wanda Dobberstein, who helped me deal with all of the student-related issues.

Along a similar path, I would like to acknowledge Jeff Ringenberg for his continued partnership in our Engineering Education endeavors. Sponsoring a number of undergraduate research assistants over the course of the past four years has been fun and taught me a lot of managing student teams effectively. In addition, I think

we have made quite an impact for computer-related learning and teaching at the University of Michigan.

Finally, I owe a great deal of gratitude to my family, especially my parents and my sister. Michael, Almut and Marisa have always been immensely supportive and I am forever indebted to their assistance and encouragement throughout my time as a student at the University of Michigan.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

CHAPTER I

Introduction

In this dissertation, we develop new methods for improving robustness and recovery in aviation planning. In addition to these methods, the contributions of this dissertation include an in-depth analysis of several mathematical modeling approaches and proof of their structural equivalence. Furthermore, we analyze various decomposition approaches, the difference in their complexity and the required computation time to ultimately provide insight into selecting the most appropriate formulation for a particular problem structure. For our work, we focus on robustness and recovery problems from the airline industry. All of our computational experiments are based on major US carrier data.

This dissertation is organized as follows. To begin, in Chapter II, we provide an overview of the airline planning process. This chapter begins with key operational problems, including fleet assignment, aircraft routing and crew scheduling. These problems are covered in detail, including common mathematical modeling variations and solution approaches. Beyond these core problems, this chapter also provides a broad overview of different areas of research that have been explored within this industry. In addition, this chapter presents the necessary background information for the remaining chapters of this dissertation.

In Chapter III, we extend the work of [8] where the authors improve flight schedule robustness through the reallocation of slack within the flight network. The optimization model in this work, while effective, does not capture the situation when delay is propagated by crew and aircraft to subsequently different flights. We develop a tail-recursive algorithm to simulate a delay propagation network to provide a measure of robustness for comparison between different flight schedules.

In Chapter IV we apply the concept of robustness to aircraft maintenance planning. In this planning problem, we improve maintenance robustness, i.e a plan's ability to withstand unforeseen changes. We do so by allocating maintenance rotations to those aircraft that will most likely benefit from the assignment. To assess the effectiveness of our approach, we introduce a new metric, *maintenance reachability (MR)*, which measures the robustness of a planned set of rotation assignments and their ability to bring aircraft to maintenance stations as necessary. In addition, we develop a mathematical programming approach to improve the MR of the rotations assigned to aircraft during the maintenance planning phase.

We continue on the path of aircraft maintenance in Chapters V and VI, transitioning from maintenance planning to maintenance recovery. On the day-of-operations, disruptions often take place and change aircraft rotations and their respective maintenance assignments. In recovery, we focus on creating feasible plans after such disruptions have occurred. We divide our recovery approach into two phases. In the first phase, in Chapter V, we solve the *Maintenance Recovery Problem (MRP)*, a computationally complex, short-term, recovery problem. We begin by presenting a mixed-integer program to maximize the number of maintenance events that can be completed by their fixed deadlines. We consider two secondary objective functions

to differentiate between equivalent solutions to the MRP. In the first, we seek to minimize the earliness of maintenance events (i.e. not performing maintenance any earlier than is required to be in compliance with regulatory policy, thus reducing long-term maintenance costs). In the second, we seek to evenly distribute maintenance events across days and across stations for balanced utilization of capacity. We formulate these problem through various mathematical formulations and employ several decomposition approaches to solve these complex problems.

The problem considered in Chapter V is a simplification of the real-world problem, in that we only consider the first (i.e. next due) maintenance event of each type. This research provides us with insights, used in Chapter VI to solve the more realistic (and more challenging) problem in which we consider not only the first, but subsequent maintenance events as well. In this problem, the decision of scheduling the day and location for a maintenance event has impact on all downstream events. To solve the problem of maintenance recovery including subsequent events, we begin by formulating several mathematical models, extending the foundations developed in Chapter IV. We subsequently solve and compare these models for their respective effectiveness in terms of solution speed and quality. Through the development of our solution approach, we are able to solve the recurring maintenance recovery problem for a medium-size airline in less fifteen minutes. Finally, we extend the recurrent maintenance problem further by increasing the time horizon over which this problem is solved. We present an extension of our algorithm to solve the maintenance recovery problem over time horizons as long as three weeks.

CHAPTER II

Overview of Major Airline Processes

2.1 Introduction to the Airline Industry

Passenger aviation is critical to today's society, with passengers relying on airlines (carriers) to provide safe, reliable, and affordable travel for both business and leisure. In 2009, more than 9.9 million flights originated and/or terminated in just the U.S. alone, carrying more than 764 million passengers [28]. Every one of these flights required the coordinated utilization of many shared resources including aircraft, crews (cockpit, cabin, and ground), taxiways and runways, the airspace, and more. In some cases, resources are shared across multiple flights within a single company (e.g. aircraft, crews) while other resources (such as runways and the airspace) must be shared across airlines, adding further complexity. This sharing of resources, along with the associated underlying network structure of an airline, results in significant coordination challenges.

The operations research (OR) community has long played an active role in virtually all aspects of the airline industry, helping to plan, schedule, coordinate, and operate it. In the past decade, this role has been particularly important. Major challenges such as SARS, the U.S. terrorist attacks of 9/11, the 2008 spikes in fuel prices, and a global economic downturn have made it increasingly important that airlines

utilize resources efficiently. To accommodate, the OR community has expanded its focus to include topics such as robust planning, integrated planning, and enhanced recovery techniques.

In the process of solving these challenging airline problems, the OR community has also made broader contributions. Specifically, several of the modeling and algorithmic techniques developed to solve airline planning problems have applicability to a broad class of other application areas as well.

This chapter of this thesis has two purposes. The first is to introduce readers new to the field of airline operations research to the problems that have been solved and the problems currently under investigation, and to provide initial references to some of the key literature. The second is to review some of the key modeling and algorithmic contributions, which serve as a cornerstone to the mathematical models that presented in the remainder of this thesis.

2.2 Operations Research Problems in the Airline Industry

Within passenger aviation, there is a vast array of complex decisions to be made, ranging from aircraft design and airport construction to the control of the airspace to airline planning and scheduling. Similarly, there is a wide range of decision makers, including carriers, government regulators, airport authorities, and passengers. We focus here on resource scheduling, from the perspective of the carriers. Resource scheduling problems range in time-scale from years, such as the decision to purchase an aircraft, to minutes, such as deciding how to re-accommodate passengers who have missed their connections. These problems cover many different resources including aircraft, pilots, flight attendants, gates, baggage, maintenance workers and facilities, and — of course — passengers. In addition, they must all address underly-

ing uncertainty, including variability in demand, inclement weather, and unexpected maintenance issues. Furthermore, these problems must all address the system complexity associated with the underlying network structure of airline systems and the sharing of a finite set of resources.

Within the set of carrier resource scheduling problems, we focus primarily on the established literature in *fleet assignment*, *crew scheduling*, and *aircraft routing*, as well as the emerging literature on integrated planning and robust planning. We also briefly touch about important future areas of research. Before doing so, we first briefly highlight some other important areas of airline OR. These references are not intended to be an exhaustive survey, but rather to give a sense of the wide range of work that has been done and some initial sources for the interested reader.

- **Schedule generation**: Airlines work on the *schedule generation* process a year or more in advance of the day-of-operations, predicting the demand for flights between origin-destination (O-D) pairs and subsequently deciding which flights to offer and with what frequency, as well as how to partner with other airlines through code-sharing and partnering agreements. [54] provide a survey paper identifying various schedule generation strategies. Early work by [36] provides a Lagrangian relaxation to solve the assignment of aircraft to routes. In [99] and [61], the authors provide a dynamic scheduling model that uses changes in the fleet assignment and minor flight re-timings to update the schedule as booking data becomes available.

- **Revenue management, pricing and passenger flow models**: *Revenue management* and *pricing* problems focus on strategies to maximize profits from ticket sales. This is an evolving field of study, especially as new purchasing

channels and new information systems become available. Related work can be found in [73], [91], [21], [47], and [68]. As a bridge between revenue management and fleet assignment, passenger flow modeling [20] finds the optimal (i.e. revenue-maximizing) selection from a set of candidate itineraries given a fixed set of flight capacities. This solution, idealized in the sense that it assumes that an airline has complete control over which itineraries are purchased, provides a bound on the revenue that can be generated from a given fleeted schedule. More recently, [42] has incorporated uncertain demand and spill estimates within passenger flow modeling.

- **Demand driven dispatch and dynamic fleeting**: Even though airline plans are set far in the future, they are subject to uncertainty until the day of operations. Early work by [24] suggests that airlines can benefit from dynamically adjusting their fleet assignment to better match aircraft capacity to passenger demand as updated information about passenger bookings is obtained. See [84] and [93] for a more recent discussion of this topic. As an extension to demand driven dispatch, two recent articles [99] and [61] explore the idea of not only modifying the fleet assignment, but also slightly altering the flight schedule itself as well to increase revenue in response to evolving information about passenger demand.

- **Recovery**: Airline plans are rarely, if ever, executed as designed. Unexpected disruptions such as inclement weather and unplanned maintenance issues often lead to flight delays. The inherent underlying network structure is such that these delays can further propagate to cause other delays (for example, a downstream flight delayed due to the delay of its incoming aircraft). The *recovery*

problem focuses on how to quickly return to the original plan, re-accommodating passengers, crews,aircraft, and more. Whereas resource planning problems focus more heavily on profit optimization and have greater flexibility in their computational solution time, recovery problems focus on returning quickly to the original plan, often through the use of very fast heuristics and rules-of-thumb. Recent research in this important and challenging area includes [43], [6], [4], [5] and [64].

- **Airport Operations**

  - **Gate assignment**: The *gate assignment* problem determines which terminal gates are assigned to which inbound/outbound flights. Objectives that have been considered in the literature include minimizing walking distance for connecting passengers or minimizing the total number of missed passenger connections. Related work can be found in [72], [25], [57], [26] and a recent survey paper by [40].

  - **Boarding strategies:** Boarding strategies among airlines vary. In most cases, the objective of a successful boarding strategy is to minimize the overall boarding time on a full aircraft. See [96] for an example of how OR is used to developing and analyzing boarding strategies.

  - **Baggage handling**: Although not as visible as passengers, baggage handling also presents many challenges for airline operations. Ensuring that baggage is transported from origin to destination, often with connections in between, presents an opportunity for OR contributions. See [3] for an example of this research.

  - **Check-in staffing**: Scheduling of ground staff has also been of recent

interest to the OR community. In [90], the authors examines the operational workforce plan at check-in counters.

- **On-demand air transportation**: In recent years, on-demand air transportation has begun to evolve as a new business model for air travel. Constructing and evaluating the networks and operating practices of such companies yields many interesting OR problems. See [45] and [46] for two recent papers in this area.

- **Congestion pricing and slot auctions:** Certain airports exhibit very high levels of congestion, often because of both very high demand for travel into and out of that area and also limited geographical opportunity for airport expansion. The volume of traffic at these airports can lead to significant congestion-based delays, which can in turn propagate throughout the aviation system. Congestion pricing [14] and slot auctions [76] are two examples of external influences on how airlines choose to generate their flight schedules. Both of these have benefited from OR tools for analysis and assessment of the impact of such approaches. For some recent work on this topic, we refer the reader to the thesis of [59].

- **Analysis of delays**: The OR community has also conducted significant empirical and quantitative analysis on passenger airline performance. Examples of this include [7], [89], [11] and [13].

We close this section by noting some valuable textbooks focusing on the airline industry and airline decision making: [2], [39], and [22].

2.3 Resource Scheduling Problems in Passenger Aviation

Within passenger aviation, the three resource planning problems that have received the greatest attention from the OR community (and achieved the greatest successes) are *fleet assignment*, *aircraft routing*, and *crew scheduling*. As such, we focus our primary attention here.

Note that most airlines typically offer between one and four flight schedules per year. For example, they may offer a winter schedule and a summer schedule. Within each schedule, there is usually a consistent pattern that repeats weekly, with many domestic flights repeating daily. Approximately six months to a year before a new schedule begins operations (varying by carrier), the solving of fleet assignment, aircraft routing, and crew scheduling typically begins, with the three problems solved in sequence. The general time-line of solving these airline resource allocation problems can be seen in Figure (2.1). In this figure, the solid lines show initial flow of information, with output from one problem providing input for the next. The dashed lines illustrate a feedback loops, where information from a later problem is used to revise the solution to an earlier problem.

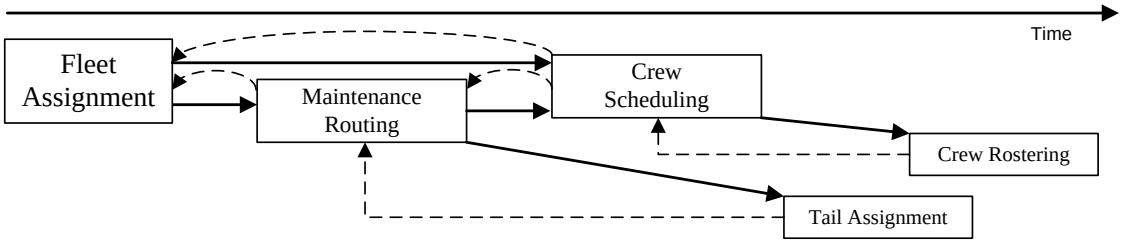


Figure 2.1: Resource allocation solution approach

First, the fleet assignment problem is solved, in which each flight is assigned a specific aircraft type. The goal is to maximize the projected revenue minus the operating cost associated with the assignments, subject to *cover constraints* (every flight must have exactly one fleet type), *balance constraints* (for each fleet type, the flow into an airport must equal the flow out), and *count constraints* (you cannot use more aircraft of a given type than you have in your fleet). Note that this problem results in a partitioning of the flights by sub-fleet. We can then solve a separate (and independent) aircraft routing and crew scheduling problem for each fleet type.

The goal of the aircraft routing problem is to build *lines-of-flight*, i.e. sequences of flights to be flown by individual aircraft (these lines-of-flight will subsequently be assigned to specific aircraft in the tail assignment problem). There are two primary concerns when establishing lines-of-flight. The first is to meet strict *maintenance requirements* as required by the Federal Aviation Administration (FAA) (in the U.S.) or other governing bodies. In order to ensure that it is possible to meet these scheduled maintenance requirements, lines-of-flight are created that start and end at *maintenance stations* (airports that have the capability to perform routine maintenance) without exceeding maintenance limits. The second goal in building lines-of-flight is to establish *flight connections*, i.e. to identify pairs of sequential flights that will share a common aircraft. This has benefits both from the revenue side (charging a premium on desirable itineraries for flight pairs that don't require passengers to change aircraft) and from the crew scheduling side (identifying good opportunities to allow crew to remain with an aircraft over multiple flights, which reduces the propagation of delays).

Once the aircraft routing problem has been solved, the crew scheduling problem

can be addressed. Like aircraft routing, a separate crew scheduling problem is solved for each aircraft type, since pilots are trained to fly a specific aircraft type. Aircraft routing and crew scheduling can largely be solved independently of one another, with the exception of *short connects*. These are pairs of flights with a *tight turn*, i.e. very little time between the arrival of the first flight and the departure of the second. Thus, it is only possible for a crew to be assigned to both of these flights if the flights were assigned to a common aircraft in the aircraft routing solution. In addition, because it is desirable to keep crew with the same aircraft, the aircraft routing problem acts as a key input to the crew scheduling problem.

Similarly to the aircraft routing problem, in which sequences of flights are constructed to be flown by a common aircraft, the main component of crew scheduling, the *crew pairing* problem, builds sequences of flights (*pairings*) to be flown by an individual crew (the specific crews are then matched to the pairings in a *crew rostering* or *bidline* problem). A pairing is a multi-day sequence of flights that are not only sequential in space and time but also comply with all federally mandated rest requirements and duty limitations. The goal of the crew pairing problem is to construct the least-cost set of pairings such that all flights are covered exactly once.

Observe that there is significant interdependence between all of these problems. In particular, the fleet assignment substantially impacts the feasible regions for the aircraft routing and crew scheduling problems by partitioning the flights into independent sets. As such, this raises the question of whether higher-quality solutions could be found by solving the three problems simultaneously. In fact, there is such benefit, but it comes at the cost of substantially increased computational challenges. A sequential approach has been used in the past primarily for reasons of tractability.

In contrast, a sequential approach not only may lead to sub-optimal solutions, but can in fact result in infeasibility. As a result, carriers typically perform multiple sequential iterations with feedback between the three problems.

Once these three problems have been solved to satisfaction, typically months before the schedule's start date, the shorter term problems of tail assignment and crew rostering are conducted — typically on a repeated, rolling horizon throughout the duration of the schedule.

Finally, fleeting, routing, and crew scheduling decisions all continue to be made even at the operational level, when recovery decisions must be made in response to disruptions. In these cases, the problem constraints are largely the same, but the goals are often quite different (for example, instead of focusing on optimizing profits, the goal may be to return to the planned schedule as quickly as possible) and the permissible run time to find solutions is much tighter, leading to a focus on fast-running heuristics and rules-of-thumb over optimization-based approaches.

2.4 Fleet Assignment

2.4.1 Time-Space Networks

In the fleet assignment problem (*FAM*), we want to assign an aircraft fleet type to each flight in the schedule. The goal is to maximize profits subject to the cover, balance, and count constraints described in §2.4.2. Before presenting formulations for this problem, we introduce the notion of *time-space networks* [58]. In a time-space network, each node in the network represents a physical location along with a specific moment in time. The arc connecting two nodes in such a network then represents a transition in both space and time. Such networks can be very powerful in variety of applications, including but not limited to passenger aviation resource planning [63].

In airline planning problems, we often make use of a time-space network in which we have a *time-line* for each station (i.e. airport). A node on this time-line indicates a *flight event* at that station, i.e. either an arrival or departure. Each flight then has two nodes, one on the time-line of its origin airport, at the time of its departure, and one at its destination airport, at the time of its arrival. We refer to the arc connecting these two nodes as a *flight arc*. In addition, we create a *ground arc* from each node on a time-line to the next node in time on that same time-line. These arcs represent aircraft remaining at the station in between the flight events. Figure (2.2) represents such a time-space network for two stations and two flights.

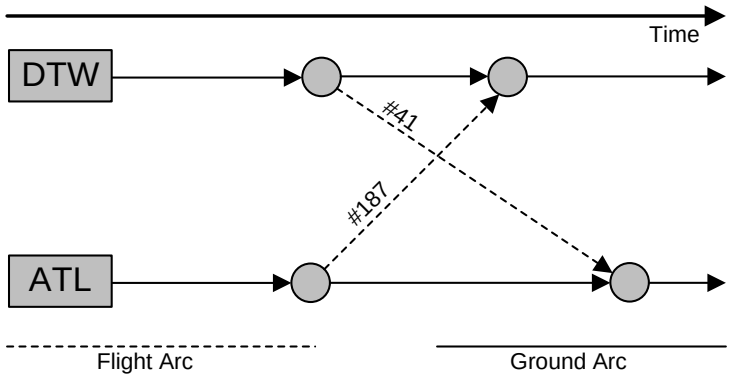


Figure 2.2: Example of a time-space network

2.4.2 Basic Fleet-Assignment (FAM)

Given the concept of a time-space flight network, we now present an integer programming (*IP*) formulation for *FAM*, as originally formulated by [58] and [60].

Sets

F	the set fleet types.
L	the set of flight legs.
T	the set of nodes, i.e. flight events.
S	the set of stations.

Parameters

c_{lf}	the profit gained when leg l is assigned to fleet type f, $\forall l \in L, \forall f \in F$.
N_f	the number of aircraft of type f in the fleet, $\forall f \in F$.
$C \subset L$	the set of flight legs that cross a count-line (e.g. 3:00am)
$I(f,s,t)$	the set of flight legs that are *inbound* to (f,s,t), $\forall f \in F, \forall s \in S, \forall t \in T$.
$O(f,s,t)$	the set of flight legs that are *outbound* from (f,s,t), $\forall f \in F, \forall s \in S, \forall t \in T$.

Decision Variable

x_{lf}	a binary variable that is 1 if aircraft type f is assigned to leg l, $\forall l \in L, \forall f \in F$.
y_{fst+}	the number of aircraft of type f on the ground at station s just *after* flight event t, $\forall f \in F, \forall s \in S, \forall t \in T$.
y_{fst-}	the number of aircraft of type f on the ground at station s just *before* flight event t, $\forall f \in F, \forall s \in S, \forall t \in T$.

Objective:

$$\max \sum_{l \in L} \sum_{f \in F} c_{lf} x_{lf} \tag{2.1}$$

Subject to:

$$\sum_{f \in F} x_{lf} = 1 \qquad \forall l \in L \tag{2.2}$$

$$y_{fst-} + \sum_{l \in I(f,s,t)} x_{lf} - \sum_{l \in O(f,s,t)} x_{lf} - y_{fst+} = 0 \qquad \forall f \in F, \forall s \in S, \forall t \in T \tag{2.3}$$

$$\sum_{l \in C} x_{lf} + \sum_{s \in S} y_{is0-} \leq N_f \qquad \forall f \in F \tag{2.4}$$

$$x_{lf} \in \{0,1\} \qquad \forall l \in L, \forall f \in F \tag{2.5}$$

$$y_{fst} \geq 0 \qquad \forall f \in F, \forall s \in S, \forall t \in T \tag{2.6}$$

Constraint (2.2) enforces the *cover* constraints, i.e. each flight must be covered by exactly one aircraft type. Constraint (2.3) enforces aircraft *balance*: the total

number of aircraft of a given type on the ground at a given station immediately prior to a flight event, plus the number of aircraft of that type which land at that event, must equal the number of aircraft of that type which leave that station at that event plus the number which remain on the ground. These balance constraints, in conjunction with (2.4), enforce *count*. Specifically, we use the concept of a *count-line* which represents a single point in time (typically a time of low activity, such as 3:00am). For each aircraft type, we force the number of aircraft of that type assigned to a flight spanning the count time, plus the number of aircraft of that type on the ground at that station at the count time, to not exceed the number of aircraft of that fleet type available. Because network balance is enforced, if we do not exceed the fleet count at the count time, then we will not exceed it at any time.

2.4.3 Arc Copies

When assigning fleet types to flights, we would ideally like to match each flight to its optimal aircraft, i.e. the one that best trades off between operating cost and capacity (and hence ability to capture revenue). Such a match is not necessarily feasible for all flights, however, because of the balance and count constraints. In practice, it has been observed that small shifts in the timing of the flight schedule can increase the feasible region of FAM, leading to solutions with reduced cost. In the simplistic example in figure (2.3), we consider two different departure times for the flight from station DTW to station ATL. By choosing the earlier of the two departures in set $\{A\}$ out of DTW, we achieve coverage of two flights using a single aircraft of a given fleet type. That is, it is possible to cover both the flight from DTW to ATL and then from ATL to DTW with a single aircraft.

To take advantage of these potential benefits, [77] introduced the *fleet assignment*

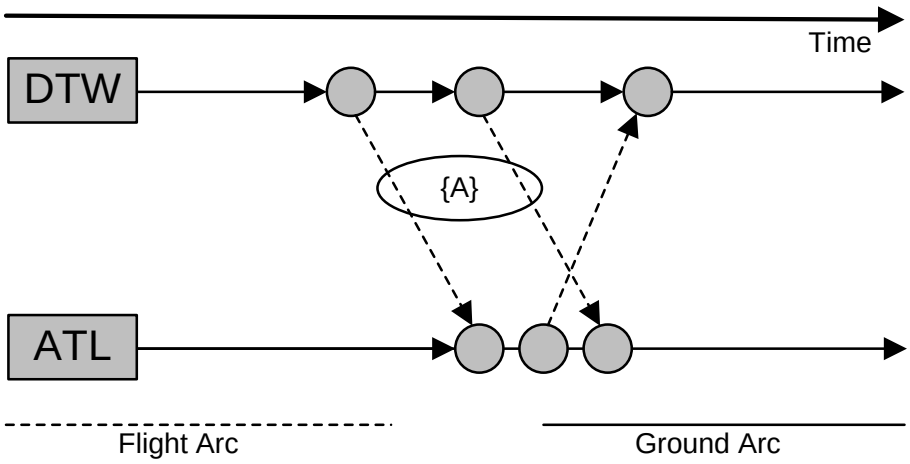


Figure 2.3: Example of the multiple arc formulation

problem with time windows. The idea is to allow small, discrete shifts in time to enable a better fleet assignment. FAM with time windows, as described here, is an example of the integration between fleet assignment and schedule design. Further integration approaches are illustrated in §2.7.1. To formulate this problem, the time-space network is first modified to contain *arc copies*. Specifically, for each flight, we create one arc for each possible time at which time that flight might depart. This is typically limited to a small window (e.g. fifteen to thirty minutes before/after the originally scheduled departure time) so that dramatic changes in potential passenger demand will not be observed. Given this modified network, the basic *FAM* formulation must also be slightly modified. Specifically, the decision variables now represent not only choosing a fleet type for each flight, but also a specific departure time. The objective then becomes equation (2.7).

$$\min \sum_{f \in F} \sum_{l \in L} \sum_{n \in N_{lf}} c_{lf} x_{lfn} \tag{2.7}$$

We replace constraint (2.2) with constraint (2.8), so the cover constraint now selects

not only a fleet type but a departure time as well. We also update the variable definition accordingly.

$$\sum_{f \in F} \sum_{n \in N_{lf}} x_{lfn} = 1 \qquad \forall l \in L \tag{2.8}$$

$$x_{lfn} \in \{0, 1\} \qquad \forall l \in L, \forall f \in F, \forall n \in N_{lf} \tag{2.9}$$

Both the basic FAM and time windows version depend on the input parameters c_{lf}. In practice, estimating these cost parameters can be difficult for many reasons, motivating the need for more advanced FAM models.

2.4.4 Itinerary-Based Fleet Assignment (IFAM)

The objective function of *FAM* depends on the objective coefficients, c_{lf}, which capture both the cost and revenue component of a fleet assignment. The cost component can be fairly straightforward to estimate, but the revenue piece is much more difficult. It is usually thought of not as revenue captured but rather potential revenue that is lost, or *spilled*, due to insufficient capacity. For example, if 200 passengers want to buy tickets for a particular flight and that flight is assigned to a fleet type with only 170 seats, then the revenue from thirty passengers is lost. This spill cost is added to the operating cost to determine the coefficients c_{lf}.

There are several challenges associated with calculating spill. The first is the fact that demand is dynamic. Only one fleeting will be chosen for the entire schedule period, but demand will vary daily over this period. An even bigger challenge is the fact that passengers do not just fly individual flights, but often fly multi-leg itineraries. By only looking at individual legs in *FAM*, we miss the interdependencies

that stem from these itineraries. For example, suppose that a passenger wants to fly from Boston to Los Angeles via Chicago. If the basic *FAM* model is solved, a large aircraft may be assigned to the Boston to Chicago flight with more than enough capacity to meet all demand. If the flight from Chicago to Los Angeles is assigned to an aircraft with inadequate capacity, however, the passenger may be spilled from this flight. In reality, we would lose the revenue of this passenger from both flights; in the model, we would still capture their revenue on the first leg, even though they were spilled from the second.

To address this, [20] developed the following extended version of *FAM*, known as *itinerary-based fleet assignment* (IFAM). In this approach, the fleet assignment decisions are augmented with passenger "spill variables" which take into account the demand for each itinerary rather than each flight leg. Specifically,a fleet assignment also implicitly defines the capacity on each flight leg. Given this capacity, IFAM can simultaneously determine the number of passengers spilled (i.e. potential passengers whose revenue is lost due to inadequate capacity) and corresponding revenue lost across the entire itinerary, not an individual flight leg. Although there are still many challenges with this approach (e.g. it ignores the fact that passenger purchases occur over a rolling time horizon and cannot be fully controlled by the airline), it nonetheless is a substantial step towards overcoming the limitations of a leg-based approach.

We augment the *FAM* formulation by replacing the objective with equation (2.10) and adding constraints (2.11), (2.12) and (2.13).

Parameters

$SEATS_f$	the number of seats available on aircraft of fleet type f, $\forall f \in F$.
$\widetilde{fare}_p$	the fare for itinerary p, $\forall p \in P$.
b_p^r	recapture rate from p to r, i.e. the fraction of passengers spilled from itinerary p that the airline succeeds in redirecting to itinerary r, $\forall p \in P, \forall r \in P$.
CAP_l	is the capacity of the aircraft assigned to leg l, $\forall l \in L$.
δ_l^p	1 if itinerary p includes flight leg l and is 0 otherwise, $\forall l \in L, \forall p \in P$.

Decision Variable

t_r^p	the number of passengers requesting itinerary p that are redirected by the model to itinerary r, $\forall p \in P, \forall r \in P$.

Modified objective:

$$\min \sum_{l \in L} \sum_{f \in F} c_{lf} x_{lf} + \sum_{p \in P} \sum_{r \in P} (\widetilde{fare}_p - b_p^r \widetilde{fare}_r) t_p^r \tag{2.10}$$

Additional constraints:

$$\sum_{f \in F} SEATS_f x_{lf} + \sum_{p \in P} \sum_{r \in P} \sigma_l^p t_p^r - \sum_{r \in P} \sum_{p \in P} \sigma_l^p b_r^p t_r^p \geq Q_l - CAP_l \qquad \forall l \in L \tag{2.11}$$

$$\sum_{r \in P} t_p^r \leq D_p \qquad \forall p \in P \tag{2.12}$$

$$t_p^r \geq 0 \qquad \forall r \in P \tag{2.13}$$

In the augmented model, we not only assign fleet types to flights (thereby determining the capacity on each leg), but also choose the number of passengers to assign to each itinerary through constraint set (2.11). We note here that the parameters ($SEATS_f$) indicates the number of seats available on aircraft type f which is followed by our decision variable, x_{lf}. On the right side of this equation, we represent the demand, Q_l as defined in equation (2.14) and subtract from it the available capacity for the particular leg, CAP_l. Finally, (2.12) ensures that we don't assign more

passengers to an itinerary than there is demand and (2.13) ensures that we do not assign negative numbers of passengers.

$$Q_i = \sum_{p \in P} \delta_i^p D_p \tag{2.14}$$

Most recently, [42] and [41] provide additional research on how to estimate and model itinerary-based passenger demand and its effect on the quality of *FAM* solutions.

2.5 Aircraft Routing

Once flights have been assigned to specific fleet types, the subsequent planning problems can be partitioned into independent sets. For each fleet type, we next solve a *aircraft routing* (*AR*) problem, as described in [53] and [31]. The primary goal of *AR* is to ensure that every aircraft has adequate opportunity to undergo required routine maintenance. For example, in the U.S., an *A* check must be completed every 65 flight hours; any aircraft exceeding this limit will be grounded by the Federal Aviation Administration. Several months before a new schedule begins, carriers therefore build *lines-of-flight* (*LOF*). These sequences dictate a consequence series of flights to be flown by a single aircraft. Aircraft routes can then be constructed by connecting *LOF* that start and end with maintenance events, while ensuring maintenance feasibility over the flights in between.

In constructing *LOFs*, *flight connections* are established as well. When two consecutive flights are flown by a common aircraft, this provides opportunities for improved

passenger itineraries and crew schedules (as there is no need to changes planes between flights), and also has implications for gate scheduling and similar operational activities.

Note that at this stage, specific aircraft (also known as *tails* because they are identified by the unique number painted on the tail of the aircraft) are not assigned to the maintenance routes. Although the intent is to repeatedly fly the same routes over the course of the schedule, these routes will not always be flown by the same aircraft. This is in part to balance the utilization of aircraft over the system, but more importantly it is a reflection of the operational deviations that often occur in practice. For example, as illustrated in Figure (2.4), suppose flights A and B are scheduled to arrive at the same station at roughly the same time, and then their aircraft will be used for departures C and D, respectively. If A is delayed in arrival, an operational decision may be made to use the aircraft from B for C instead of D, as was originally scheduled (for example, to ensure that passengers on C can make international connections that they would otherwise miss). In doing so, the aircraft routes have been swapped, with the aircraft from A now flying B's route and vice versa. In the process, because the different aircraft have different histories (e.g. one may have already flown more hours since its last maintenance than the other), the new routings may be maintenance infeasible.

Although processes vary substantially by carrier, it is not uncommon for the assignment of specific tails to routes to occur on a rolling horizon five to seven days before the day of operation. Each day these assignments are modified both to add a new day to the end of the horizon and also to modify the existing routes to take into account any changes such as aircraft swaps, unplanned maintenance needs, etc.

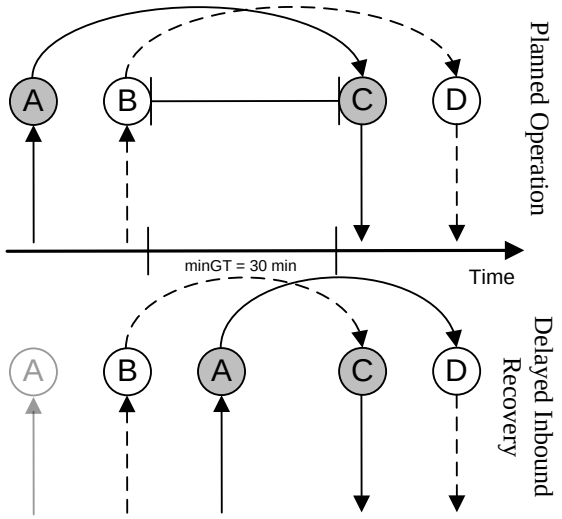


Figure 2.4: Recovery swaps under disruption

For more information, we refer the reader to [51].

2.5.1 Aircraft Routing Models

There are several different ways to model and solve the aircraft routing problem, each with different benefits and challenges, each appropriate for different carriers and different contexts. For example, [62], [53] and [92] present some of the seminal early work in this area, focusing on challenges such as building aircraft routings that spend every third or fourth night in a maintenance station (i.e. an airport with maintenance capabilities) so that short-term maintenance checks can be completed on a regular basis. This work draws largely from graph theory and the development of Euler tours. [31] pose the problem as being to similar to an asymmetric Traveling Salesman Problem (TSP) [65], which they solve using a Lagrangian relaxation [49] and sub-gradient optimization techniques.

We focus here on two particular modeling approaches, one based on *multi - commodity flow* formulation techniques, and the other on *string-based* models.

2.5.2 Multi-Commodity Flow Formulations

More recent formulations of the aircraft routing problem have focused on traditional, linear-programming formulations. As detailed in [56], the maintenance-routing problem can be formulated as a variation of the *multi-commodity flow (MCF)* problem. In the general *MCF* problem [50] and [18], we are given a set of commodities, a set of nodes (each with a supply or demand for each commodity), and a set of arcs. The objective is to find the least-cost way to move commodities across the network from supply to demand while satisfying capacity constraints on the individual arcs.

Sets

F the set of all flights (recall that AR is solved separately for each fleet type)

Parameters

c_{ij} the cost of assigning the connection i to j. This cost often represents the (negative of) the potential additional revenue that can be gained by offering this flight connection as a direct flight with no change of planes. In actuality, carriers are usually more concerned with feasibility than optimality when solving the aircraft routing problem.

Decision Variable

x_{ij} a binary variable that indicates if flight i is followed by flight j and 0 otherwise.

Objective:

$$\min \sum_{i \in F} \sum_{j \in F} c_{ij} x_{ij} \tag{2.15}$$

Subject to:

$$\sum_{j \in F} x_{ij} = 1 \qquad \forall i \in F \tag{2.16}$$

$$\sum_{j \in F} x_{ij} - \sum_{j \in F} x_{ji} = 0 \qquad \forall i \in F \tag{2.17}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in F, \forall j \in F \tag{2.18}$$

To formulate aircraft routing as a variation of *MCF*, we define a network in which each commodity is an aircraft route, each flight is represented by a node, and an arc exists between each pair of nodes corresponding to flights that form a feasible connection. Constraints (2.16) require each flight to be covered exactly once (i.e. to be included in exactly one route). Constraints (2.17) enforce balance. Additional constraints are used to enforce maintenance feasibility as implemented by [56].

2.5.3 String-Based Models

The challenge of the multi-commodity flow formulation lies in capturing all maintenance requirements. Therefore, as alternative, several researchers have taken a string-based modeling approach to solve aircraft routing, often in the context of integrating *AR* with other planning problems [34] and [74].

In the string-based approach, a variable corresponds to the assignment of a particular route to a complete "string", that is, a complete line-of-flights (*LOF*). Each string has an associated cost that spans the entire set of assignments in that string. Constraints primarily focus on building continuous aircraft routes out of strings; the maintenance constraints by definition are enforced through the variable definition, with a variable not included in the model unless the corresponding string is maintenance feasible. The following is an example of a string-based model.

Sets

S	the set of all possible strings.
F	the set of flights.
N	the set of nodes which represent time and space points in the flight network.
$S(s,n)$	the set of strings s that start at node n, $\forall n \in N, \forall s \in S$.
$E(s,n)$	the set of strings s that end at node n, $\forall n \in N, \forall s \in S$.

Parameters

K	a parameter indicating the number of available aircraft.
α_{fs}	a binary parameter that indicates if string s contains flight f, $\forall s \in S, \forall f \in F$

Decision Variable

g_n	variables representing ground arcs which indicate the number of aircraft on the ground at node point n, $\forall n \in N$.
d_s	a binary variable that indicates if string s is chosen in the final solution, $\forall s \in S$.

$$\min \sum_{s \in S} c_s d_s \tag{2.19}$$

Subject to:

$$\sum_{s \in S} \alpha_{fr} d_s = 1 \qquad \forall f \in F \tag{2.20}$$

$$\sum_{s \in E(s,n)} d_s + g_n^- - \sum_{s \in S(s,n)} d_r - g_n^+ = 0 \qquad \forall n \in N \tag{2.21}$$

$$\sum_{s \in S^T} d_s + \sum_{n \in Z^T} g_n^+ \leq K \tag{2.22}$$

$$d_s \in \{0,1\} \qquad \forall s \in S \tag{2.23}$$

$$g_n^+, g_n^- \geq 0 \qquad \forall n \in N \tag{2.24}$$

The objective in equation (2.19) minimizes the cost of the chosen strings. Constraint set (2.20) ensures that each flight is included in exactly one chosen string,

with α_{fs} indicating whether string s covers flight f. Constraint set (2.21) ensures that continuous aircraft routes can be formed from the given strings (note that the mapping from strings to routes may not be unique). Finally, constraint set (2.22) provides a count constraint. As in the *FAM* models, the total number of available strings that are assigned at a given time cannot exceed the number of aircraft available.

The challenge in solving a string-based formulation is the exponentially-large number of variables. One approach to overcoming this challenge is to solve the LP relaxation of the problem via *column generation* [19]. In column generation, a *restricted master problem* is provided in which a limited subset of the variables (here, the strings) are included. This restricted master is solved to optimality. The dual information is then passed to a *sub-problem*, a secondary optimization problem used to identify the string with most the negative reduced cost. If this yields a string with strictly negative reduced cost then this can be passed back to the restricted master which will then continue pivoting. Otherwise, if there are no negative reduced cost strings then the optimality of the problem has been established.

The key challenge in this approach is to find an efficient way to solve the sub-problem. This can be done, for example, by formulating and solving a network flow problem (similar to the MCF approach above) where the arc costs now include the dual information as well as the true costs.

2.6 Crew Scheduling

Just as aircraft are required to follow a complex set of maintenance requirements, there are also many rules that restrict how crews can be assigned to flights. For

example, on a given work day (known as a *duty*), crew members are limited in both the total number of hours that they can fly and also the total *elapsed time* from the start of the duty's first flight to the end of the duty's last flight. There are also limits on the minimum and maximum time between any two consecutive flights. In addition, crew members often are on duty for multiple sequential days. Their multi-day schedule is known as a *pairing*. A pairing is simply a string of consecutive duties, where the first duty starts and the last duty ends at the airport where the crew is *based*, and nights in between the duties are spent at a hotel. A pairing also has many restrictions on the amount of flying and on-duty time permitted, as well as on the amount of rest required between duties. More information on crew restrictions can be found in [16].

In addition to these complex feasibility rules, the cost structure for paying crews is quite complex as well. For example, the cost of a duty is defined by equation (2.25).

$$b_d = \max\{mg_1, f_1 \times elapse, fly\} \tag{2.25}$$

Here mg_1 represents a guaranteed minimum number of hours. For example, if a crew flies a short flight, followed by a substantial wait time on the ground, followed by another short flight, the number of compensated flying hours could be minimal. To prevent such a situation from occurring, each crew is paid at least mg_1 which represents a lower threshold on the number of hours. In addition, f_1 represents a contractual fraction that is multiplied by the total elapsed duty-period, to ensure adequate compensation for a duty period of very limited duration. Finally, fly rep-

resents the total number of flying hours in a duty-period.

$$c_p = \max \left\{ NDP \times mg_2, f_2 \times TAFB, \sum_{d \in P} b_d \right\} \tag{2.26}$$

In the first term, NDP represents the number of duties in a pairing. This is multiplied by mg_2, which is the minimum guarantee per duty. Next, f_2 is again a contractual fraction that is then multiplied by the time-away-from-base, $TAFB$. The final term represents the total of all of the individual duty periods as computed in equation (2.25).

Given the cost functions stated above, the crew scheduling problem thus becomes finding a minimum-cost assignment of crews to flights while satisfying all of these feasibility requirements. For a more detailed discussion of this complex problem, see [16].

Crew scheduling has many parallels to aircraft routing in the sense of assigning resources to sequences of flights while ensuring a number of complex constraints. And like aircraft routing, crew scheduling is solved in two stages. In the first stage, several months in advance of the start of the upcoming schedule, a set of *pairings* is constructed that collectively cover all of the flights. These pairings (analogous to the lines-of-flight in aircraft routing) will be repeated throughout the schedule period, but not always by the same crew members (just as *LOFs* are flown by different tails on different days). It is not until the second stage (typically solved on a monthly basis) that specific crew members are actually assigned to these pairings (analogous

to the tail assignment problem in aircraft routing). This assignment of crews to pairings is typically solved through either a *crew rostering problem* or a *bid-line problem*. For more information on these problems we suggest [29]. We focus for the remainder of this section on the crew pairing problem.

2.6.1 Crew-Pairing Formulation

A crew pairing is a fully self-contained assignment for an individual crew member. That is, given the set of pairings that make up a crew member's monthly schedule, if each pairing is feasible then the full schedule will be feasible. Furthermore, the cost of a schedule is simply the sum of the costs of the pairings.

Thus, we can formulate the crew pairing problem quite simply as a *set partitioning problem* [81] in which each variable represents a feasible pairing. The sole constraint is to choose a set of pairings such that each flight is included in exactly one pairing.

Sets
P the set of all possible strings.
F the set of flights.

Parameters
δ_f^p a binary parameter that indicates whether flight f is included in pairing p, $\forall f \in F, \forall p \in P$.

Decision Variable
x_p a binary variable that indicates if pairing p is included in the solution, $\forall p \in P$.

Objective:

$$\sum_{p \in P} c_p x_p \tag{2.27}$$

Subject to:

$$\sum_{p \in P} \delta_f^p x_p = 1 \qquad \forall f \in F \tag{2.28}$$

$$x_p \in \{0, 1\} \qquad \forall p \in P \tag{2.29}$$

In constraint set (2.28), the parameter δ_f^p is a binary parameter that indicates whether flight f is included in pairing.

Note that although this problem is very concise to formulate, it can be quite difficult to solve for a moderately-sized airline, the number of feasible pairings (and thus binary variables in the model) can easily reach billions. Again similarly to aircraft routing, this problem is often solved using column generation to solve the linear programs [for an alternative solution technique, see [97]].

When solving the LP relaxation of the crew pairing problem via column generation, it is necessary to pose a sub-problem in which we generate the pairing with the most negative reduced cost. This is more challenging than the aircraft routing sub-problem because of the parameters of the complex feasibility rules as well as the non-linear cost function, both of which cannot simply be summed across the flights in a pairing. To overcome these challenges, one of the most successful approaches has been through *multi-label shortest path* algorithms [69]. These approaches are similar to Dijkstra or other label setting algorithms, with the key distinction being that at each node, rather than just keeping one cost label and pruning any path to that node with a higher cost, we must keep multiple labels (e.g. one for cost, one for elapsed time in the duty accrued so far, one for flying time in the duty accrued so far, etc.), and we can only prune when *all* labels are dominated.

Finally, we conclude this section with a discussion of branching strategies. When using column generation to solve string-based models such as aircraft routing or crew scheduling, a sub-problem is used to generate candidate pivot variables, rather than explicitly enumerating all of the variables and computing their respective reduced costs. Note that this approach only solves the LP relaxation of the problem. When column generation is embedded within *branch-and-bound* to solve an integer program, it is often referred to as *branch-and-price* [19]. Using column generation within branch-and-bound introduces its own new set of challenges, as we need to be able to enforce a branching strategy that is consistent with the sub-problem formulation. This is not necessarily a trivial task. For example, in traditional branching strategies, we often pick some variable that has fractional value and impose additional constraints to rule out this value. If x is a binary variable with value 0.5 in the current solution to the LP relaxation, we might enforce $x = 0$ on one half of the tree and $x = 1$ on the other. These new constraints impose an additional dual value in the reduced cost calculation for variable x however, which is not easily captured without modifying the sub-problem structure to treat that variable as a special case.

As an alternative, we can use a strategy known as *branching-on-follow-ons* [10]. We explain this strategy via a simple example. Suppose we have a pairing comprised of four sequential flights, $A \rightarrow B \rightarrow C \rightarrow D$, and that this pairing is assigned to a fractional value. To prevent this fractional solution in subsequent nodes of the tree, we do not branch on the fractional pairing variable, but rather on a fractional flight connection. For example, we can force A to be followed by B in one half of the tree and A to *not* be followed by B in the other half. Forcing A to be followed by B can be imposed by simply combining the nodes representing the two flights into one single node, while forcing A to be followed by B can be imposed by simply deleting

the arc connecting the nodes corresponding to these flights. The structure of the sub-problem remains unchanged, and in fact each progressive sub-problem becomes easier to solve, as more connections are forced a priori.

We conclude by noting that this branching strategy extends to set partitioning problems in the broader sense, where we can branch on items A and B *are* in the same set on one side of the tree and A and B are *not* in the same set on the other.

2.7 Beyond Basic Planning

2.7.1 Integrated Planning

There is clearly a strong link between each of the three planning problems described above (as well as with the schedule design problem, in which the set of flights itself is determined). Significant benefits can therefore be achieved through an integrated rather than sequential approach to solving them. On the other hand, given that each problem is itself challenging to solve individually, solving them simultaneously requires significant advances in modeling and algorithms in order to ensure tractability.

Over the past fifteen years, many advances have been made in this area, with two primary focuses. One is in developing heuristics to quickly find high-quality solutions to large-scale integer programs. The other is in partial integration, trying to identify the most critical relationships between different problems and focusing on capturing these relationships in an integrated approach. The following is merely a sampling of this rich literature.

In [30], the basic fleet assignment problem is extended to include maintenance and crew considerations. That is, although *FAM* is still solved prior to aircraft routing

and crew scheduling, extra constraints are added to increase the likelihood that the *FAM* solution will be maintenance- and crew- feasible.

In [86] fleet assignment is augmented with additional schedule design decisions, and focuses on computational techniques to solve the resulting large-scale integer program. Schedule design and fleet assignment are also integrated in [71] and [17].

Finally, we note that there have been several papers on the integration of aircraft routing and crew scheduling, exploiting the fact that the link between these two problems is fairly narrow – the key connection is that when two flights with a very tight connection time are assigned to a common aircraft, then these two flights become a viable connection for a crew as well. [If the flights were assigned to two different aircraft, there would not be time for the crew to move through the terminal and cover both flights.] Thus decisions made in the aircraft routing problem impact the feasible region of decisions to be made in the crew scheduling problem. This problem structure naturally lends itself to a variety of decomposition approaches. These are explored by [35], [34], [74], [75] and [100].

2.7.2 Robustness & Recovery

It is important to note that airline planning problems are often modeled as static and deterministic, although the real-world problems are both dynamic and stochastic. For example, the same fleet assignment is flown repeatedly over the course of a schedule period, even though demand varies daily over this horizon. Furthermore, the flight times needed to define the time-space network are taken as fixed, whereas actual flight times can vary quite substantially in practice.

The reasons for static and deterministic modeling are two-fold. The first is that repeating schedules have operational benefits, reducing the number of decisions that

must be made and communicated, and allowing workers to develop familiarity with a plan over time. The second is that even solved statically and deterministically, these problems are computationally very challenging.

The cost of these simplifying assumptions is that it is quite common that a carrier will be unable to fully operate a schedule as planned on any given day. Maintenance problems, weather delays, and many other sources of disruption will require modifications to the original plan to recover from these disruptions. To reduce their impact, there are two ways to reduce the impact of disruptions.

One is through sophisticated recovery tools that allow the user to quickly modify the current schedule in response to a disruption. There is a vast literature on this topic. The aircraft recovery problem is formalized in [78] and is further studied in [43]. For more information on disruption management, [32] provides a survey paper that covers aircraft recovery, crew recovery as well as integrated crew aircraft and passenger recovery. In [82], the authors use column generation approach to deal with day-of-operations disruption and subsequent recovery.

The second approach is to incorporate *robustness* into the planning process. This can take the form of reducing the impact of delays (for example, adding buffer between flights decreases the likelihood that one flight delay will propagate to a subsequent down-stream flight [7]. It can also take the form of creating greater opportunities for recovery when disruptions do occur (for example, building crew pairings that provide extra swap opportunities when an inbound crew is delayed, enabling another crew to take over the delayed crew's outbound flight [85] and [52]

We conclude by highlighting a few recent approaches to improving robustness in airline planning:

- In [79], the authors note that when airlines cancel flight, they tend to cancel entire cycles, i.e. sequences of flights that begins and ends at the same airport. As a result, a fleet assignment and aircraft rotation with many short cycles is frequently less sensitive to a flight cancellation than one with only a few long cycles.

- In [7], the authors quantify the prevalence of delay propagation in modern airline schedules. This provides motivation for [8], in which minor modifications are made to flight departure times to redistribute the network's existing slack, moving extra slack to turns that are historically prone to delay propagation and away from turns that are historically reliable. This work builds on [66] and most recently in [38].

- The motivation behind [67] is the fact that tail assignments are frequently swapped over the course of the day to adjust for disruptions. This can make longer-term maintenance plans infeasible. This paper takes an existing set of LOFs and modifies them so as to maintain important crew and passenger connections while maximizing the number of opportunities for over-night recovery of the maintenance plan.

- In [83], simulation (in the form of a tool called SimAir) is used to evaluate the quality of crew schedules when implemented under stochastic conditions.

- In [44], [102] and [93] the authors explore the construction of robust crew schedules under uncertainty.

- The authors use the idea of a *station purity* measure to improve the robustness of fleet assignments in [87].

- The authors of [100] provide an iterative approach to generating integrated aircraft routing and crew scheduling. By first solving one problem to optimality, the authors are able to obtain trade-off points between cost and robustness.

We believe that this chapter provided a general overview of the aviation industry and its close connection to the operations research community. In the remainder of this dissertation, several of the mathematical formulations, such as the network-based flow model, the connection-based model and the string-model will be revisited. In Chapters III, IV, V and VI we apply these formulations to airline robustness planning and maintenance routing problems.

CHAPTER III

A Recursion-based Approach to Simulating Airline Schedule Robustness

3.1 Introduction

Flight delays are an increasingly common occurrence in today's air travel industry. The Air Transport Association estimates that there were a total of 134 million system delay minutes during 2007, which amounted to a $8.1 billion increase in direct operating costs to U.S. airline carriers. (Air Transport Association, 2008). In addition, the U.S. Department of Transportation reports only 78.5% of flights as on time from 2003 until 2005, a number that decreased to 75.4% during 2006 and shrank even further to 73.4% during 2007. (U.S. Department of Transportation, 2008) [95]. Many of these delays are what we refer to as root delays — delays stemming from events which are intrinsic to the particular flight (e.g. a mechanical failure or weather delay). These delays are predominantly independent of the flight schedule. However, there is also a significant impact on the system stemming from propagated delays. These are delays passed on from one flight to its subsequent connections, which await its crew, aircraft, or connecting passengers. Clearly, these delays can be impacted by the schedule, as the schedule determines the flight connections, as well as the slack between flights that can be used to absorb disruption.

In this paper, we present a recursive algorithm that simulates a daily airline

schedule by generating root delays and measuring their propagation to downstream flights. Specifically, we are interested in the metric of total propagated delay minutes, which is the cumulative number of minutes that flights will be delayed due to the propagation of root delays. This metric is of great interest to an airline, as it provides them with a way to compare the expected operational performance of two different flight schedules.

3.2 Related Literature

Simulation is frequently used in airline applications as a way to overcome network complexity and the impacts of stochasticity, both of which can make closed-form solutions intractable.

In [15], the authors presents an algorithm for simulating a flight reservation system. Specifically, the presented algorithm allows an airline to analyze how changes in flight schedule will alter demand, and eventually affect overall revenue.

The authors in [23] present a simulation to determine the effects of yield management, the controlled allocation of seats (between business and leisure travelers) and their respective prices on high-demand flights.

Along the lines of flight schedule simulation, in [80] the authors introduce SIMAIR, a modular airline simulation built on three sub-components: a controller module, a recovery module and an event generator module. This airline simulation software allows for the evaluation of schedules and possible recovery scenarios with a variety of performance measures, the main focus of their investigation.

Finally, in [70], the authors present further work in flight schedule simulation by extending SIMAIR. The authors develop a module that allows for the importation of different recovery scenarios based on what a specific airline carrier is capable of

performing. Our approach extends this literature by using simulation to assess the robustness of a fixed schedule. This is an important step towards improving the link between planning and operations.

3.3 Key Challenges

This section provides an introduction to flight schedule simulation, a technique used to evaluate a given flight schedule under various performance metrics. For example, network planners may be interested in analyzing a schedule's performance under heavy delay, present during severe weather conditions.

Specifically, this section uncovers the challenges associated with such flight schedule simulations, and exposes why a simple first-in, first-out simulation (FIFO) fails to capture an accurate measure of the total amount of propagated delay that occurs in a flight network.

3.3.1 Flight Network Properties

Flight schedules feature intricate relationships among aircraft, crew, and possibly other resources. These relationships can be expressed as out-trees, as seen in Figure (3.1). In this example, we notice that the aircraft from Flight 515 proceeds to Flight 234, while the cockpit crew connects to Flight 562. Both of these flights subsequently influence additional flights as well. A successful simulation of such a flight schedule must take into account any effects that a flight can have on all flights in its corresponding out-tree.

In addition, flight relationships can also be represented as in-trees, where resources from different flights merge. This occurrence is depicted in Figure (3.2). In this example, we notice that the aircraft from Flight 623 combines with the cockpit crew from Flight 1262 to conduct Flight 763. In this case, when a simulation is performed,

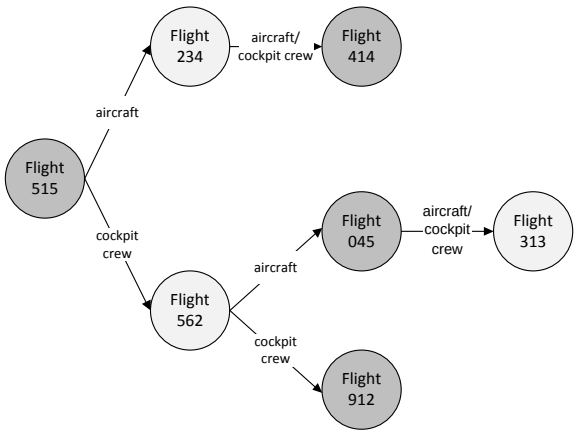


Figure 3.1: Outbound resource effects

all possible effects from incoming flights must be taken into consideration.

Because flight schedules contain both in- and out- trees, the network cannot be defined as a pure tree structure.

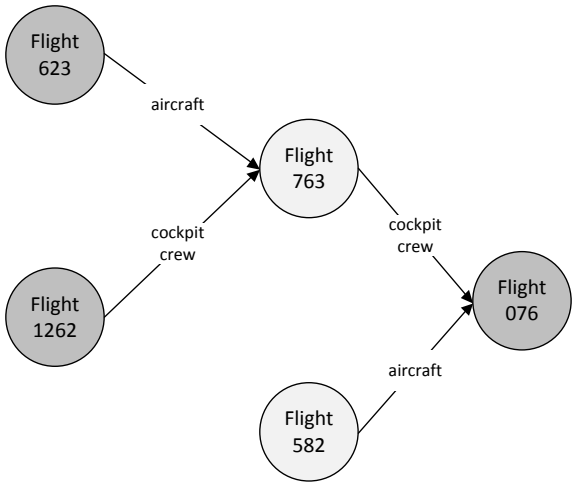


Figure 3.2: Inbound resource effects

3.3.2 Flight Schedule Time Issues

In addition to the intricate propagation relationships presented in the previous section, flight schedules also exhibit complicating timing properties that do not allow for first-in, first-out simulation approaches. Specifically, flight schedules repeat daily, making the schedule circular rather than linear. Thus, there is no "first flight" and

any flight in the day's schedule can be delayed by upstream flights, and also cause delay to downstream flights.

Consider the example in Figure (3.3). A first-in, first-out algorithm would simulate this schedule in the order of departure time. It would start by simulating Flight 466 and propagating its effects to its connection, Flight 098. Next, Flight 098 would be evaluated and again, the respective propagation effects measured. Finally, after simulating Flight 231 (at the end of the day), the algorithm would terminate. However, in some cases this termination would be premature, because Flight 231 has the potential to propagate delay back to Flight 466.

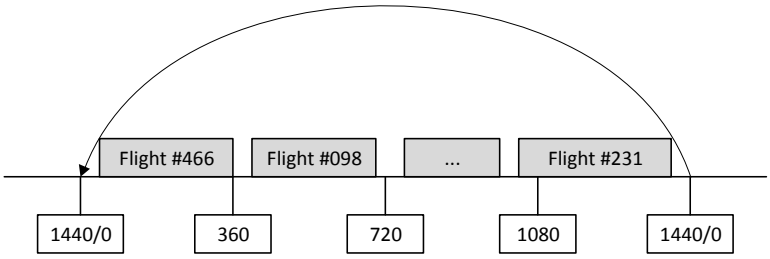


Figure 3.3: Recurring schedule example

As seen from this example, using an iterative, first-in, first-out simulation methodology when comparing two or more flight schedules is ineffective, since the concept of a particular "first flight" in a flight schedule does not exist.

Furthermore, Figure (3.3) depicts only a simple example, with two resources remaining paired throughout the day. As presented in Figure (3.2), a flight may be preceded by two distinct flights, leading to further difficulty during the propagation time calculation.

A successful simulation must take into consideration the fact that there is no predetermined start and end for a given simulation, such that when a flight is simulated, it can potentially suffer from incoming delays, or propagate delays to flights occur-

ring on another day. In addition, the simulation routine must be able to accurately accrue the amount of total propagated delay from distinct incoming flights.

3.4 Recursion-Based Approach

Prior to presenting our simulation algorithm, we describe several assumptions made during the creation of this model.

A1 We only consider root delays of 60 minutes or less in duration. Although longer root delays certainly occur (e.g. when a major storm impacts an airport, or an air-craft becomes unusable to a malfunction), such delays are typically not appropriate to plan for in the scheduling process.

A2 - Our probability distributions, which were derived from actual carrier data (see Section 4.1), are premised on the assumption that root delays are independent and identically distributed. Although root delays certainly do show some correlation effects (e.g. flights originating at a common airport during a common time period will experience the same inclement weather), we did not have adequate data to include this in our probability distributions. Our approach however, allows for the substitution of other probability distributions, including those exhibiting correlations.

A3 - We assume that all delays continue to propagate until they are fully absorbed by slack in the network. In fact, this is not always the case, and recovery options such as canceling flights or swapping aircraft can also be used to absorb delay. Such recovery decisions are difficult to replicate, however, as they are typically made by individual operations controllers and thus vary from person to person. Furthermore, many of these recovery options are not appropriate for delays of lesser magnitude and instead are used for higher-impact delays.

A4 Our recursive algorithm requires that the total amount of propagated delay

does not exceed 24 hours. This is a reasonable assumption given that root delays do not exceed 60 minutes, and that there are typically down-times throughout the schedule (especially overnight) that can absorb residual delays. This assumption held true in all data sets that we were provided.

A5 When a flight incurs separate propagated delays, both from its incoming aircraft and incoming crew, we assume the propagated delay is the maximum of the two rather than their sum. In other words, as soon as both resources are available, the flight can depart. However, we then add any root delay in addition to this root delay. For example, a mechanical problem would not be observed until the aircraft was available.

3.4.1 Generating Delay Data

In developing the delay generation unit of our model that is used to simulate randomness in our flight schedules, we follow a similar approach to Rosenberger et. al. (2000), who use an Event Generator Module that implements a probability distribution to generate random events/delays in a network.

As previously mentioned, our probability distributions are based on actual U.S. carrier data. Initially, this data set included different types of delays, such as mechanical failure, weather, delays due to late incoming air-craft, etc. We filtered this data so as to omit all types of delay that indicate propagation, leaving us with only root delay data.

Using the available, filtered data, we attempted to fit various distributions for each of the origin airports that appeared in the flight schedule. However, since deviations from this set of possible distributions tended to exceed statistical thresholds, we reverted to empirical distributions instead. The delays are separated into bins of 0, 10, 20, 30, 40, 50, and 60 minutes of delay, each bin with a respective probability of

occurring.

In developing these empirical distributions, we assume that the origin airport is responsible for a root delay of a given flight. Arguably, there are situations in which the destination airport is the cause of a late departure or possibly a mid-flight weather problem causes a late arrival. Nonetheless, given the fact that most of the delays of inclement weather/mechanism failure occur prior to take-off, we feel that this assumption is valid, but could benefit from further research. Again, as mentioned previously, our simulation algorithm does not depend on the probability distributions.

3.4.2 Algorithm Design

Before presenting the actual algorithm, we define several terms commonly used in the airline industry. Minimum required turn time is the amount of time deemed necessary to "turn around" an aircraft or crew before their next flight. This may include the time required to clean the aircraft, or for the pilot to perform inspection. Block time refers to the difference in time from the point at which an aircraft has departed from its origin gate until its arrival at its destination gate.

In order to overcome the problems associated with the first-in, first-out simulation model, we implement a recursive algorithm. This algorithm explores all possible flight connections until the total amount of propagated delay has dissipated. We encourage the reader to follow along with the diagram presented in Figure (3.4) that visually explains the simulation algorithm.

We begin by adding all flights in the schedule to our future event list, in arbitrary order, initializing each flights propagated delay and root delay to zero. We then do the following:

We select the next flight in the future event list to process and call upon the

random delay generator to provide a root delay for this particular flight. If this root delay is non-zero, we update the departure time of the current flight to be the sum of the (current value of the) propagated delay and this additional root delay. We then consider all outbound connections from this particular flight. For each connection, we determine how much (if any) the current flight's delay would propagate to this child by subtracting the existing slack from the current delay value. If this residual delay is non-negative, we then check whether it is larger than the connecting flight's cur-rent propagated delay.

If it is, we replace the connecting flight's propagated delay with this new, larger value. Next, we add this to the connecting flight's root delay (which would be 0 if that flight had not been processed yet) to determine its new departure time. Finally, we recursively examine the current flights children to evaluate the impact of the residual delay on these flights, repeating until the flight delay is fully absorbed.Once the algorithm has explored all flights in the out-tree of the root flight, it moves to the next flight in the future event list and the process repeats.

It should be noted that some flights are updated more than once in our algorithm. This is essential in accurately determining the amount of delay that could be propagating from two different parent flights. In addition, each flight in the schedule will be exposed to a possible root delay from the our random delay generator. This delay will be added to any propagated delay that has already been simulated and downstream flights will be updated as well.

3.4.3 Simulation Example

This algorithm, due to its recursive nature, overcomes the challenges presented in Section 3, and accurately models the propagated delay in the flight network, even when multiple parent flights propagate flights to the same child node. Consider the

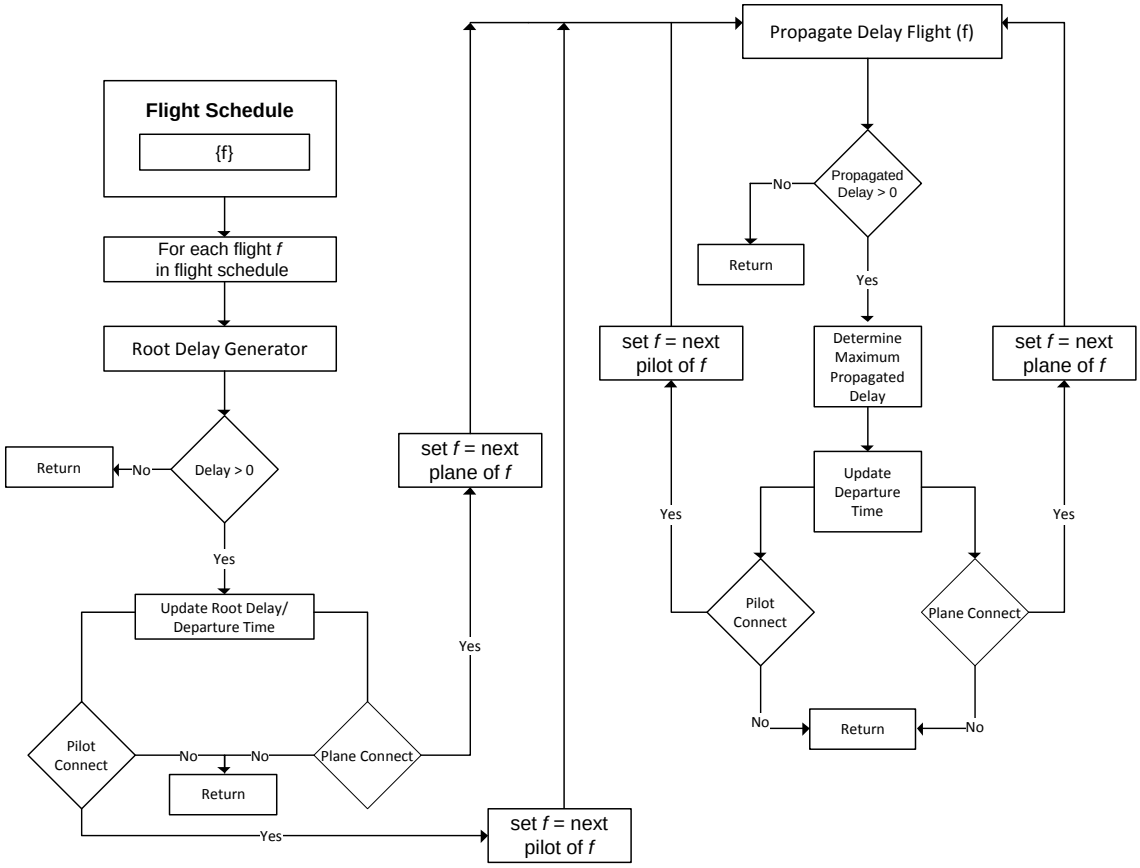


Figure 3.4: Simulation order of execution

following example in Table (3.1).

Flight No.	Root Delay	Depart. Time	Block Time	Pilot Connect	Aircraft Connect
752	30	055	135	823	641
214	50	124	062	Off	823
⋮	⋮	⋮	⋮	⋮	⋮
823	20	240	090	024	024

Table 3.1: Multiple parent flights result in additional delay

Our algorithm starts by simulating Flight 752, which incurs a root delay of 30 minutes. This means that the connecting flights (Flight 823 and 641) would have the earliest possibility of departing at time: 255 (55 (Departure) + 30 (Delay) + 135 (Block) + 35 (Turn Time) = 255). We can see that Flight 823 was supposed to leave at time 240, so at this point, we have incurred a propagated delay of 15 minutes

(255-240).

Suppose, we now simulate Flight 214, which also incurs a root delay, however, it is a 50 minute delay. This results in Flight 823's earliest departure time as 271. (124 (Departure) + 50 (Delay) + 62 (Block) + 35 (Turn Time) = 271). We know Flight 823 has already been delayed until 255 from Flight 752; however, due to Flight 214, the departure time of Flight 823 has to be pushed back further, to 271 in this case.

This example shows that when multiple parent flights interact, the departure time of a subsequent flight may be impacted by (n)either/both parents.

3.4.4 Model Implementation

The simulation algorithm is implemented according to the algorithm and diagram provided in Figure 4. We use the C++ programming language for its ability to easily store and create linked lists (flight schedules) and speed to implement the actual simulation.

If we consider flight schedules with their connecting flights to be represented as out-trees, we determine that the depth of these trees rarely exceed four levels in our given data set. With this information, we are able to run a large number of replications in a matter of seconds; more information about the actual computational results is presented in our Results Section.

3.4.5 Model Verification

As part of the simulation process, we verified our model to ensure that it is indeed measuring our objective function of total propagated network delay. Our verification process consisted of both white-box and black-box testing, which will be discussed in this section.

As a first attempt at the verification process, our simulation code was tested at

the functional level. Using a generated set of input values for which the output had been pre-computed, the results from function calls were verified.

To complete functional verification, we created several complex flight schedule scenarios to ensure that our model functions correctly under all situations.

3.4.6 Model Validation

To validate the results of our simulation, we followed advice presented in Sargent (2007). As suggested, we use a technique referred to as "comparing to other models." As part of another project, we had access to expected values of propagated delay given a pre-computed delay occurrence. These data values were in the form of: Given the fact that Flight 1 suffered a 60 minute delay, there exists a total of 35 minutes of propagated delay in the network.

While this model only considered one delay at a time, it allowed us to compare the expected values to our simulation. Matching up these values allowed us to further conclude that our simulation was indeed functioning correctly given our set of assumptions.

3.5 Computational Experiments

AhmadBeygi et al. (2010) [8] presents a linear program to reallocate extra minutes (slack time) that exist between flights. Theoretically this algorithm creates a flight schedule that intrinsically is able to deal with delays by reallocating slack time to those flight connections that are most likely to experience delay. Using our simulation algorithm, we are able to verify the results of the presented model and conclude it is indeed a better schedule due to a reduced amount of propagated delay.

The linear program provides for solutions at different layers of delay propagation. The Single-Layer Model (SLM) restricts itself to only consider the propagation of

delay from a parent flight to immediate possible connections, while the Multi-Layer Model (MLM) considers all layers of propagation. Using origin-based probabilities, the program computes a new schedule which should be less sensitive to delays occurring in the flight network.

Both the Single-Layer and Multi-Layer Models presented in AhmadBeygi et al. (2010) [8] use a surrogate objective function to approximate the amount of delay propagated by the flights in the network. In the case where two different inbound flights both affect a common connecting flight, this model provides an over-estimate of the amount of propagated delay that is incurred by this new schedule. Conversely, in the case where two connecting flights both incur a root delay, this model provides an under-estimate of their impact on subsequent downstream flights. Our simulation model provides us with a means to assess the significance of these errors.

Using an implementation of this linear program, we were able to generate several schedules based on two original input schedules (data set 1 and data set 2). For each of the respective data sets we generated four new schedules representing various levels of flexibility. We use our simulation algorithm to determine the robustness of such a generated schedule versus the original flight schedule. The results from this simulation are presented in Table (3.2), table which shows 95% confidence intervals for the amount of reduction experienced in the Single-Layer and Multi-Layer Scheduling model when compared to the original schedule. Each confidence interval was generated from 1000 replications for each data set/duty restriction/layer type combination.

As evident from the above-seen values, we are able to conclude that schedules produced by Single-Layer and Multi-Layer Scheduling models are more robust than their respective original schedules in terms of their ability to absorb randomly occurring

Data Set	Duty Rest.	Single-Layer Model Schedule	Multi-Layer Model Schedule
1	0	(3.2%, 5.4%)	(4.4%, 6.6%)
1	5	(21.8%, 23.9%)	(23.4%, 25.5%)
1	10	(35.6%, 37.5%)	(37.5%, 39.2%)
1	15	(46.0%, 47.6%)	(47.8%, 49.3%)
2	0	(2.6%, 5.0%)	(3.5%, 5.9%)
2	5	(22.7%, 24.8%)	(24.3%, 26.3%)
2	10	(37.0%, 38.8%)	(38.9%, 40.6%)
2	15	(47.9%, 49.5%)	(50.1%, 51.7%)

Table 3.2: 95% confidence intervals for delay reduction

network delay.

Even though our simulation algorithm is recursive in nature, it still performs well for a large number of replications. Simulating a schedule with more than 500 flights for 1000 replications takes less than 5 seconds to run on a 2.2GHz Intel machine. It is our conjecture that our approach will remain tractable even for larger flight net-works with deeper propagation trees.

In addition, in an effort to reduce the variance and the overall number of replications required, we use common random numbers when simulating both the original and enhanced flight schedules. This practice allows us to construct a paired-t test for which we check whether the confidence interval of the difference between two schedules includes zero. If it is not included, we can conclude that with some alpha-confidence that one schedule is indeed more robust than another

3.5.1 Conclusion

The presented algorithm successfully simulates a flight schedule, incorporating the relationships between inbound and outbound flight connections, as well as the cyclic nature of the daily repeating schedule. Our recursive approach addresses these challenges while maintaining tractability. This enables us to provide an accurate estimate of the total propagated delay minutes, a measure that can be used to draw

conclusions about the robustness of a given flight schedule and, in particular, its ability to deal with common, everyday delays.

3.5.2 Future Work

As presented, our algorithm works well in estimating the total minutes of propagated delay given a particular flight schedule; however it does not take into consideration the use of recovery operations. While it is often difficult to provide accurate costs for each of the possible recovery operation and the fact that such operation decisions are usually made ad-hoc as Rosenberger et. al (2000) notes, it may be worth exploring the use of scenarios. Such an ex-tension would then serve to explore the comparison be-tween two schedules, subjecting them to the same delay with an identical set of possible recovery operations. This would provide a more accurate depiction of actual flight schedules, since recovery operations are frequently implemented in practice.

In addition, it may be worth incorporating passenger connections as a measure of robustness. While it may be the case that one schedule may be significantly better than another when it comes to minimizing the total amount of propagated delay, this may not minimize the number of passengers missing a connection.

CHAPTER IV

Modifying Lines-of-Flight in the Planning Process for Improved Maintenance Robustness

4.1 Introduction

As part of their planning process for a new schedule, domestic U.S. airlines typically construct *lines-of-flight* (*LOFs*), day-long sequences of flights that will each be flown by a single aircraft. These LOFs are then repeated on a daily basis throughout the duration of the schedule. Such repetition provides operational stability, defines opportunities to keep crews and aircraft together, and generates a consistent set of passenger itineraries that do not require changing aircraft.

Once the planning process has been completed and operation of the schedule begins, specific aircraft (*tails*) must be assigned to the LOFs. The *tail assignment* problem is often solved on a rolling n-day horizon (e.g. one week), with specific tails being assigned to a sequence of n consecutive and connecting LOFs. Such *aircraft rotations* not only ensure coverage for each of the LOFs, but also provide scheduled times at which the specific tails will remain overnight at a maintenance station where routine maintenance checks can take place. [For the carrier with which we worked, aircraft require routine maintenance checks approximately every seven days. For the sake of exposition, we will focus on this case. We use the term *day-seven aircraft* to

refer to an aircraft that is beginning its seventh day and thus must terminate the day at a maintenance station.] These rotations are updated on a daily basis, in part to add a new nth day to the end of the rotation, but also to correct for any over-the-day operational swaps that occurred throughout the day.

During day-of-operations, it is not uncommon for an operations controller to perform a tail swap in order to mitigate a variety of unexpected disruptions. Such swaps can be beneficial in the immediate future by providing coverage for a disrupted flight. Downstream problems can arise, however. For example, although the operations controller will typically not swap a day-seven aircraft with another tail whose current LOF does not end the day at a maintenance station, it is not uncommon for a day-six aircraft to be swapped onto a rotation that does not terminate at a maintenance station the following day. Thus, when the rotations are repaired at the end of the day, this aircraft must be assigned to a revised rotation that terminates that following day at a maintenance station.

We define a *maintenance line-of-flight* (*MLOF*) as a LOF that terminates at a maintenance station, as seen in Figure (4.1). Here, a LOF directs an aircraft to fly DTW→BOS→FLL→DFW over the course of a day. If the number of day-six aircraft terminating the day at a given station is larger than the number of MLOFs that originate at that station, then additional costly over-the-day changes to the planned LOFs will be required the following day (when these day-six aircraft become day-seven aircraft) to ensure that all of these tails can get to a maintenance station. We refer to the difference between the number of day-seven aircraft starting the day at a station and the number of MLOFs originating at that station as its number of *maintenance misalignments*.

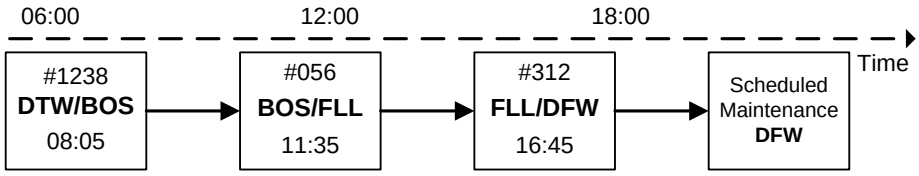


Figure 4.1: Example of lines-of-flight

The more MLOFs a station has, the less likely it is to experience maintenance misalignments, and incur the associated costs and complexities of modifying the day's LOFs to recover and re-establish maintenance feasibility. Moreover, the more outgoing LOFs a station has, the more benefit it receives from an additional MLOF (in terms of being able to easily recover from over the day swaps), because the expected number of day-seven tails at the start of each day is larger. The total number of possible MLOFs that can be built in the planning process is limited, however, by the number of end-of-day flights in the flight schedule that terminate at a maintenance station.

The focus of our research is on improving the robustness of a planned schedule by redistributing these *maintenance opportunities* across all stations, while only making limited changes to the originally-planned LOFs. Our goal is to minimize the overall expected number of maintenance misalignments. Operationally, this provides a higher likelihood of being able to re-assign disrupted tails to new rotations that ensure maintenance feasibility without further disruption to the planned LOFs.

The contributions of our work are two-fold. First, we introduce a new robustness metric called *maintenance reachability* (MR), with which we are able to analyze a set of LOFs and quantify its ability to withstand unplanned changes from a maintenance feasibility standpoint. Second, we develop an optimization model to construct improved LOFs so as to maximize maintenance reachability by minimizing the total

number of maintenance misalignments.

In the next section, we review the relevant literature. In §4.3, we provide an introduction to the airline planning process, define maintenance reachability, and discuss how the maintenance reachability of a given schedule can be improved with only minor changes to the original set of LOFs. In §4.4, we present a model for optimizing the maintenance reachability of a planned set of LOFs. In §4.5, we provide computational results based on U.S. carrier data. Finally, in §4.6 we offer concluding thoughts and motivate future work to be done in the area of maintenance robustness.

4.2 Literature Review

Airline operations have benefited greatly from the advent of applied optimization techniques. Such techniques have been used to address various processes of the airline operations portfolio, including *schedule planning* [53], *fleet assignment* [58], *crew scheduling* [16] and *aircraft routing* [31].

These planning problems are typically solved once per schedule, often several months in advance of the execution of a particular schedule, and set the framework for daily operations. It is during the aircraft routing phase that the lines-of-flight are created. [37] use a column-generation approach to generate feasible aircraft routings with maintenance events. [54] introduce the concept of a LOF within an optimization framework and provide a polynomial-time algorithm to solve a three-day routing problem for a fixed set of lines-of-flight.

Once the execution of the schedule begins, shorter-term decisions must be made on an ongoing basis leading up to day-of-operations. Such decisions include the

assigning of specific aircraft to multi-day rotations consisting of consecutive and connected LOFs. This process, known as the *tail-assignment problem*, is detailed by [55]. The tail assignment process achieves two primary goals: to cover each LOF with an aircraft and to build rotations that include planned maintenance events for each tail.

During the actual day-of-operation, disruptions may occur, such as mechanical problems or weather delays. To recover from these disruptions, carriers often swap aircraft between two flights. Thus, an aircraft originally on a rotation that included a planned maintenance event may be swapped to a rotation that does not enable required maintenance checks. As a result, additional modifications must be made to ensure overall feasibility of the aircraft with respect to maintenance. Recovery of the tail assignment under uncertainty is explored by [94], in which the authors provide a model that alerts the dispatcher to inconsistencies in a particular maintenance routing plan. It is then the responsibility of the dispatcher to find a line-splicing opportunity and route aircraft accordingly to receive maintenance. [48] provide a survey of various models that provide recovery methods under uncertainty.

An alternative approach to dealing with uncertainty is through building a more robust schedule to begin, where robustness includes both reducing the likelihood of a disruption and increasing the number of opportunities to recover from a disruption easily. There has been significant research in building more robust airline plans as a way to reduce the potential impact of real-time disruptions. For example, [66] explores a gain in schedule robustness through re-timing of flights. This idea is further extended by [38] and [8] by re-timing individual flights based on the underlying delay distribution of individual stations. [27] use a column-generation approach to generate

tail assignments, while minimizing the expected delay. Recently [43] solve the airline recovery problem with considerations for maintenance events using the concept of a recovery network.

The focus of our work falls into this category -- improving the robustness of an airline plan to reduce the impact of daily disruptions. Our approach consists of taking the initial set of planned lines-of-flight and making limited changes to this plan to enhance maintenance robustness.

4.3 Maintenance Reachability

4.3.1 Airline Planning Process

Before presenting our approach to developing planned lines-of-flight that are more robust in responding to operational disruption, we first describe the planning process at the major U.S. carrier with whom we conducted this research.

To begin the planning process, the airline sets its schedule of daily-repeating flights for a fixed time period, for example, a quarterly schedule. Given the set of flights, the fleet assignment problem is then solved, assigning each flight a specific aircraft type. Once fleeting is completed, the schedule can be decomposed by fleet type and, for each fleet type, the crew scheduling problem and maintenance routing problems are solved.

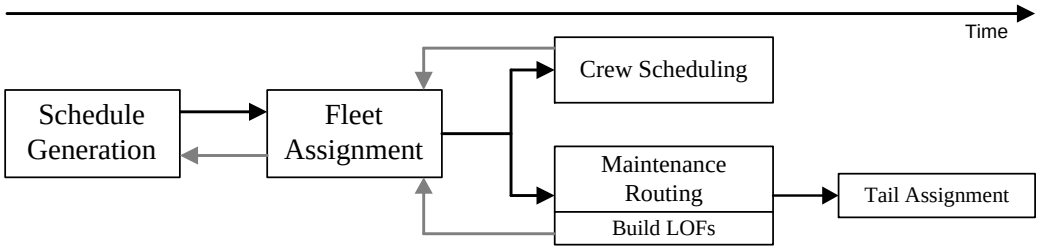


Figure 4.2: The typical airline planning process

Within maintenance routing, the lines-of-flight are constructed. These lines-of-flight are then used to construct feasible aircraft rotations. The rotations are not assigned to specific aircraft (and, in fact, may be modified on a daily basis once the operation of the schedule begins), but they provide a mechanism for ensuring that a maintenance-feasible set of rotations exists before the schedule is set. This process is illustrated in Figure (4.2).

On any given morning, each station in the flight network will have a number of outgoing LOFs, some of which will terminate in a maintenance station. If a station has at least as many MLOFs as it does *day-7 aircraft* (aircraft starting the day at the station which require maintenance at the end of the day), then the schedule can be executed as planned. However, should the station hold more day-7 aircraft than MLOFs, then it may be necessary to perform costly, over-the-day swaps with other lines-of-flight to ensure that all aircraft reach their required maintenance events. From a *planning* perspective, it is of interest to airlines to ensure that whenever an aircraft begins its seventh day of operation, there is a high likelihood that there is an available MLOF to which it can be assigned, ensuring that it ends the day at a maintenance station without the need for costly over-the-day swaps.

4.3.2 Disruption Management

Although the *planned* lines-of-flight and corresponding aircraft rotations are maintenance feasible, these plans are often disrupted in practice during day of operations. In such cases tail assignments, rotations, and even the LOFs themselves may need to be modified to regain maintenance feasibility. These disruptions tend to manifest themselves in the form of unexpected equipment changes and other measures to counter delay propagation. For example, an operation controller may choose to alter

the tail assignment to mitigate a delay propagation effect by performing an aircraft substitution. Such a scenario is presented as a numerical example next.

Table (4.1) depicts a small portion of a flight schedule, as well as corresponding lines-of-flight. In this example, flight sequences $\{1, 4\}$, $\{2, 3\}$, $\{5\}$, $\{6\}$ and $\{7\}$ each form a LOF. Table (4.1) also indicates the aircraft, A, B and C, and the rotations to which they have been assigned.

Table 4.1: Example aircraft schedule

Flight	Day	Origin	Dest.	STD	STA	Tail
1	1	SEA	TUS	08:10	10:30	A
2	1	DTW	TUS	09:05	10:35	B
3	1	TUS	BOS	11:05	15:00	B
4	1	TUS	ORD	11:20	15:30	A
5	2	BOS	DAL*	09:00	14:30	B
6	2	ORD	DFW	09:30	10:45	A
7	2	DFW	LAX	11:15	12:45	A
8	2	LAX	SEA	14:10	16:50	A
9	2	BOS	MCO*	10:30	13:45	C

*MCO and DAL are maintenance stations.

Supposed that Flight 2 from DTW to TUS is delayed by 15 minutes. Incorporating a 30 minute turn-time, the earliest the subsequent flight, Flight 3, can depart is at 11:20, 15 minutes past its scheduled departure time. Suppose also that Flight 1 has landed on time and thus, even with a 30 minute turn-time, could satisfy an 11:05 departure. The operation controllers now have the choice of performing an aircraft swap (assuming the aircraft are of the same fleet type). If an aircraft swap is performed, then the planned schedule is changed as illustrated in Table (4.2). The result of this aircraft swap is that the delay is absorbed, and both Flights 3 and 4 are able to depart on time. On the other hand, the maintenance plan for each aircraft has now been disrupted, because the swap caused the aircraft to exchange their respective routings.

Table 4.2: Example aircraft schedule *after* performing swap

Flight	Day	Origin	Dest.	STD	STA	Tail
1	1	SEA	TUS	08:10	10:30	A
2	1	DTW	TUS	09:05	**10:50**	B
3	1	TUS	BOS	11:05	15:00	**A**
4	1	TUS	ORD	11:20	15:30	**B**
5	2	BOS	DAL*	09:00	14:30	**A**
6	2	ORD	DFW	09:30	10:45	**B**
7	2	DFW	LAX	11:15	12:45	**B**
8	2	LAX	SEA	14:10	16:50	**B**
9	2	BOS	MCO*	10:30	13:45	C

*MCO and DAL are maintenance stations.

We now extend this example to add maintenance implications. Suppose that tail B, which is swapped to mitigate delay propagation, is a *day-6* aircraft, referring to an aircraft that is on day six of its maintenance rotation. Given that aircraft must be maintained every seven days, the aircraft in this example must receive maintenance at the end of the day *tomorrow*.

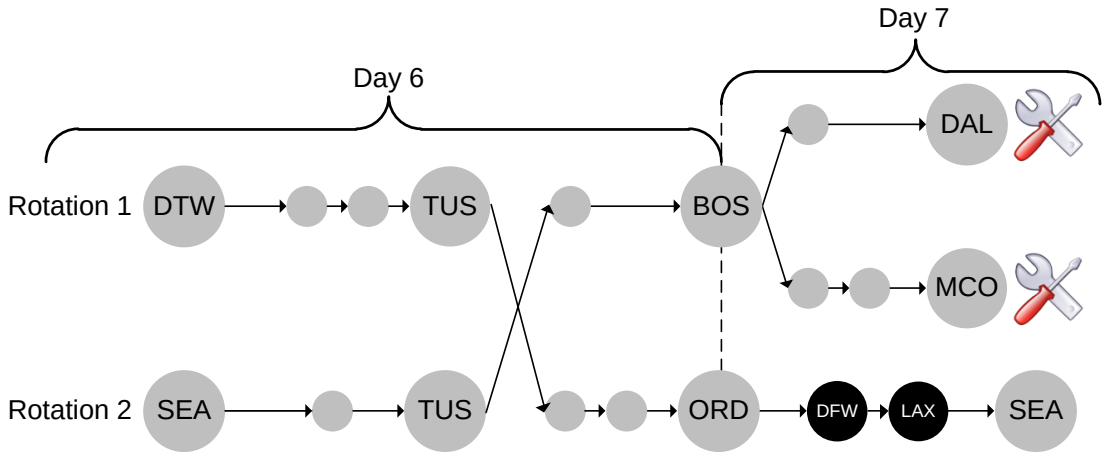


Figure 4.3: Aircraft routing on day 6 without maintenance opportunity

Performing a swap in this situation may have costly consequences for scheduled maintenance. Tail B, which started flying the first rotation, now ends the day in ORD. It turns out that the following day, day seven, the only outgoing line from ORD is destined for SEA which is *not* a maintenance base, i.e. no MLOF exists. This example is seen graphically in Figure (4.3). In this situation, we have no possible

maintenance outlets for the aircraft, and thus we must find a swap opportunity over the course of the day to ensure this aircraft enters required maintenance. In other words, the intermediate stations indicated in Figure (4.3), as DFW and LAX, must intersect with another line that does end in a maintenance station. Furthermore, if an intersection with a maintenance line is found, then this MLOF must not already be required to bring another tail to a maintenance station, such that we can perform a swap and allow the aircraft departing ORD on the beginning of day seven to enter maintenance at the end of the day.

By using swap opportunities at either DFW or LAX to bring the aircraft to its required maintenance station, we are changing our operational schedule again, i.e. forcing another aircraft to change its route, causing further network effects. Additionally, we may be breaking a *through-flight*, causing connecting passengers and crew to be disrupted. In either case, we end with a situation that may be sub-optimal and costly.

At the carrier with whom we worked, over the day swaps are pronounced. On an average day at least 20% of all LOFs incur the type of swap illustrated in the previous examples to mitigate an operational disruption. Such swaps result in aircraft ending their day-of-operations at random stations in the network with various amounts of flight-time left before maintenance must be performed. It is these significant network impacts that motivate our search for a schedule that is resistant (*robust*) to such unplanned changes.

4.3.3 *Maintenance Reachability* - A New Metric

In this section, we describe how to compute the expected number of maintenance misalignments for a given station and its set of MLOFs. This input parameter will

later feed into the optimization models presented in §4.4.1.

An aircraft that ferries passengers on a daily basis is required to receive a routine "A-Check" based on the block or flying time [88]. For the major U.S. carrier considered in this analysis, the maintenance requirement can be translated into an aircraft requiring an A-Check every seven days. To model this recurring maintenance event, we initially assign all aircraft in the fleet a 1/7 probability ($p_r = 1/7$) of being a day-7 aircraft, i.e. an aircraft that requires maintenance at the end of the day. [We later relax the assumption that $p_r = 1/7$ for all aircraft.]

Given the individual probability of requiring maintenance for each aircraft, we use the binomial distribution to derive the expected number of maintenance misalignments at station s given n MLOFs using Equation (4.1).

$$p_s^n = \sum_{i=n+1}^{L_s} \binom{L_s}{i} (p_r)^i (1-p_r)^{L_s-i} (i-n) \tag{4.1}$$

In Equation (4.1), the parameter L_s indicates the total number of LOFs exiting from station s. To determine the expected number of maintenance misalignments, we iterate through the number of LOFs beyond the availability of MLOFs and compute the probability that aircraft will require such a line of maintenance. We then multiply this probability by the number of aircraft that are misaligned, i.e. require maintenance, but do not have access to an MLOF, to compute the expected value. To illustrate this concept more clearly, we present a numerical example next.

Consider two stations, BOS and ORD, each with ten outgoing lines-of-flight. Out of the ten LOFs exiting station ORD, two end in a maintenance station, i.e. are

MLOFs. On the other hand, of the ten stations leaving BOS, zero are MLOFs. Of the ten aircraft starting the day in BOS, on average 1/7 or ≈ 1.43 will require maintenance. However, these aircraft will not have the opportunity to be routed to a maintenance station directly due to the fact that none of the LOFs leaving BOS are scheduled to end the day in a maintenance station. Hence, for BOS, the expected maintenance misalignment is 1.43. This lack of access to MLOFs forces manual intervention (precisely the scenario we aim to avoid) over the course of the day to force a swap with another line, such that a maintenance opportunity is created.

On the other hand, applying a binomial approximation to the ten LOFs out of ORD, of which two are MLOFs, we see that on average only 0.21 day-7 aircraft will be unable to reach a required maintenance station. Alternatively, it would be beneficial to reallocate one of the maintenance opportunities out of ORD to BOS so as to decrease the overall expected number of maintenance misalignments. For this example, using the binomial approximation, moving one of the two maintenance opportunities at ORD to BOS would reduce the expected number of maintenance misalignments from 1.64 $(1.43 + 0.21)$ to 1.28 $(0.64 + 0.64)$.

4.3.4 Achieving Maintenance Reachability

In our research, we aim to combat the effects of unplanned operational schedule changes, such that day-seven aircraft that end up starting the day at an unplanned station can still be assigned to an MLOF. To do so, we examine the schedule from a planning perspective and attempt to "pre-plan" for recovery from operational, over-the-day swaps, as illustrated in Figure (4.4).

In our approach, we seek changes to the original plan that can improve mainte-

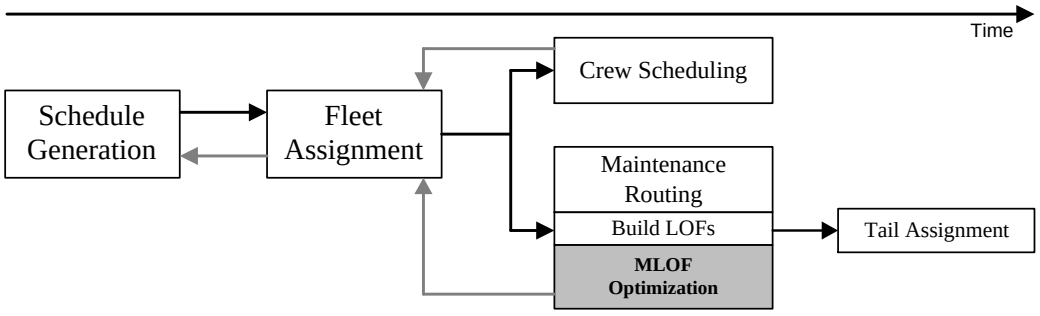


Figure 4.4: Typical airline planning process with maintenance reachability

nance reachability without substantially altering the initial lines-of-flight. We do so by identifying candidate LOFs for *splicing opportunities*. That is, when two lines-of-flight departing from differing stations intersect at some point in space and time, these lines become candidates for a line-splice. In Figure (4.5), two lines are candidates for splicing during an intersection in TUS. Such a splice would send Rotation 2 to PHX, while Rotation 1 would now end at maintenance station MCO. In effect, at the splice, the lines-of-flight of the respective aircraft are changed such that each aircraft now flies the remaining route of the other. As a result, if one line-of-flight ends in maintenance, while the other does not, a re-distribution of maintenance opportunities has occurred. It should be noted that these swaps are fleet-restricted. That is, we only consider line-swaps that involve the same aircraft type to ensure complete downstream compatibility.

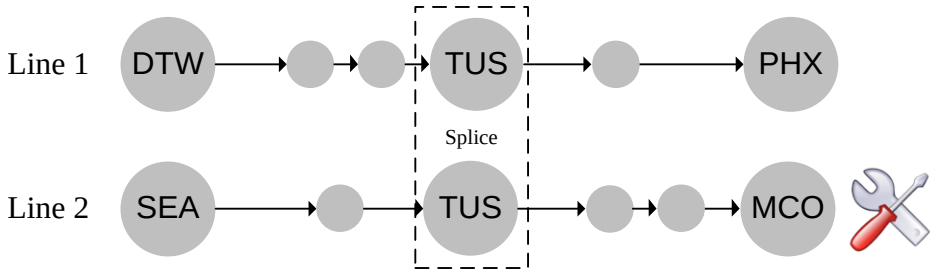


Figure 4.5: Two line splice example

This approach of performing line splices ignores the underlying network structure

and its possible effect of changing the probability of the having a day-7 aircraft. From our experience, it is not clear whether re-routing, by changing the assignment of MLOFs, will impact the probability of having a day-7 aircraft. However, as we will show in §4.5, changes in the day-7 aircraft probabilities still allow our model to outperform the original schedule.

Returning to the example in §4.3.3, if one of the ten LOFs that depart BOS and and one of the two MLOFs leaving ORD intersect at some point in time and space, we can perform a line-splice. In this example, such a line-splice results in two changes: the line leaving BOS gains a maintenance opportunity, while the line leaving ORD loses one. In other words, we have improved maintenance reachability, illustrated by the reduced number of maintenance misalignments from the numerical example in §4.3.3.

This example demonstrates an important property about the problem situation. A more *balanced* allocation of maintenance opportunities can produce an overall reduction in the expected number of aircraft requiring additional, disruptive LOF swaps. In effect, each additional maintenance opportunity has a smaller probability of being needed and thus provides incrementally less value. Therefore, for two stations with the same number of outgoing lines, the improvement of going from $n-1$ to n at one station outweighs the loss of going from $m+1$ to m at the other.

If we have m stations in the flight network with n flights ending the day in a maintenance station, then maintenance reachability is optimized by assigning exactly $\frac{n}{m}$ maintenance opportunities to each station. This optimal allocation is formally proven in Theorem (A.1) and the subsequent Corollary (A.2) in the appendix. It should be noted that this result only holds in the case where each station features the

same number of outgoing lines, a scenario unlikely to be true for any major airline.

Of course not all stations will have the same number of total outgoing LOFs, meaning that an equal allocation will most likely not be optimal. The goal of our research is to determine the optimal allocation of maintenance opportunities in such cases. This re-allocation is performed by identifying a set of line-splices, each of which re-distributes a maintenance opportunity from one station to another, such that the optimal balance is achieved.

4.4 Optimization Model

We now present an optimization model to improve maintenance reachability. This approach takes an existing flight plan, designed to ensure maintenance feasibility, provide desirable through-flight connections and keep crew with aircraft. We then identify limited changes that improve maintenance reachability without major deviations from this original plan. First, in §4.4.1, we present a mathematical model that solves a relaxation of the maintenance lines-of-flight assignment problem, providing a lower-bound on the optimal solution. That is, we solve a simplified optimization problem to determine the band of the optimal solution space. In §4.4.2 we provide an overview as to how we determine line-splicing opportunities. In §4.4.3, we present a more restrictive version of the model from §4.4.1 in which we ensure that viable line-splicing opportunities exist in order to achieve the reconfiguration of maintenance opportunities. Finally, §4.4.4 features a second-stage optimization model that achieves the maximum amount of maintenance reachability, while minimizing the number of changes to the flight plan.

4.4.1 MLOF Assignment Problem

We begin by finding a lower bound on the expected number of maintenance misalignments. We do so by optimizing the distribution of maintenance opportunities, without regard for whether this distribution can actually be feasibly achieved given the existing schedule.

We do so by making the decision to assign a particular *number* of maintenance opportunities to each station in the flight network. That is, we minimize the overall expected number of maintenance misalignments across all stations by assigning a number of MLOFs to each station. The number of MLOFs assigned to each station is constrained by the total (finite) number of MLOFs that exist in a single day-of-operations.

Sets

S	Set of stations in the flight network.

Parameters

p_s^n	Expected number of maintenance misalignments at station s given n MLOFs, $\forall s \in S, \forall n \in \{1 \dots T\}$. [The derivation of this parameter was detailed in §4.3.3].
τ	Total number of flights that terminate at a maintenance station at the end of a day.

Variables

x_s^n	Binary variable that is 1 if station s has n MLOFs assigned and 0 otherwise, $\forall s \in S, \forall n \in \{1 \dots T\}$. For example, $x_1^5 = 1$ refers to station 1 having 5 MLOFs assigned.
q_s	Integer variable that represents the number of MLOFs assigned to station s. For example, $q_1 = 5$ refers to station 1 having 5 MLOFs assigned.

Objective:

$$\min \sum_{s \in S} \sum_{n=0}^{\tau} p_s^n x_s^n \tag{4.2}$$

Subject to the following constraints:

$$\sum_{n=0}^{\tau} x_s^n = 1 \qquad \forall s \in S \tag{4.3}$$

$$\sum_{n=0}^{\tau} n x_s^n = q_s \qquad \forall s \in S \tag{4.4}$$

$$\sum_{s \in S} q_s = \tau \tag{4.5}$$

$$x_s^n \in \{0, 1\} \qquad \forall n \in \{1 \dots \tau\}, \forall s \in S \tag{4.6}$$

$$q_s \geq 0 \qquad \forall s \in S \tag{4.7}$$

In the above formulation, the objective, seen in Equation (4.2), minimizes the expected number of maintenance misalignments across all stations in the flight network. Constraint (4.3) ensures that only one numeric value of maintenance opportunities is assigned to a particular station. Constraint (4.4) provides a link between the number of maintenance opportunities assigned to station s and the numeric equivalent q_s. Finally constraint (4.5) accounts for all maintenance opportunities available for assignment in the network.

4.4.2 Determining Splice Opportunities

The previous section provided a lower-bound formulation on the allocation of maintenance opportunities to stations, based on the total number of outgoing lines-of-flight. This is done without considering the feasibility of such an allocation. In this section, we identify an optimal allocation based on modifying the existing schedule.

We begin by finding line-splice opportunities. That is, given two lines exiting from differing stations, we determine if these lines can be spliced, such that at the

point of interchange, the aircraft routing may be changed, if so desired by the optimization model. A re-assignment of aircraft at a line-splice opportunity effectively re-distributes the maintenance opportunity from the line that ends in a maintenance opportunity to the one that does not.

In this model, we restrict ourselves to simple line-splices. That is, at any point, only two lines are considered candidates for splicing. [In §6, we discuss more complex line-splices]. In order for a line-splice to occur, several conditions must be met. These conditions include:

- **Space** : The lines must intersect at a common station.
- **Time** : The lines must intersect at this station at the same time.
- **Aircraft** : The aircraft type must match (assuming crews are not interchangeable).
- **Benefit** : A splice is only worth considering if one of the two lines gains a maintenance opportunity while the other loses one. (Splicing two MLOFs or two non-MLOFs provides no operational benefit).
- **Line-Locks** : Certain connections within a line may be "locked", eg. they represent a high-value one-stop itinerary that should not be altered.

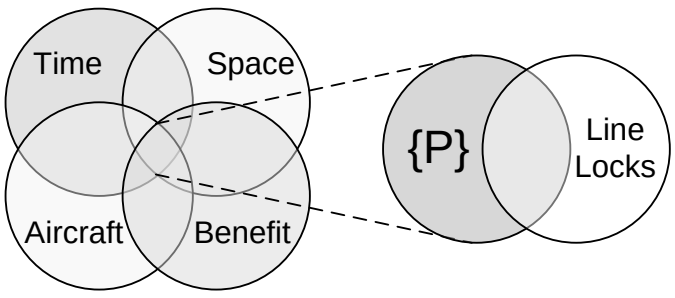


Figure 4.6: Overlap diagram illustrating swap opportunities

It is the overlap in the above-mentioned factors, and as illustrated in Figure (4.6), that provide a *splice-opportunity* during the pre-planning process. Thus, our solution approach is now two-fold. First, we determine all splice-opportunities that exist in a schedule. This information then serves as input to the optimization model (presented next) to determine which splicing opportunities should be implemented to minimize the expected number of maintenance misalignments.

4.4.3 Restricted MLOF Assignment Problem

Given a set of candidate line-splice opportunities, derived from the criteria as introduced in the previous section, we now pose an optimization model that selects the optimal set of line-splices to perform so as to minimize the total number of expected maintenance misalignments. While the objective function remains the same as in the previous model, we now account for a line-splice that takes place as part of the optimization process. In essence, we attempt to match the previous model, except we now limit the decisions to include only valid line-splices.

Sets

P	Set of all possible line-splices.
L	Set of all lines-of-flight in the flight network.
S	Number of stations in the system.

Parameters

m_s	Number of MLOFs associated with station s in the original schedule, $\forall s \in S$
p_s^n	Expected number of maintenance misalignments at station s given n MLOFs, $\forall s \in S, \forall n \in \{1 \dots T\}$. [The derivation of this parameter is detailed in §4.5].
δ_{sp}^+	Binary parameter that is 1 if splice p increases the number of maintenance opportunities at station s by 1 and 0 otherwise, $\forall s \in S, \forall p \in P$
δ_{sp}^-	Binary parameter that is 1 if splice p decreases the number of maintenance opportunities at station s by 1 and 0 otherwise, $\forall s \in S, \forall p \in P$
ω_{pl}	Binary parameter that is 1 if line l is part of splice p, and 0 otherwise, $\forall p \in P, \forall l \in L$
τ	Total number of flights that terminate at a maintenance station at the end of a day.

Variables

x_s^n	Binary variable that is 1 if station s has n MLOFs assigned and 0 otherwise, $\forall s \in S, \forall n \in \{1 \dots T\}$.
y_p	Binary variable that is 1 if splice p is performed and 0 otherwise, $\forall p \in P$

Objective:

$$\min \sum_{s \in S} \sum_{n=0}^{\tau} p_s^n x_s^n \tag{4.8}$$

Subject to the following constraints:

$$\sum_{n=0}^{\tau} x_s^n = 1 \qquad \forall s \in S \tag{4.9}$$

$$\sum_{n=0}^{\tau} n x_s^n = m_s + \sum_{p \in P} \delta_{sp}^{+} y_p - \sum_{p \in P} \delta_{sp}^{-} y_p \qquad \forall s \in S \tag{4.10}$$

$$\sum_{p \in P} \omega_{pl} y_p \leq 1 \qquad \forall l \in L \tag{4.11}$$

$$x_s^n \in \{0, 1\} \qquad \forall n \in \{1 \dots \tau\}, \forall s \in S \tag{4.12}$$

$$y_p \in \{0, 1\} \qquad \forall p \in P \tag{4.13}$$

As before, the objective as seen in Equation (4.8) is to minimize the total number of expected maintenance misalignments. Constraint (4.9) ensures that only one number of maintenance opportunities is assigned to each station. Constraint (4.10) performs the numerical assignment based on the splice choices (y_p) made. That is, for every splice that adds an MLOF to station s, we increase the total count by 1. On the contrary, when a splice is made that removes an MLOF from station s, we decrease the total number of MLOFs for station s by 1. Finally, constraint (4.11) requires that all lines are spliced at most once.

4.4.4 Minimization of Schedule Changes

There may be multiple equivalent solutions to the mathematical model presented in §4.4.3. For example, suppose we have two stations, BOS and SEA, both with two lines-of-flight. For each station, one of the LOFs ends in a maintenance station while the other does not. Further suppose that both LOFs from BOS can splice with both LOFs from SEA. The optimization approach presented thus far could splice the MLOF from BOS with the LOF from SEA and the LOF from BOS with the

MLOF from SEA. The net-effect of these splices is zero with respect to maintenance reachability, unnecessarily changing the original planned LOFs.

To minimize the impact in terms of the number of changes to the schedule, we formulate a second-stage optimization model. This optimization model minimizes the total number of line-splices required to achieve the objective function value as derived during the first-stage optimization problem presented in §4.4.3. We denote the objective function value of the previous optimization model by C, an input parameter to this model.

Objective:

$$\min \sum_{p \in P} y_p \tag{4.14}$$

Subject to the following constraints:

$$\sum_{n=0}^{T} x_s^n = 1 \qquad \forall s \in S \tag{4.15}$$

$$\sum_{s=1}^{S} \sum_{n=0}^{T} p_s^n x_s^n = C \tag{4.16}$$

$$\sum_{n=0}^{T} n x_s^n = m_s + \sum_{p \in P} \delta_{sp}^{+} y_p - \sum_{p \in P} \delta_{sp}^{-} y_p \qquad \forall s \in S \tag{4.17}$$

$$x_s^n \in \{0, 1\} \qquad \forall n \in \{1 \dots T\}, \forall s \in S \tag{4.18}$$

$$y_p \in \{0, 1\} \qquad \forall p \in P \tag{4.19}$$

In the above formulation, objective (4.14) seeks to minimize the total number of line-splices that are made. We are re-using the notation from the previous formulation, such that y_p is a $\{0, 1\}$ variable allowing for a summation of the total number of splices performed. Constraints (4.15) ensure that we still only assign a single number of maintenance opportunities to each station. Constraint (4.16) requires that our

new assignment must yield the same objective value (C) as the value obtained during the previous optimization step. Finally, constraints (4.17) perform the required accounting of maintenance stations, as in the first-stage optimization model.

4.5 Computational Results

To validate the optimization model introduced in §4.4, we provide results from computational experiments that: demonstrate the tractability of our approach, provide a detailed summary of the quality of the solution and its impacts, and assess the sensitivity of the model to the probabilistic objective coefficients.

To do so, we first use the approximation of the expected number of day-7 aircraft at a station requiring maintenance at the end of the day-of-operations, as developed in §4.3.3. We then refine this approximation using carrier-based data, showing the impact of more precise coefficients on the overall solution quality. Finally, we conduct a sensitivity analysis to determine how much impact errors in the objective coefficients have on the quality of the solution that is found.

The input data for our model was sourced from a major U.S. carrier. The carrier provided two weeks' worth of tail assignments from which LOFs were built with maintenance and non-maintenance stations indicated. Additional details regarding the input plan can be found in Table (4.3).

Number of Flights:	3353
Number of LOFs:	512
Number of Stations:	64
Number of MLOFs:	206

Table 4.3: Characteristics of input data for optimization model

Using the criteria provided in §4.4.2, Table (4.4) illustrates the number of line-splicing opportunities in one-day worth of data. It should be noted that the last two rows of this table are indicative of the size of the problem that must be solved by the optimization models.

Airport:	126,507
Airport and Time:	8,815
Airport, Time and Benefit:	4,076
Airport, Time, Benefit, Aircraft:	2,074
Airport, Time, Benefit, Aircraft and Unlocked:	1,728

Table 4.4: Line-splicing opportunity count for various conditions

4.5.1 Experimental Results

We begin by first solving the lower-bound formulation presented in §4.4.1, followed by the schedule modification model presented in §4.4.3. Using the seven-day maintenance assumption with p_s^n approximated using the binomial distribution presented in Equation (4.1) with $p_r = 1/7$, we obtain the following results in terms of the expected number of maintenance misalignments.

Schedule without Optimization:	7.33 misalignments
Lower-bound Optimization:	**0.84** misalignments (-88%)
Optimized Schedule (with $p_r = 1/7$)	**0.84** misalignments (-88%)
Runtime:	< 5 seconds*

*Computation performed on an Intel Duo-Core 2.2GHz processor with 4GB of memory using CPLEX 11.0 C++ API

Table 4.5: Experimental results, including runtime using ($p_r = 1/7$)

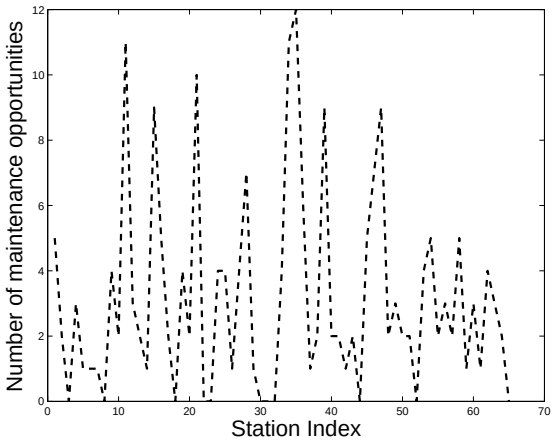
The optimization model is able to reduce the expected number of maintenance misalignments by 88%. This means that when the newly formed LOFs are operated, on average, our plan will incur 0.84 day-7 aircraft (as opposed to 7.33) that will require manual re-routing to end their day in a maintenance station. Note that the

restricted optimization model is able to achieve the same objective value as the lower-bound. Further detail as to the implications of these results on the actual aircraft plan and the airline are detailed in §4.6.

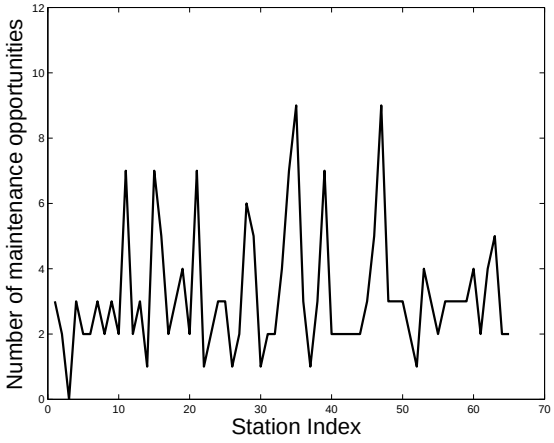
Figure (4.7(a)) and Figure (4.7(b)) illustrate the allocation of maintenance opportunities in the original and optimized sets of LOFs. Overall, we notice a general re-balancing of maintenance opportunities, with much lower peaks but nonetheless higher numbers of maintenance opportunities at stations with larger numbers of LOFs. Evident from these figures, the variability in terms of the number of maintenance opportunities is lower than pre-optimization. Correspondingly, Figure (4.8) shows an overall decrease in the expected number of maintenance misalignments at every station. In this figure, we provide the original expected number of maintenance misalignments (dashed-line), as well as the expected number of maintenance misalignments under the new LOF configurations (solid-line).

4.5.2 Probability Scenarios

Having first considered the simple case where all aircraft at all stations are equally likely to be day-seven aircraft (with a probability of 1/7), we next conduct experiments using empirical data from a major U.S. carrier to provide insights into whether using more finely-tuned values for the probability of a day-7 aircraft will change the results of the optimization. To evaluate the effectiveness of our optimization model under this data set, we evaluate five probability vectors, as provided in Table (4.6). Each vector provides empirical estimates, from different periods of time, of the probability of a day-7 aircraft appearing at a maintenance station (m), a large station (l) and a small (s) station. Our primary data set consists of 10, 12 and 42 of these respective stations. A "large" station differentiates itself from a "small" station by



(a) Maintenance opportunity assignment - Original Plan



(b) Maintenance opportunity assignment - Optimized P_0

Figure 4.7: Maintenance opportunity assignment using ($p_r = 1/7$) approximation

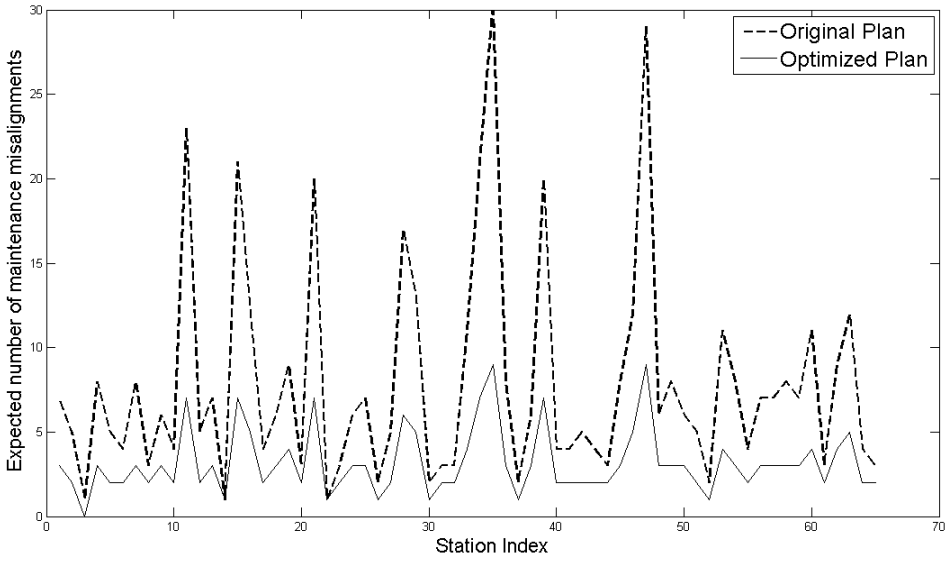
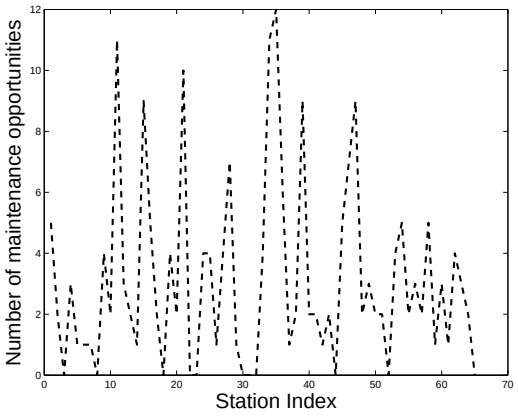


Figure 4.8: Expected number of maintenance misalignments using ($p_r = 1/7$)

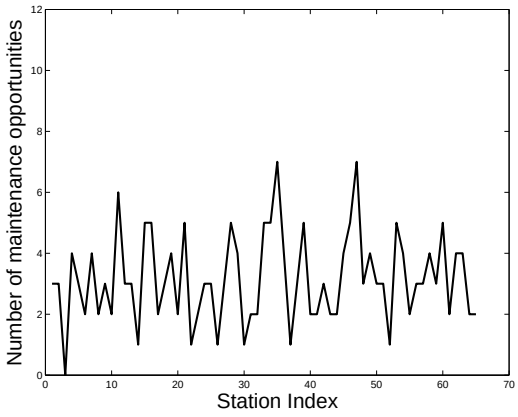
the number of LOFs that depart each day. Following the convention provided by our partner airline, a large station is defined as one which has at least seven outgoing LOF. It should be noted that such an aggregation of stations by type is not required by the model, which takes as input a potentially-different value of p_r for each station. In addition, Table (4.6) provides the required number of MLOFs, determined by the number of day-7 aircraft at each station in the network. The remaining capacity refers to the difference between the total MLOFs available and the number of day-7 aircraft at each station.

	MX	Large	Small	Req'd MLOFs	Remaining MLOF Capacity
P_0	0.1429	0.1429	0.1429	73	64.4%
P_1	0.1162	0.1926	0.1689	84	58.7%
P_2	0.0462	0.1824	0.1181	61	69.9%
P_3	0.0829	0.1585	0.1048	58	72.8%
P_4	0.1632	0.1722	0.1571	82	59.8%
P_5	0.0653	0.1216	0.0685	41	80.2%

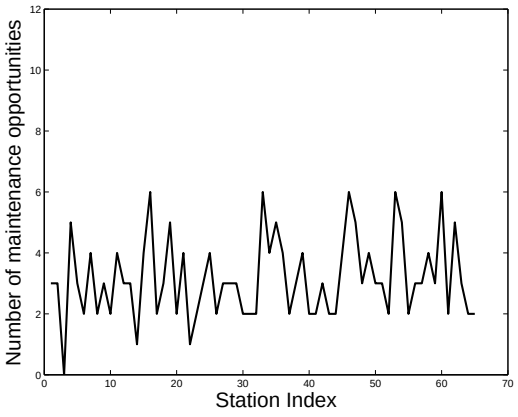
Table 4.6: Probability vectors based on actual US carrier data



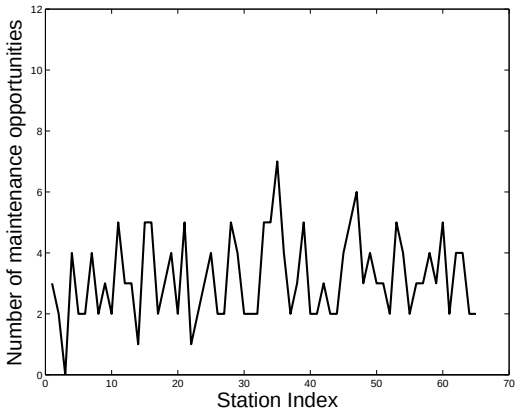
(a) Maintenance opportunity assignment - Original Plan



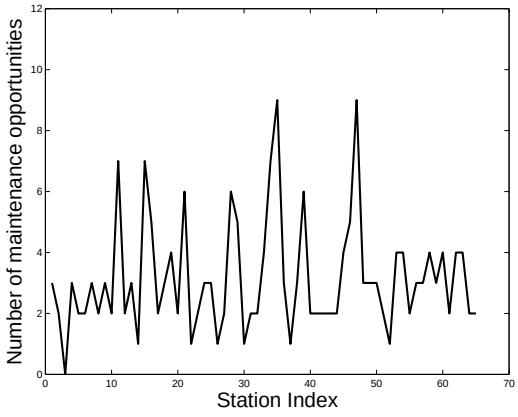
(b) Maintenance opportunity assignment - Optimized P_0



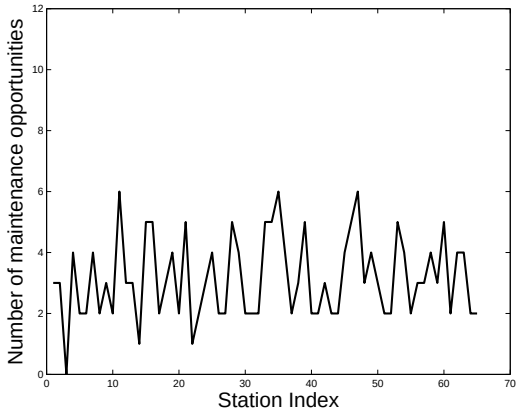
(c) Maintenance opportunity assignment - Optimized P_1



(d) Maintenance opportunity assignment - Optimized P_2



(e) Maintenance opportunity assignment - Optimized P_3



(f) Maintenance opportunity assignment - Optimized P_4

Figure 4.9: Maintenance opportunity assignment comparing each of the probability scenarios

Using the parameters shown in Table (4.6) along with the data set described in Table (4.3), we solve six different instances of the problem, i.e. we generate six optimal sets of LOFs, one for each probability vector. Probability vector P_0 denotes the case where all station types have probability $p_r = 1/7$ as assumed in §4.3.3. We then evaluate how each of these sets of LOFs performs under all of the other probability vectors. In other words, we ask the question: If we assume one probability vector to represent the true parameters and optimize accordingly, but another probability vector is in fact the true data, how will our "optimized" LOFs perform? This question is answered in Table (4.7). Each row corresponds to a given set of LOFs −− the original set of LOFs and the results of optimizing over each of the six probability vectors, as shown given in the left-most column. The remainder of each row then gives the objective value for that set of LOFs as evaluated under each of the six probability vectors. For example, if probability vector P_1 is used to optimize the LOFs but then P_2 is realized, the expected number of maintenance misalignments will be 0.45.

		Probability Vectors					
		P_0	P_1	P_2	P_3	P_4	P_5
LOFs	Original	7.33	8.98	5.64	5.01	8.69	3.02
	P_0	**0.84**	1.51	0.95	0.62	1.39	0.24
	P_1	1.14	**1.10**	0.45	0.35	1.89	0.14
	P_2	3.17	1.93	**0.31**	0.56	4.90	0.23
	P_3	1.33	1.15	0.42	**0.34**	2.20	0.13
	P_4	0.87	1.45	0.81	0.54	**1.35**	0.21
	P_5	1.35	1.16	0.42	0.34	2.25	**0.13**

Table 4.7: Objective value comparison for scenario data

In Table (4.7), the optimal values are highlighted for each set of LOFs and its corresponding probability vector. Of course, by definition, this value is the lowest value in each column. These values illustrate the impact of the optimization model under

perfect information with each optimized solution offering a significant improvement in terms of maintenance reachability. In addition, the remainder of the columns show that even without perfect information, the optimization model is able to reduce the total number of maintenance misalignments relative to the original schedule. This improvement is due the relationship between the total number of outgoing LOFs and MLOFs available for assignment. Irrespective of the day-7 probability, stations with a higher number of outgoing LOFs generally receive a greater share of the available MLOFs.

Furthermore, we graphically compare the effects of the various probability scenarios on the final MLOF assignment in Figure (9). Even under uncertain probabilities, the optimization changes the maintenance opportunity assignment typically by +1 or -1 MLOF per station. As noted, regardless of which probability scenario is used, the impact of the optimization is noticeable when compared to the original schedule; however, between probability scenarios, the change to the final maintenance opportunities assigned to each station is minimal.

	Probability Vectors					
	P_0	P_1	P_2	P_3	P_4	P_5
Original	7.33	8.98	5.64	5.01	8.69	3.02
Worst-Case Scenario	3.17	1.93	0.95	0.62	4.9	0.24
Best-Case Scenario	0.84	1.10	0.31	0.34	1.35	0.13
Max Improvement	6.49	7.88	5.33	4.67	7.34	2.89
	88.5%	87.8%	94.5%	93.2%	84.5%	95.7%
Min Improvement	4.16	7.05	4.69	4.40	3.79	2.78
	56.8%	78.5%	83.2%	87.6%	43.61%	92.1%

Table 4.8: Objective function value (maintenance misalignments) improvement

As noted in Table (4.8), under various probability scenarios, the impact of the optimization model is still significant. That is, even under imperfect information

regarding the location of day-7 aircraft, the line-splices suggested by the optimization yield a reduction in the expected maintenance misalignments across the flight network. In addition, we note a relationship between the optimization model's performance in terms of the improvement as noted in Table (4.8) and the available MLOF capacity in Table (4.6). As the available capacity of additional MLOFs increases (MLOFs that are *not* required by day-7 aircraft), so does the possible improvement. We attribute this result to the impact of assigning available MLOF capacity to stations that reap the greatest incremental benefit from such an additional MLOF.

4.5.3 Sensitivity of Expectation Coefficients

In the previous section we considered a variety of possible values for the probability vector P. We showed that even if the incorrect value of P was used to optimize the lines-of-flight, the resulting schedule (when evaluated under the true value of P) yields higher maintenance reachability than the original schedule. In this section, we further explore the relationship between P, the "optimized" LOFs, and the realized maintenance reachability. We show that in most cases, even with large deviations from the actual probability, the "optimized" LOFs still gain relative to the original LOFs with respect to maintenance reachability.

As introduced in the previous section, we divide stations into three different types: maintenance (m), large (l) and small (s). We use a probability vector (p_m, p_l, p_s) that indicates the probability that an aircraft at each of the respective station types is a day-7 aircraft. In this analysis, we vary the probability of a day-7 aircraft from 0.04 up to 0.2 incrementing by 0.04 for each of the station types, resulting in a total of 125 scenarios. We optimize with respect to each of these probability vectors and then evaluate each of these 125 optimized plans relative to the other 124 probability

vectors. We quantify the results from this analysis next.

To do so, we first define the notion of *perfect information*. By this, we consider the case where the probability vector under which the LOFs were optimized is in fact the realized probability vector. Clearly, this is the best case scenario, whereas the worst case scenario is when the LOFs are optimized relative to a probability vector that is very different from the realized probability vector. For the case of perfect information, in all 125 cases, the optimized LOFs performed *strictly* better than the original schedule. Table (4.9) shows the average improvement across all 125 probability vectors.

Average (across scenarios) of Original Plan:	6.27 misalignments
Average (across scenarios) of Optimized Plan:	0.57 misalignments
Average Improvement:	5.70 misalignments (-90%)

Table 4.9: Maintenance misalignments improvement across all probability vectors

As alluded to before, our optimization model also performs well in situations where the LOFs are optimized under imperfect information, i.e. relative to the wrong probability vector. Consider an arbitrary probability vector. Suppose that we optimize relative to this probability vector and then evaluate this set of LOFs under all other possible vectors. If we take the maximum of each of these objective function values *and* this maximum is lower than the original plan, then it must be the case that even if the probabilities of a day-7 aircraft are unknown, the optimization still outperforms the original plan. In our analysis, there are 71 out of 125 cases where the optimized LOFs out-perform the original LOFs, *independent of what probability vector is actually realized*. That is, if the input data is erroneous and thus deviates far from the actual day-7 aircraft probability, the optimization model presented here *still* outperforms the original assignment, illustrating the stability of our approach.

In the remaining 54 scenarios the optimized LOFs are not guaranteed to perform as well as the original schedule under *every* scenario, but only when the optimized plan is based on a probability vector that deviates by least 0.66 in total error from the actual realized probability vector is the optimized schedule worse than the original. Even in these 54 scenarios, the optimized LOFs out-perform the original LOFs on average.

4.6 Impact & Conclusions

Aircraft routings are designed to meet strict FAA mandates for aircraft maintenance intervals. As emphasized in this paper, on any given day, these plans are routinely changed to mitigate unforeseen events, such as delay propagation, which disrupt airline schedules. To evaluate these day-of-operation changes and their impact on maintenance routing, we provide a metric, known as *maintenance reachability*. This metric captures a schedule's ability to satisfy maintenance constraints. More specifically, for each station in the flight network, maintenance reachability quantifies the ability of an aircraft to reach a maintenance opportunity on its last day-of-operation.

When maximizing maintenance reachability, the goal is to minimize the expected number of aircraft that are *unable* to reach a maintenance station on their seventh day of operation using existing lines-of-flight. As we have shown, this is done by correlating the number of maintenance opportunities with the total number of lines-of-flight for each station. In special cases, where each station has the same number of LOFs, the optimal solution is one where each station has the same number of maintenance opportunities. However, since airlines rarely have the same number of LOFs leaving each station, an optimization model is required to perform the

assignment of maintenance opportunities to stations, so as to minimize the expected number of day-7 aircraft without a maintenance opportunity.

The optimization model presented in this paper provides the necessary network balance of maintenance opportunities to maximize maintenance reachability. That is, we strategically splice lines-of-flight to change the allocation of maintenance opportunities from stations with less need to stations with greater need. Using a sequential optimization approach, we are able to show significant improvements with respect to maintenance reachability by making no-cost changes to the flight plan. As a result, the new schedule is more robust to unexpected changes that may occur on the day-of-operation, reducing the need for costly interventions of manual re-routing aircraft to ensure maintenance feasibility. While our mathematical model depends on probability parameter estimates, we show through a sensitivity analysis that those parameter estimates can often deviate from the actual probability while still producing an improved schedule.

4.6.1 Future Work

To develop this model further, several avenues of extension exist.

Maintenance Capacity Constraints

While the optimization model presented in this paper does assign maintenance opportunities to stations, it does so without constraints on the actual maintenance station. Since maintenance stations have a finite capacity of the number of aircraft that can be serviced on any given evening, a further extension of this model could incorporate a maintenance balancing decision when making maintenance opportunity assignments. More specifically, the maintenance opportunities and their respective maintenance station should be balanced across each of the stations. Such an assign-

ment would allow for a more equal work-balance in terms of the number of aircraft that will actually require maintenance at the end of a day-of-operations. Alternatively, capacity constraints could be assigned to each maintenance station, restricting the number of maintenance tasks at each facility in accordance with its capabilities.

Complex Line-Splicing

As mentioned in §4.1, this model was developed with input from a major U.S. carrier. Within this carrier's network, we were able to achieve the lower bound on maintenance reachability by employing single line splices. This may not be the case for other carriers' networks. For example, one can imagine that if a required maintenance opportunity cannot be obtained through single line splices, another level of splicing could provide additional maintenance access. That is, line A is spliced with line B for no gain in maintenance reachability. However, further down the line-of-flight, line B is spliced with line C, creating a maintenance opportunity for the aircraft flying A. Future research would investigate alternative levels of splicing, and the corresponding modification of the optimization model, to achieve higher levels of maintenance reachability.

In addition to more complex line-splicing, existing maintenance robustness can also be evaluated and improved through "like-swaps". Specifically, having two MLOFs that intersect with an opportunity for an over-the-day swap can enable a carrier to make an operational recovery decision that does not disrupt a day-seven aircraft en route to a maintenance station at the end of the day. Increasing the number of such swap opportunities in the planning process could further improvement maintenance robustness.

Through-Flight Connection Cost

Finally, in our model, we assume that line-splices come at virtually no cost. In fact, there may be a revenue impact associated with breaking an existing through-flight connection, and/or an operating cost associated with changing crew/aircraft pairings. Conversely, new LOFs may create new revenue opportunities (by introducing new through itineraries) and new opportunities for operational savings. Thus, it may be worth extending the existing model to incorporate the costs and benefits of LOF modification.

CHAPTER V

Aircraft Maintenance Recovery Problem

5.1 Introduction

Aircraft maintenance is an integral part of any airline and provides the foundation for operating safely and effectively. In addition, aircraft maintenance accounts for a large portion of the operating expenses for any airline. For example, in 2010, Delta spent a total of $1.6 billion USD, about 5% of its operating expenses, to maintain its fleet of approximately 750 aircraft, while US Airways spent approximately 6.5% of its operating expenses in 2010 on aircraft maintenance.[1]

Several weeks prior to the day-of-operations, airlines solve the *Tail Assignment Problem (TA)*, in which specific aircraft are assigned to multi-day sequences of flights, commonly known as *aircraft rotations*, and their associated maintenance events, i.e. the assignment of specific maintenance activities to specific airports on specific days. The aircraft rotations directly affect aircraft maintenance in two ways. First, a rotation determines when an aircraft overnights at an airport where maintenance can take place (a *maintenance station*) and thus when the aircraft has *maintenance opportunities*. Second, the rotation defines an aircraft's *maintenance counters*, e.g. the number of flight hours and the number of take-offs and landings. Given an

[1]This data was obtained from publicly filed 10-K reports by Delta Airlines and US Airways, respectively.

aircraft's current maintenance counters and assigned rotation, we can determine when this aircraft will be due for a routine maintenance check. Thus disruptions to the rotation will in turn disrupt the maintenance plan. Such disruptions are quite commonplace, due to unexpected operational events, such as aircraft failure, weather related incidents or propagated delay.

In this chapter, we address and recover disruptions to the maintenance plan. First, we formulate and solve the *Maintenance Recovery Problem (MRP)* to find an updated maintenance plan that is feasible relative to a given set of rotations that have been modified in response to a disruption. In our work, we assume that rotations have been fixed and use this assignment as input to fix the maintenance plan. Next, we compute the set of upcoming maintenance events, one for each type of check, that each aircraft requires to continue legal operation. Our MRP is a type of maximum coverage problem, in which the objective is to cover as many maintenance events as possible, given the underlying network structure of the assigned rotations. Second, we recognize that there may be multiple optimal solutions to the maximum coverage problem, and thus we provide a second-stage optimization model that assigns maintenance activities based on additional objectives. Finally, we provide a methodology by which we integrate changes in the assigned rotations with maintenance recovery to improve maintenance coverage.

The remainder of this chapter is organized as follows. In §5.2 we provide a brief background on the airline planning and recovery processes from a maintenance perspective. In §5.3 we provide an overview of the current literature on maintenance planning and recovery problems. In §5.4 we detail the first-stage optimization model maximizing maintenance coverage. Next, in §5.5, we consider two secondary objec-

tive functions for maintenance recovery. In §5.6 we extend the initial formulation by integrating the tail assignment process into our maintenance recovery approach. Finally, we offer concluding remarks and ideas for future work in §5.7.

5.2 Airline Planning and Implementation

Airline processes can be divided into two categories: planning and implementation. While the maintenance recovery problem falls into the implementation stage, it is still beneficial to briefly illustrate the steps in the planning phase, as these steps provide the necessary input for the recovery problem.

5.2.1 Maintenance Terminology

In the remainder of this chapter, we use the following terminology. *Maintenance counters* define the accounting standards in aircraft maintenance. Such counters generally include: 1) *flight-hours*, 2) *cycles* and 3) *calendar days*. Here, flight-hours refer to the *off-block* to *on-block* time, i.e. the time the aircraft departs the origin gate until it arrives at the destination gate. Aircraft cycles count the number of take-offs and landings performed by an aircraft over the course of a specific time horizon. It is important to realize the connection between a day-long sequence of flights that will be flown by a single, common aircraft, i.e. *line-of-flight or LOF*, and its implication on aircraft maintenance. A line-of-flight contains a set of flights, each with a value for a respective maintenance counter that when summed defines the number of flight-hours and cycles for the entire LOF.

Next, each type of aircraft is required to periodically undergo a set of *maintenance checks*, where the frequency depends on the values of different maintenance counters. When solving the Tail Assignment Problem, feasibility is established by

guidelines set forth by the Federal Aviation Administration (FAA) in conjunction with aircraft manufacturers in the United States. For example, in our work, an aircraft must receive a small-size (A) check every 40 flying-hours to continue legal operation. Although our approach is not limited to specific checks, we consider those checks common to most commercial airlines, including small (A) checks, medium (B) and large (C) checks in our computational experiments.

A maintenance check is further defined by a man-hour requirement at a maintenance-capable station in the flight network. This requirement comes in two forms. First, for a check to be completed at a maintenance station, the aircraft must be on the ground for a sufficiently-long time window, e.g. 8 hours (typically overnight). Furthermore, each check features a specific number of man-hours that must be allocated (from the total available capacity at a maintenance station) for this particular check. It should be noted that each station provides only a finite number of man-hours of available maintenance capacity on any given day. Both of these limits (time-window and available capacity) must be observed for a maintenance check to take place.

Next, we define *maintenance events* as aircraft-specific maintenance checks relative to the state of the system on the day-of-operations, i.e. *time-zero*. In other words, given the current set of counters of a specific aircraft and its planned future rotation, we can identify the *next* required maintenance event of each check type that must be completed by each aircraft. We note that our focus in this preliminary research is to schedule the first of each (recurring) type of check — that is, we only schedule one check of each type for each aircraft. Furthermore, for each of these maintenance events, we define a deadline. That is, given an aircraft's current assignment, in terms of the lines-of-flight it is scheduled to fly, we can compute the

deadline by which a particular maintenance event must be completed.

Finally, aircraft can only receive maintenance at specific stations in the flight network. These maintenance stations provide man-power capacity for a subset of maintenance checks that may be performed at the respective station. When an aircraft visits one of these stations during an overnight stop, we refer to this as a *maintenance opportunity*. That is, if a tail is scheduled to spend the night at a maintenance station (maintenance is generally performed overnight when aircraft utilization is low) which is equipped to perform maintenance for the appropriate fleet type and of the appropriate check type, then an opportunity exists. Visually, the relationship between maintenance counters, checks, events and opportunities is shown in Figure (5.1).

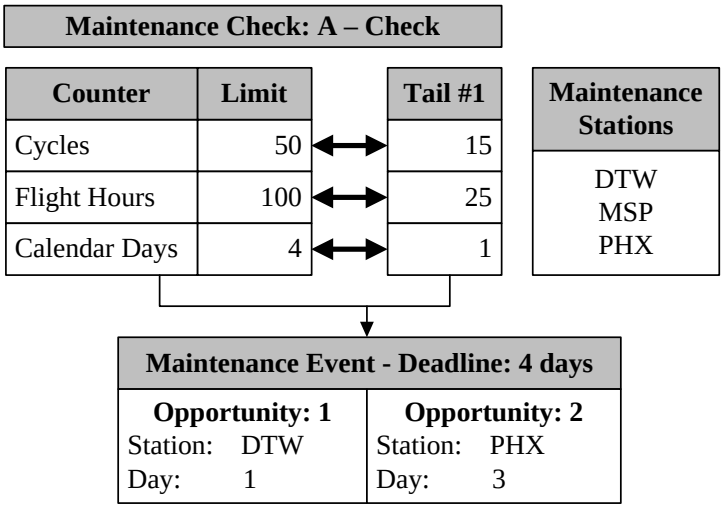


Figure 5.1: Maintenance recovery terminology illustration for a specific tail, Tail #1

5.2.2 Planning Process

To provide background for the recovery problem, we first briefly review the airline planning process. More details about this process can be found in Chapter II of this thesis or in [33], both of which provide detailed information on the planning process.

As demonstrated in Figure (5.2), the airline planning process is typically decomposed into stages occurring at different time points during the planning horizon. In the fleet assignment problem, specific aircraft types are assigned to each flight, with the goal of matching aircraft capacity to demand. This fleet assignment is subsequently used in the crew scheduling and the routing and maintenance planning stages. In crew scheduling, we build crew pairings for each fleet type which are later turned into crew schedules. The fleet assignment specifies the type of aircraft that must be used for each flight in the routing and maintenance planning stage. Furthermore, this stage creates non-tail specific lines-of-flight (LOFs), day-long sequences of flights, that are used in long-term maintenance capacity planning. In addition, these LOFs determine, among other things, thru-connections and provide information about crew turns. These LOFs may also be grouped further into multi-day aircraft rotations, which determine maintenance capacity requirements at stations in the network. The line-of-flight and rotation decisions made in this stage are typically planned to recur (for domestic U.S. operations) repeatedly every day in the planning horizon.

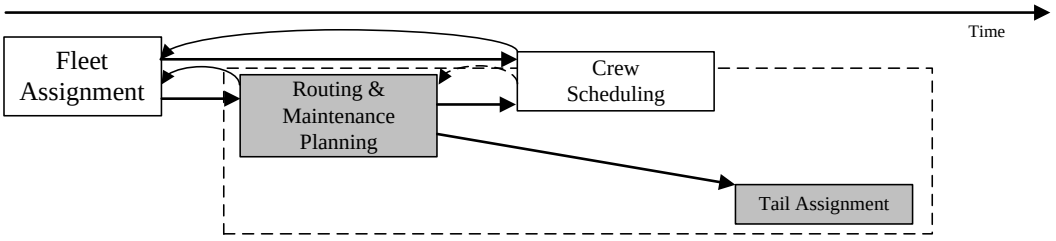


Figure 5.2: Airline planning and execution processes

During the short-term planning phase, airlines solve the Tail Assignment Problem through which specific aircraft (i.e. tails) are assigned to rotations, ensuring that every LOF on every day of the upcoming time horizon is covered. Also, within

the Tail Assignment Problem all vital maintenance events are assigned. As these maintenance feasible rotations are assigned to aircraft, each is then matched to long-term capacity plans [56].

5.2.3 Maintenance Recovery Problem Overview

Upon reaching the day-of-operations, the airline planning process has ended and the implementation stage has begun. However, upon execution, this particular plan may change due to unforeseen circumstances. For example, a station manager may request to swap two aircraft to mitigate a delay situation. When such a swap occurs, aircraft will generally trade the entire remainder of their respective lines-of-flight, which alters the remaining rotation and thus may make the current maintenance plan infeasible, either because the maintenance counters will change or because the tail will no longer reach an intended maintenance station at the right time.

In a disruption the rotations that are assigned to aircraft are modified. We take this updated assignment and determine the set of immediate maintenance events that must now be completed on each aircraft in the near future, relative to these new assignments. With the recovered schedule and all identified maintenance events, we can formulate and solve an optimization problem in which we assign the maximum number of maintenance events to newly created opportunities in our flight plan, while observing the capacity limit that exists at each of the maintenance facilities.

We conclude this section by emphasizing the fact that in our proposed *MRP* approach, we do not alter the rotations assigned to aircraft. That is, a recovered (changed) assignment is provided as input to our model, and we adjust the maintenance activities relative to this new assignment. While we do consider changes to the assigned set of rotations in §5.6, we do not change the composition of the

lines-of-flight, because such alterations create other operational issues, such as a broken thru-connections and incorrect crew assignments, all which can pose significant financial implications for an airline. Furthermore, we assume that in our formulation, the maintenance checks of interest are the ones that must be completed as soon as possible. Finally, as mentioned earlier in this chapter, we do not consider maintenance event recurrence, but address this topic in Chapter VI. In section §5.7, we discuss additional extensions including incorporating multiple instances of checks and the implications of down-stream recurrence.

5.2.4 Sample Recovery Allocation

To further illustrate MRP, we consider a small instance of this problem. Consider a tail, Tail #1, on the beginning of the day-of-operation. The tail's current maintenance counter status is shown in Figure (5.3). For simplicity, we only consider A and B-checks in this example. Figure (5.4) presents the rotation for this tail and its possible maintenance opportunities. Based on this rotation and its current status, the tail must complete an A-check in the near future. Upon execution, the maintenance recovery problem can choose to perform the A-check at the termination of line-of-flight 3 at the end of day 3 as seen in Maintenance Assignment 1. This would reset the counters for the A-check at the beginning of day 4 as shown in the diagram.

Tail #1		Current	Limit
A-check	Hours (H):	0	40
	Cycles (C):	0	10
B-check	Hours (H):	20	100
	Cycles (C):	10	50

Figure 5.3: Tail #1: Beginning maintenance counter status

Alternatively, the A-check could also be assigned at the termination of line-of-

flight 2 as seen in Assignment 2. In this assignment, maintenance counters for the A-check are reset to zero at the beginning of day 3. Contrasting the assignments, Assignment 1 leads to lower maintenance counters towards the next check as the current check is performed closer to its deadline, while Assignment 2 performs the A-check one day early. This example illustrates the various choices that exist when scheduling a maintenance event for an aircraft. In §5.5, we explore methods by which we can quantify the difference between these possible choices for maintenance event assignments.

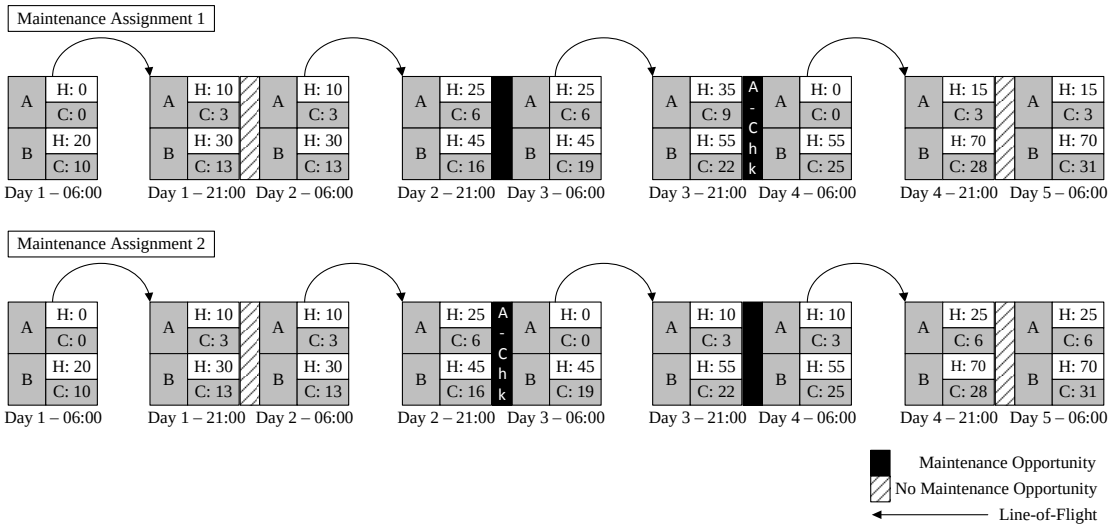


Figure 5.4: MRP maintenance decision example

5.3 Literature Review

Airline maintenance planning has been studied extensively. The survey paper by [53] provides a general overview of the aircraft (tail) assignment problem. One of the earlier formulations of the tail assignment problem is illustrated in [31]. The authors in this paper consider the assignment of aircraft to flights without maintenance planning implications for the man-power capacity planning. Additional work

by [34] integrate the flight-crew planning process with aircraft maintenance routing to derive synergies in the planning process.

In [88] the authors investigate maintenance planning by assigning aircraft to flight legs so as to minimize the cost of the maintenance that is performed. The proposed solution approach uses a heuristic that is computationally fast. On the other hand, in [101] the authors address the problem of manpower supply planning. Aircraft are scheduled to receive their respective A- and B-checks given the manpower that is available at certain maintenance stations in the network. The authors solve their problem using a mixed-integer programming approach.

Applying a Lagrangian relaxation solution approach to the tail assignment problem is explored in [36]. Here, the authors use a Lagrangian relaxation approach to determine the assignment of aircraft to routes with the objective of maximizing profit. The authors note that the Lagrangian approach is effective at providing a bound on the optimal solution and when paired with heuristic approaches provides a good solution to the assignment problem. We also apply a Lagrangian approach in our efforts to solve *MRP*.

Airline recovery models have received more recent attention. In [43], the authors pose a holistic airline schedule recovery framework by which various operational constraints are taken into consideration. Here, the authors pre-process a set of recovery networks that can be implemented as over-the-day disruptions take place. The authors note that the effectiveness of this approach is governed by the flights/planes ratio, an indicator of the possible number of assignments and thus a surrogate for the size of the optimization problem. In their work, [43], the authors focus on a total recovery solution including passengers and crew rotations. In our approach we

focus strictly on the tail assignment and its maintenance implications, while placing emphasis on solution speed and quality of maintenance coverage. That is, we only consider aircraft maintenance implications and thus can build a recovery plan quickly. In our approach, we operate at the lines-of-flight level and are able to solve longer horizon models because crew and thru-connections are not changed. To achieve this, we implement decompositions approaches including Lagrangian relaxation and column-generation.

5.4 Maintenance Recovery Problem

In the Maintenance Recovery Problem (MRP), we assume that the tail assignment has been disrupted. We further assume that the assignment of rotations to tails has been recovered, i.e. each aircraft now has a new rotation. These rotations are unlikely to be compatible with the original maintenance plan, because the counters of the aircraft, as well as the locations where the aircraft will overnight between lines-of-flight, have potentially changed. This in turn impacts the specific days and locations where maintenance is possible. Given these disruptions to the original maintenance plan, the goal of MRP is to create a new maintenance plan in which as many of the required maintenance events as possible are assigned.

As noted earlier, we emphasize that in this version of maintenance recovery we only schedule the first instance of each check type. Furthermore, given the current set of assigned rotations, complete maintenance coverage is not guaranteed. That is, each aircraft is assigned to a single rotation that may prevent it from overnighting at a maintenance station by the required deadline. We motivate an extension to this problem in which we seek to further increase the flexibility of the flight network to cover additional events by allowing for overnight swaps in §5.6.

Upon realization of changes to the assigned rotations due to disruptions on the day-of-operations, we can solve MRP. We first begin by identifying all the required maintenance events for each of the tails in the fleet. Each tail has an associated set of maintenance events, each which is a realization of a specific maintenance check based on the current state of its maintenance counters and its assigned rotation. Given this information we can determine a maintenance event a priori and the corresponding deadline by when it must be completed.

Each maintenance event features a set of possible opportunities to which it can be assigned. A maintenance opportunity is a tail-specific portion of time between two connecting lines-of-flight (generally overnight) at a specific station (airport) in the flight network. In addition, if that station has the capability to perform a particular maintenance check type, there is adequate ground time for the aircraft between two LOFs and the opportunity occurs on or before the deadline of the maintenance event, then this opportunity is a candidate for assignment to the corresponding maintenance event.

By first determining both the set of tail-specific maintenance events and opportunities, we can then formulate the MRP with which we optimize the assignment of maintenance events to maintenance opportunities.

5.4.1 Recovery Model Formulation

We present a formulation in which the primary decision variable represents the choice to assign a maintenance event to a given compatible maintenance opportunity, so as to maximize the total number of maintenance events that are assigned. In other words, we aim to perform as many of the required maintenance events as

possible, given the realized set of next instances of specific maintenance checks and opportunities for the recovered aircraft routings.

The following notation is required for this formulation of the Maintenance Recovery Problem.

Sets

E	The set of maintenance events.
O	The set of maintenance opportunities.
$O(e) \subseteq O$	The set of maintenance opportunities able to accommodate maintenance events e, $\forall e \in E$.
M	The set of maintenance stations.
D	The set of days in the planning horizon.

Parameters

γ_e	The required capacity (man-hours) of maintenance event e, $\forall e \in E$.
ρ_{md}	The available capacity (man-hours) at station m on day d of the planning horizon, $\forall m \in M, \forall d \in D$.
$d(o) \in D$	The specific day of the planning horizon of maintenance opportunity o, $\forall o \in O$.
$s(o) \in M$	The specific station at which maintenance opportunity o takes place, $\forall o \in O$.

Variables

x_{eo}	A binary variable that is 1 if maintenance event e is assigned to maintenance opportunity o and is 0 otherwise, $\forall e \in E, \forall o \in O(e)$.

Objective:

$$\max \sum_{e \in E} \sum_{o \in O(e)} x_{eo} \tag{5.1}$$

Subject to:

$$\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d \\ s(o)=m}} \gamma_e x_{eo} \leq \rho_{md} \qquad \forall m \in M, \forall d \in D \tag{5.2}$$

$$\sum_{o \in O(e)} x_{eo} \leq 1 \qquad \forall e \in E \tag{5.3}$$

$$x_{eo} \in \{0, 1\} \qquad \forall e \in E, \forall o \in O(e) \tag{5.4}$$

Objective (5.1) maximizes the total number of maintenance events assigned. Constraint (5.2) ensures that the collective man-hour requirements of the assigned maintenance events cannot exceed the station's capacity. Constraint (5.3) ensures that each maintenance event can be assigned to at most one compatible maintenance opportunity. Finally, constraint (5.4) requires variable integrality.

To solve this model we use the 32-bit version of the CPLEX v12.0 C++ API solver, running on an Intel Quad-Core, 2.8GHz processor, and a total of 8GB of main-memory. We tested our models on three data sets, the details of which are shown in Table (5.1). As noted in Table (5.2), the mathematical model is solved quickly, on the order of 8 seconds for the longest planning horizon. Furthermore, given the underlying network structure of our data, we are able to cover more than 62% of the required maintenance events. At this point, we assume that the operations controller will break the flight plan such that the remaining 38% of maintenance events can also be covered. In §5.6.3 we provide a detailed comparison between the results of the MRP and another recovery problem in which we create additional network flexibility by allowing overnight swaps in the rotation assignment.

So far, our maintenance recovery problem has focused only on the aircraft and its respective checks. While ensuring the completion of all required maintenance checks

Data Set	Days in Horizon	Flight Count	Tail Count	Maintenance Events
1	7	2,543	71	101
2	14	5,063	71	155
3	21	7,568	71	159

Table 5.1: Input data sets used for computational performance

is of utmost importance, maintenance events also occupy the stations through which aircraft rotate. While the primary objective of maintenance recovery model presented in this chapter is to ensure maintenance event coverage, secondary objectives based on a particular airline's business processes are explored in the next section.

Data Set	Days	Events Covered	Runtime (sec.)
1	7	63	0
2	14	112	2
3	21	115	8

Table 5.2: Computational run-time results for model

Next, we explore two additional objective functions in a secondary optimization for the maintenance recovery problem. First, in addition to maintenance event coverage, an airline may try to delay maintenance requirements as far into the future as possible. That is, if there is a choice between performing a maintenance event today or tomorrow, the maintenance event tomorrow is preferred as this (insofar maintenance feasible) uses more of the available maintenance counters. In other words, performing a maintenance event early implies unnecessary use of maintenance resources. In addition, pushing events further into the future features ripple-like effects, such that other events may also be pushed further into the future.

Another objective of interest to airlines is the utilization of maintenance stations. When planning for maintenance capacity, a uniform maintenance plan is preferred.

That is, ideally, maintenance events are spread evenly across the days of the planning horizon at a particular station. We explore these secondary objectives in the following section.

5.5 Secondary Objective Functions for the MRP

5.5.1 Minimizing Maintenance Event Earliness

In the first-stage optimization model, the goal was to maximize the number of maintenance events that are assigned over a given planning horizon. In this section, we explore a secondary objective: *minimizing the earliness* of the maintenance events, i.e. we wait as long as possible before performing an event. From an airline perspective, performing maintenance is an expensive undertaking. Performing maintenance early implies that time left on the maintenance counters goes unused and thus represents lost value. As such, maintenance events should be performed only when necessary.

Defining Maintenance Event Earliness

In the MRP, each maintenance event features a deadline by which it must be completed. Consider the example that is shown in Figure (5.5). This figure depicts a solution to the first-stage optimization problem, showing the set of events scheduled and how many days early (with respect to its deadline) each respective event will be completed.

We may be able to improve the maintenance assignment by moving events such that we minimize the number of days a maintenance event is performed early. For example, it might be possible to move the day 1 maintenance event that is currently scheduled to be performed one day early, to day 2 instead. To perform such a

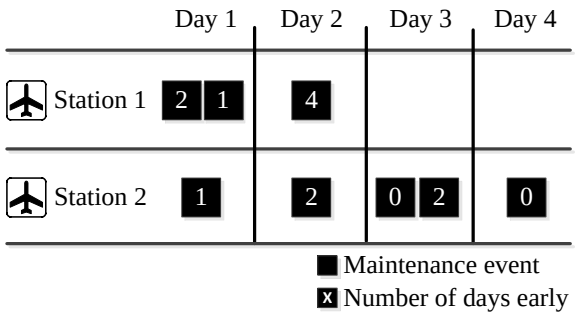


Figure 5.5: Maintenance allocation earliness from a first-stage optimization model.

reallocation, the station that the corresponding tail overnights at on day 2 must feature an available maintenance opportunity, including the ability to handle the specific check-type, as well as adequate man-power capacity. A new maintenance assignment, in which the total number of days early is minimized, is shown in Figure (5.6).

It should be noted that from a robustness perspective scheduling maintenance events right up until the respective deadline leaves little flexibility for additional disruptions. As such, from an airline perspective, when pre-computing the maintenance opportunities, it may be desirable to add a buffer to increase the total amount of flexibility and thus artificially decrease the actual deadline.

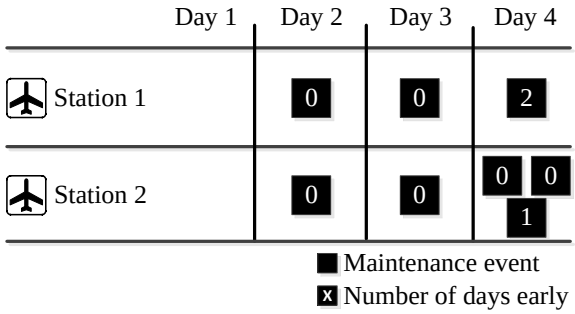


Figure 5.6: Maintenance allocation earliness from a second-stage optimization model.

Minimizing Earliness Model Formulation

The primary goal of the maintenance recovery problem remains to cover the maximum number of events possible over the course of the recovery horizon. As such, we consider the objective of minimizing the total earliness as a secondary objective. We alter the objective function in the second-stage optimization model, but require the same number of maintenance events to be performed as determined during the first-stage optimization from §5.4. More specifically, during the first-stage we record the total events covered as a parameter C. We then use this parameter in the form of a constraint. Mathematically, the parameter C is defined by equation (5.5), which is the objective function from the first-stage optimization model.

$$C = \sum_{e \in E} \sum_{o \in O(e)} x^*_{eo} \tag{5.5}$$

Given a solution to the maintenance recovery problem that maximizes the number of events covered (our primary concern), we can then formulate and solve a secondary optimization problem to find, within the set of optimal solutions to the primary optimization problem, a solution that minimizes the total earliness.

New Parameters

C	The number of maintenance events covered during the first-stage optimization.
$l(e) \in D$	The deadline for maintenance event e, $\forall e \in E$.

Objective:

$$\min \sum_{e \in E} \sum_{o \in O(e)} (l(e) - d(o))\, x_{eo} \tag{5.6}$$

Subject to:

$$\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d \\ s(o)=s}} \gamma_e x_{eo} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \tag{5.7}$$

$$\sum_{o \in O(e)} x_{eo} \leq 1 \qquad \forall e \in E \tag{5.8}$$

$$\sum_{e \in E} \sum_{o \in O(e)} x_{eo} \geq C \tag{5.9}$$

$$x_{eo} \in \{0, 1\} \qquad \forall e \in E, \forall o \in O(e) \tag{5.10}$$

The objective function in equation (5.6) minimizes the total earliness of all maintenance events, where earliness is defined by subtracting the scheduled day of the event from the deadline of the event. Constraint set (5.9) ensures that the total number of maintenance events from the first-stage optimization are met during this secondary optimization. Finally, constraint (5.10) requires variable integrality.

Minimizing Earliness Computational Results

In our second set of computational experiments, we use the solutions from the first stage optimization model as input to the second stage. Table (5.3) summarizes these results. Here, "Min Earliness" refers to the minimum earliness, i.e. the smallest difference between the deadline of any of the events and the day that respective event is scheduled, comparing the solutions found in the MRP and the MRP with the secondary objective. Analogously, "Max Earliness" refers to the maximum of

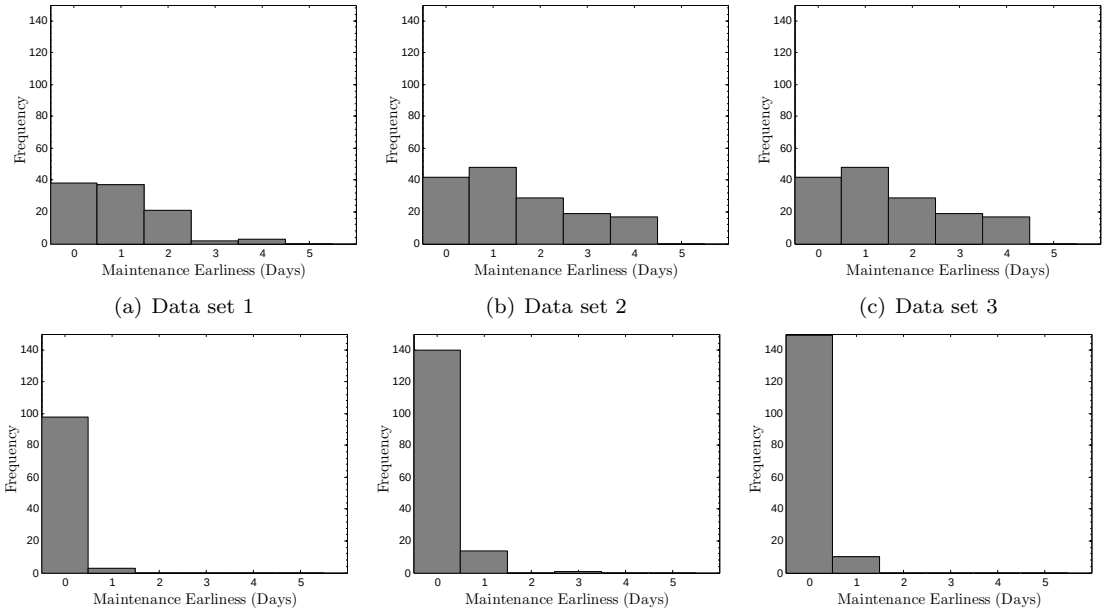
this difference. We note that the MRP with the secondary objective significantly decreases the maximum earliness compared to the solution to MRP. In addition, "Total Earliness" refers to the summed difference between the event deadline and the day the event is scheduled over all assignments in the respective problem. As noted in Table (5.3), this value is significantly lower under the MRP with the secondary objective.

	MRP			Minimizing Earliness		
	Data Set 1	Data Set 2	Data Set 3	Data Set 1	Data Set 2	Data Set 3
Min Earliness	0	0	0	0	0	0
Max Earliness	4	4	4	1	3	1
Total Earliness	97	118	124	3	13	3

Table 5.3: Summary statistics when minimizing total earliness

We can compare these solutions graphically. Figures (5.7(a)), (5.7(b)) and (5.7(c)) illustrate the number of days maintenance events are performed early for each data set. Contrasting these results to those from the optimization model in Figures (5.7(d)), (5.7(e)) and (5.7(f)), we note the significant improvement that can be achieved with this second-stage process. Most maintenance events can be re-allocated such that the total earliness (optimization objective function) for a given plan is decreased to three days for data set 1 and data set 3, and thirteen days for data set 2.

Based on the results obtained in this section, we conclude that a secondary optimization with the alternate objective of minimizing maintenance earliness has significant potential. Given the flexibility of the flight network, a function of the underlying input data, we are able to accommodate most maintenance events on their respective



(a) Data set 1 (b) Data set 2 (c) Data set 3

(d) Data set 1 (e) Data set 2 (f) Data set 3

Figure 5.7: Maintenance earliness objective using optimization model.

deadline and thus ensure that maintenance events are performed, without wasting costly resources.

5.5.2 Even Maintenance Event Distribution

An additional objective function that is of interest to airlines is a balanced utilization of maintenance stations capacity throughout the days of the planning horizon. We measure this capacity by the man-hours required at each station. Similar to the maintenance earliness metric described in §5.5.1, this objective relates to the total cost incurred by the airline as a function of the maintenance staff that must be hired to perform the scheduled work. Given that maintenance workers are employed on a regular schedule, smooth utilization on a day-to-day basis implies that adequate man-power capacity planning can be performed and any overtime compensation costs are minimized.

Defining Even Maintenance Allocation

We consider a sample allocation shown in Figure (5.8) below. As before, in this example, we have solved the maintenance recovery problem to cover as many events as possible. However, if the maintenance events can be reallocated, a more desirable solution may be achievable, as shown in Figure (5.9). In this case, the maintenance events are more evenly distributed across stations one and two, which from the perspective of an airline can be much more desirable as it allows for simpler workforce planning and reduced worker compensation.

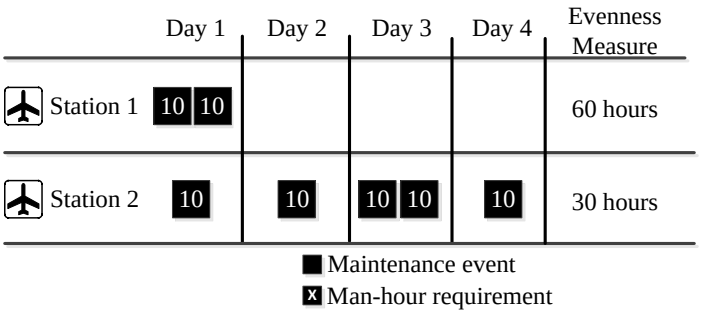


Figure 5.8: Maintenance allocation from MRP.

We can numerically evaluate the allocation of a schedule with respect to the even allocation of maintenance events by analyzing the difference in maintenance work-hour assignments across days in the planning horizon at each of the stations in the flight network. We will refer to this metric as the *evenness measure*, referring to the absolute difference between maintenance man-hours assigned to a station on a set of days during the recovery horizon.

Even Maintenance Event Distribution Formulation

In this formulation, we maximize the evenness of a set of maintenance events at a particular station. We define $t_{md_1d_2}$ as the absolute-value of the *difference* in the

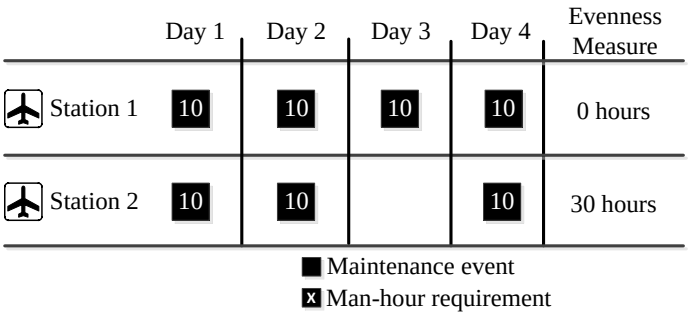


Figure 5.9: Maintenance allocation from an even distribution assignment.

number of maintenance man-hours required at station m between recovery days d_1 and d_2. Our objective is to minimize the total sum of this variable, over all stations and pairs of days. As in the previously model, we will require that the solution covers the same number of maintenance events as determined previously by the parameter C.

Objective:

$$\min \sum_{m \in M} \sum_{d_1 \in D} \sum_{\substack{d_2 \in D: \\ d_2 > d_1}} t_{m d_1 d_2} \tag{5.11}$$

Subject to:

$$\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d \\ s(o)=s}} \gamma_e x_{eo} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \tag{5.12}$$

$$\sum_{o \in O(e)} x_{eo} \leq 1 \qquad \forall e \in E \tag{5.13}$$

$$\sum_{e \in E} \sum_{o \in O(e)} x_{eo} \geq C \tag{5.14}$$

$$\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_1 \\ s(o)=m}} \gamma_e x_{eo} - \sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_2 \\ s(o)=m}} \gamma_e x_{eo} \leq t_{md_1d_2} \tag{5.15}$$ [1]

(5.16)

$$-\left(\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_1 \\ s(o)=m}} \gamma_e x_{eo} - \sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_2 \\ s(o)=m}} \gamma_e x_{eo} \right) \leq t_{md_1d_2} \tag{5.16}$$ [2]

$$t_{md_1d_2} \geq 0 \qquad \forall m \in M, \forall d_1 \in D, \forall d_2 \in D \tag{5.17}$$

$$x_{eo} \in \{0, 1\} \qquad \forall e \in E, \forall o \in O(e) \tag{5.18}$$

[1] $\forall m \in M, \forall d_1 \in D, \forall d_2 \in D : d_1 < d_2$

[2] $\forall m \in M, \forall d_1 \in D, \forall d_2 \in D : d_1 < d_2$

The new objective function is shown in equation (5.11). Constraint sets (5.15) and (5.16), in combination, force the variable ($t_{md_1d_2}$) to take on the positive difference between the total number of man-hours assigned to days d_1 and d_2 in the planning horizon for maintenance station m.

Difficulty of Solving Level Maintenance Allocation Problem

Unlike the first two models we have presented, both which solve quickly, there is strong incentive in our third model for variable fractionality, leading to expansive branch & bound trees and overall tractability challenges. The following example helps demonstrate this issue. Suppose we have a single maintenance event that has four possible maintenance opportunities, comprised of two stations on two days of the planning horizon, to which it could be assigned. During the first pass, the LP relaxation is solved. This assignment is seen in the Table (5.4).

Station	Day	Opportunity	Variable Assigned
1	1	1	$x_{11} = 0.25$
1	2	2	$x_{12} = 0.25$
2	1	3	$x_{13} = 0.25$
2	2	4	$x_{14} = 0.25$

Table 5.4: Solving the LP-relaxation, first pass

To create an even allocation, the optimizer assigns an equal allocation of the maintenance event to each of the stations on each of the days during the recovery horizon. As a result, the objective function takes a value of 0 since there is no difference between the allocation at each station. Following the solution to the LP relaxation, branching on each of the variables is possible. Suppose the first maintenance opportunity (station one, day one) is chosen as the branching variable. As such, this variable is fixed to 0 and the LP relaxation is solved again. Because we do not optimize across stations, in the solution to the next LP-relaxation, the variable assignment will split the event at station 2 as seen in Table (5.5).

Now suppose that branch & bound picks opportunity 3 as the next variable to branch on and sets this variable to value 0. This effectively leaves two variables that

Station	Day	Opportunity	Variable Assigned
1	1	1	$x_{11} = 0$ (branch variable)
1	2	2	$x_{12} = 0$
2	1	3	$x_{13} = 0.5$
2	2	4	$x_{14} = 0.5$

Table 5.5: Solving the LP-relaxation, second pass

must sum to 1. In addition, the difference between each of these two variables and 0 forms the objective function. As such, any assignment between 0 and 1 is possible which results in the same objective function value. Arguably, once a variable is fixed to 1, the branching tree is significantly decreased, especially in this case where only one event must be assigned. It turns out however that the branch & bound strategy cannot easily bound any particular part of the branching tree, simply because the assignment of an event to a station can iterate between many possible assignments, as seen in the example above. Furthermore in our findings, the branch & bound strategy does not converge within an acceptable amount of time.

The fact that this exhaustive exploration of the search tree takes place is the underlying reason as to why the branch & bound algorithm requires a significant amount of time and memory to compute the optimal solution to this problem. In our computational experiments, using data sets of various sizes, the branch & bound tree quickly consumed the total amount of system memory available and thus no optimal solution could be determined using a standard MIP-solver approach within our three hour time limit.

A Lagrangian Approach

A common approach to solving large mixed-integer problems is through a decomposition approach. To overcome the computational challenges that we have observed while solving the even maintenance assignment model, we take advantage of the fact

that, without the absolute-value constraints, the problem is easy to solve. As such, we apply a Lagrangian relaxation to this problem as described in [49].

The Lagrangian relaxation is a maximization problem. That is, we seek to find the values of λ that will result in the maximum minimization of the Lagrangian relaxation function $\mathrm{L}(\lambda)$, as shown in Equation (5.19). By definition, this function is a relaxation of the true objective, because two constraints are removed from the LP. In our case, we are solving a minimization problem and thus the Lagrangian relaxation will result in an underestimate of the true objective function value. As such, the goal of the Lagrangian relaxation is to find the maximum underestimate, which provides the best-possible lower bound on the actual objective function value. In certain cases, it may turn out that the Lagrangian relaxation may actually provide the optimal solution. This occurs when the optimal solution to the Lagrangian relaxation actually forms the true objective function of the underlying optimization problem. It turns out that the even maintenance allocation problem is indeed one of these special problems. This fact is proven in the appendix.

We now explore how the Lagrangian relaxation approach can be formulated with the even maintenance capacity allocation problem. In our approach, we use the load balancing problem and relax the constraints on the actual counting variable $t_{md_1d_2}$. As such, our relaxed problem is shown in equation (5.19) below.

Variables

$h^+_{md_1d_2}$ A continuous variable that represents the *positive* maintenance workload difference at station m between day d_1 and day d_2, $\forall m \in M, \forall d_1 \in D, \forall d_2 \in D$.

$h^-_{md_1d_2}$ A continuous variable that represents the *negative* maintenance workload difference at station m between day d_1 and day d_2, $\forall m \in M, \forall d_1 \in D, \forall d_2 \in D$.

Lagrangian Multipliers

$\lambda_{md_1d_2}$ The Lagrangian penalty multiplier on the difference of the capacity assignment constraint.

Objective:

$$\max_{\lambda_{md_1d_2}} \min \sum_{m \in M} \sum_{d_1 \in D} \sum_{d_2 \in D} t_{md_1d_2} - \lambda_{md_1d_2}[t_{md_1d_2} - (h^+_{md_1d_2} + h^-_{md_1d_2})] \tag{5.19}$$

Subject to:

$$\sum_{e \in E} \sum_{\substack{b \in O(e): \\ d(o)=d \\ s(o)=s}} \gamma_e x_{eo} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \tag{5.20}$$

$$\sum_{o \in O(e)} x_{eo} \leq 1 \qquad \forall e \in E \tag{5.21}$$

$$\sum_{e \in E} \sum_{o \in O(e)} x_{eo} \geq C \tag{5.22}$$

$$\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_1 \\ s(o)=m}} \gamma_e x_{eo} - \sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_2 \\ s(o)=m}} \gamma_e x_{eo} \leq t_{md_1d_2} \tag{5.23}$$ [1]

$$\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_1 \\ s(o)=m}} \gamma_e x_{eo} - \sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_2 \\ s(o)=m}} \gamma_e x_{eo} = (h^+_{md_1d_2} - h^-_{md_1d_2}) \tag{5.24}$$ [2]

$$h^+_{md_1d_2}, h^-_{md_1d_2} \geq 0 \tag{5.25}$$ [3]

$$x_{eo} \in \{0, 1\} \qquad \forall e \in E, \forall o \in O(e) \tag{5.26}$$

[1] $\forall m \in M, \forall d_1 \in D, \forall d_2 \in D : d_1 < d_2$

[2] $\forall m \in M, \forall d_1 \in D, \forall d_2 \in D : d_1 < d_2$

[3] $\forall m \in M, d_1 \in D, d_2 \in D$

Solving the Lagrangian Relaxation

For any given value λ, the Lagrangian relaxation provides a *lower* bound on the objective function of the problem that is solved. As mentioned earlier, the Lagrangian relaxation for this particular problem, at optimality, forms the true objective function. From the appendix, we can re-write the objective function as shown in equation (5.27). The objective represents the absolute difference in terms of mainte-

nance workload assigned at each station on two different days, which is the original objective.

$$\max_{\lambda_{md_1d_2}} \mathcal{L}(\lambda_{md_1d_2}) \tag{5.27}$$

where

$$\mathcal{L} = \min \sum \lambda_{md_1d_2}(h^+_{md_1d_2} + h^-_{md_1d_2}) \tag{5.28}$$

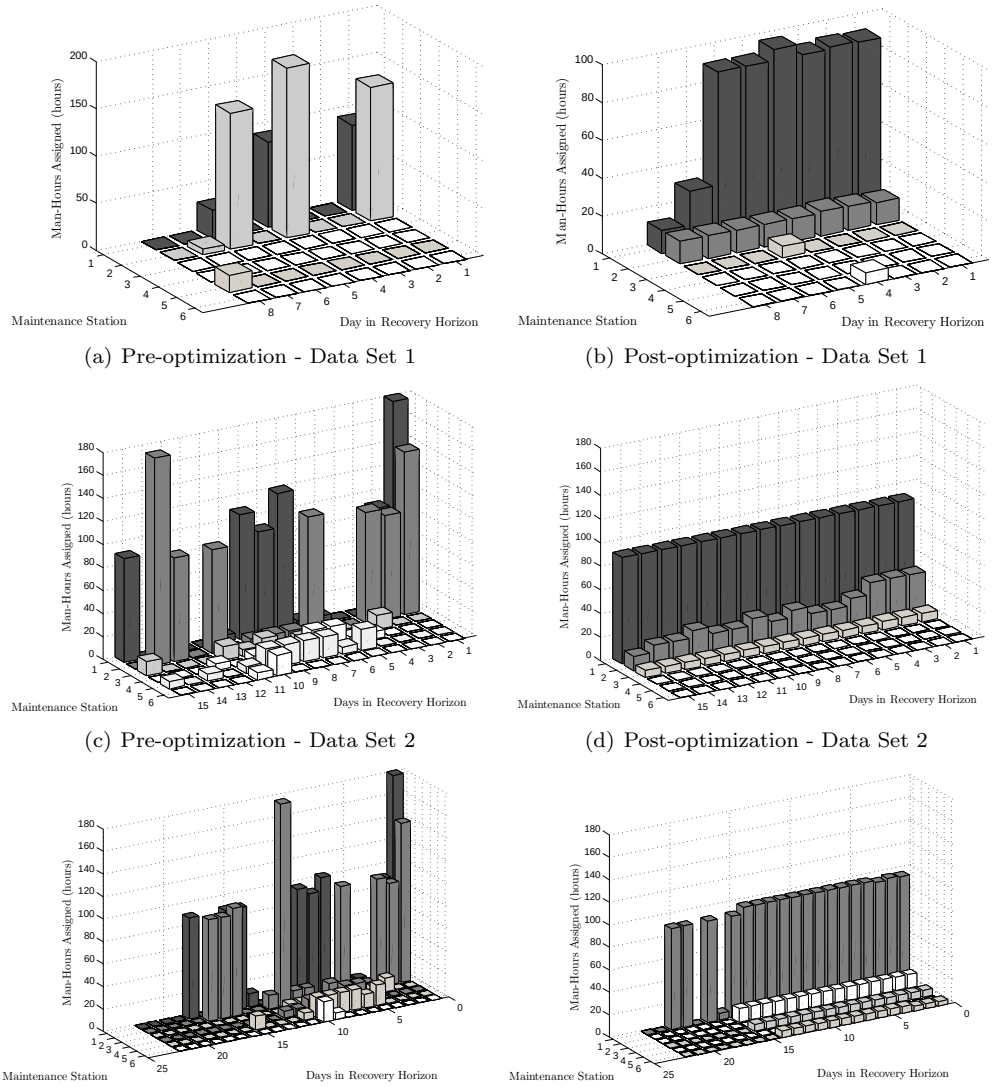
When solving the Lagrangian relaxation function $\mathcal{L}(\lambda)$, we need to find the λ values that maximize the objective function. The problem of maximizing the Lagrangian relaxation is a non-linear optimization problem. More specifically, the Lagrangian relaxation forms a piece-wise convex function as noted in [49]. Many algorithms can be employed to solve the Lagrangian relaxation. In our approach, we use a standard sub-gradient method to solve the Lagrangian relaxation also outlined in [49].

Even Maintenance Event Distribution Computational Results

Using the Lagrangian relaxation solution approach, we optimize the maintenance allocation by minimizing the overall difference between days in the recovery horizon at each of the stations in the network. The results of this approach are based on the data sets presented in the previous section. These results are shown graphically in Figures (5.10). Here, we illustrate the impact of our solution approach when managing the capacity allocation across the stations on each day of the planning horizon.

As noted in these figures, the x-axis represents a particular station in the flight network (1 through 6). On the y-axis, we indicate the particular day in the planning horizon. This planning horizon depends on the data set used for analysis and thus may vary from 1 through 7, 14 or 21 days. Finally, the z-axis indicates the amount of maintenance man-hours required at a particular station. As evident from these graphs, we are able to balance the maintenance requirements across various stations in the flight network ultimately resulting in better planning and forecasting of the maintenance crews required at the particular maintenance stations at various days of the planning horizon.

A (possibly desirable) side-effect of this approach may be that maintenance is concentrated at a particular station. If a maintenance event is assigned to a particular station, the optimization process will attempt to find other events to assign to this station during the remainder of the planning horizon, as this will benefit the objective function. This will cause, insofar possible, maintenance events to be moved from one station to another to help balance the overall assignment. This effect is also evident from Figures (5.10). Given this result, maintenance planners may be required to shift man-power capacity within the set of stations to accommodate the shift in capacity requirement.



(a) Pre-optimization - Data Set 1

(b) Post-optimization - Data Set 1

(c) Pre-optimization - Data Set 2

(d) Post-optimization - Data Set 2

(e) Pre-optimization - Data Set 3

(f) Post-optimization - Data Set 3

Figure 5.10: Maintenance capacity allocation objective pre- and post-optimization.

5.6 Maintenance Recovery Problem with Overnight Swaps

5.6.1 Overview of Overnight Swaps

Allowing changes to the underlying rotations (and not just the assignment of tails to the original rotations), can significantly increase flexibility for maintenance events that must be assigned. Compared to MRP, we can now direct aircraft to a maintenance station as necessary. However, changing a rotation by performing over-the-day swaps can have severe operational impacts, including broken thru-connections and

altered crew assignments.

Therefore, to create flexibility with minimal impact on other operational processes, we restrict our modifications to the original rotations in the following way. First, we only recombine the original lines-of-flight to form new rotations, not making any changes (i.e. over-the-day swaps) to the LOFs themselves. That is, we consider lines-of-flight to be indivisible blocks that must remain intact. Second, we only allow changes to occur within the same sub-fleet type. That is, two lines-of-flight may only be connected if they are both scheduled to be flown by the same type of aircraft. A fleet-family restriction eliminates issues with the crew's ability to fly such particular equipment, as well as passenger-capacity and seat assignment issues.

Benefits of Overnight Swaps

Allowing overnight swaps significantly increases the flexibility of the flight network and thus creates additional maintenance opportunities that did not exist previously. For example, suppose the current state of the system features two aircraft at station DTW at the beginning of day 1. Tail #1 requires maintenance at the end of day 2, while Tail #2 requires maintenance at the end of day 1. This situation is depicted graphically in Figure (5.11). Given the current assignment, only Tail #1 will end at a maintenance station at the end of day 1. That is, we can complete the required maintenance event of Tail #1, albeit a day early. For Tail #2, under the proposed schedule, maintenance coverage is not possible.

However, if we were to swap the assignment, exchanging the assignments of Tail #1 and Tail #2, both aircraft would receive their respective maintenance checks. That is, Tail #2 now terminates in BOS at the end of the first day and is able

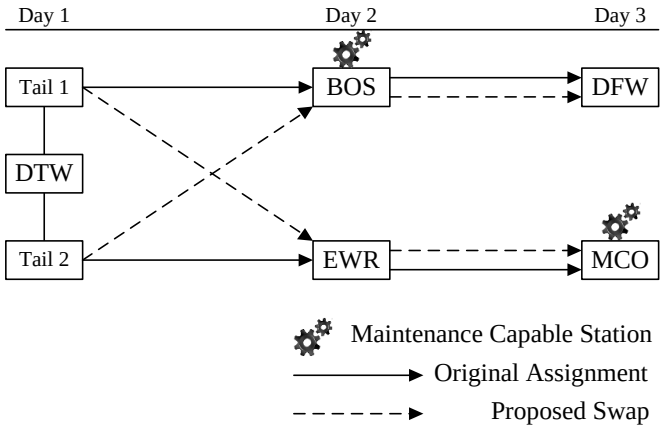


Figure 5.11: Benefit of changing the tail assignment

to receive its required check. In addition, Tail #1 terminates at MCO at the end of day 2 and can receive its required check on the more desirable second day. By allowing such overnight swaps to occur, we are able to create additional flexibility in the network, which will allow for additional maintenance event coverage that was not possible with the formulation presented under the Maintenance Recovery Problem (MRP).

Impact on Maintenance Opportunities

Allowing overnight swaps creates additional flexibility by altering the set of possible opportunities at which maintenance events can be completed. In the MRP, presented in §5.4, maintenance opportunities were specified a priori. This approach was appropriate, because the tail assignment was fixed and thus all maintenance opportunities could be pre-processed. In this extension, the lines-of-flight that make up a rotation are a decision, i.e. we connect lines-of-flight to form rotations. Deciding the LOFs in a rotation implies that the overnight stops an aircraft makes at a particular station are also part of the decision variable. An overnight stop between two LOFs can potentially provide a maintenance opportunity if several conditions are met. As illustrated in Figure (5.12), these conditions include:

1. The two LOFs terminate and subsequently originate at the same station.
2. The station between two LOFs is maintenance capable of a specific maintenance check.
3. The inbound and outbound LOF allow for adequate time between arrival and departure to perform a maintenance check.
4. The maintenance shift (for the ground crew) overlaps with the time window between the two LOFs for a sufficient length of time to perform the check.
5. These maintenance opportunities may start once the maintenance shift has started or the aircraft has arrived at the station, whichever is later.

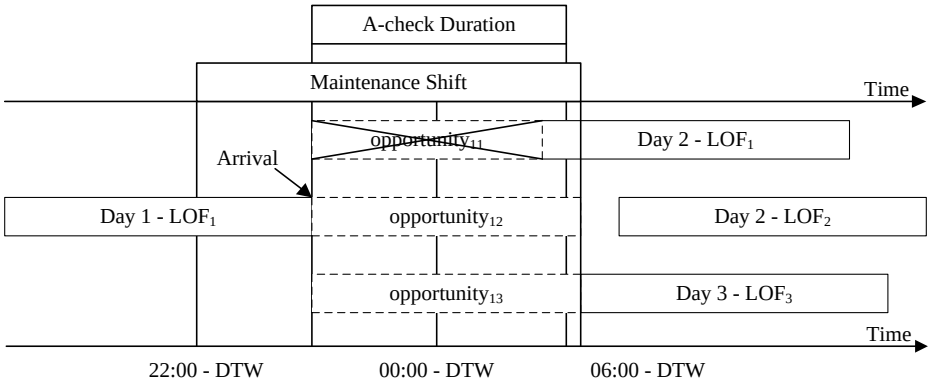


Figure 5.12: Maintenance block pre-processing

As shown in Figure (5.12), a LOF arrives shortly before midnight at station DTW. This LOF can be paired with three outgoing LOFs the following morning. As noted in the figure, there are a total of three possible opportunities, however, the first opportunity is not sufficiently long for an A-check. Thus only the other two opportunities can be used to perform an A-check.

In the MRP model maintenance opportunities were limited by the planned rotations; we now create additional opportunities by deciding which lines-of-flight to

connect. We will refer to this problem as the *Maintenance Recovery Problem with Overnight Swaps (MRP-OS)*

Constraining the Overnight Swap Choices

In the previous section, we presented our approach for identifying maintenance opportunities between LOFs, given that we allow changes to the underlying rotations through overnight swaps. One of the assumptions of this formulation is that changing the assignment of LOF has no implications on the maintenance events. As alluded to in §5.4, the deadlines for maintenance events are pre-processed based on a set of counters, including: flight hours, cycle count and calendar day frequency of the aircraft at the start of the recovery horizon and the impact on those counters from the assigned rotations.

Changing the LOFs for a particular tail will alter its required maintenance events and their respective deadlines. For example, suppose a maintenance event must be performed every 20 cycles. As such, any changes in the set of assigned LOFs that feature additional cycles as compared to the planned set of LOFs must be evaluated for feasibility. That is, if in the new rotation, we assign a line-of-flight that deviates from the original counters, we must ensure that such an assignment does not render the aircraft infeasible by violating any of the maintenance limits.

In this formulation, we address this issue by restricting which overnight swaps are allowed to take place. More specifically, a line-of-flight assignment that requires the aircraft to perform additional cycles or flight-hours by the deadline of the original maintenance event is constrained by the formulation to ensure maintenance feasibility relative to the original deadlines. For example, if the original maintenance event featured a deadline on day 3 of the planning horizon for an A-check, then the

only overnight swaps that are permitted are those that will not violate the A-check counters of the tail under consideration by day 3. This constraint artificially reduces the search space for all possible overnight swaps. We will relax this requirement in Chapter VI.

5.6.2 Formulating the MRP-OS

In this section, we provide several possible formulations that can be used to mathematically represent MRP-OS. As we explore each of these formulations, we emphasize trade-offs between each of these models.

We begin this section by presenting a *line-of-flight-based* formulation. In this approach, the primary decision holds the least amount of information, namely whether an aircraft is assigned a particular line-of-flight. The second approach consists of two formulations, both using a *connection-based* model in which the primary decision variable represents a connection between two lines-of-flight. Finally, we provide two *rotation-based* approaches in which we generate entire strings of LOFs, including our *augmented* rotation-based approach, where maintenance events are built directly into the rotation variable.

Line-of-Flight-based Formulation

The overnight-swap model can be formulated using a line-of-flight-based representation. The formulation, similar to that found in the fleet assignment literature as illustrated in [58], is applied to the maintenance recovery problem and is depicted graphically in Figure (5.13). Here, the nodes represent points-in-time at various stations in the network. The arcs connecting the nodes represent entire LOFs, which are made up of individual flights. As shown in the figure, at some point in time,

a LOF departs DTW, performs several flights ($DTW \rightarrow PHX \rightarrow LAS \rightarrow EWR$) and terminates at EWR at another point in time.

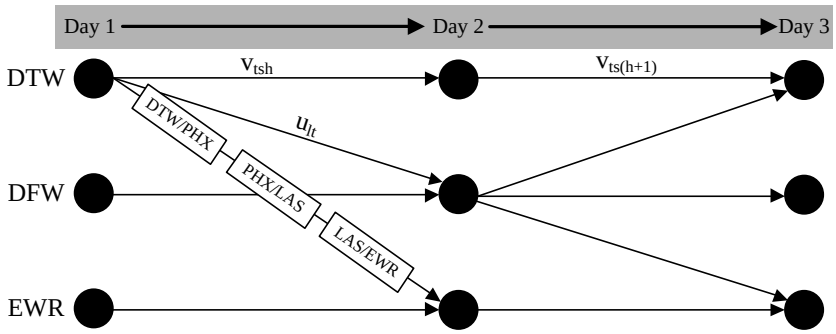


Figure 5.13: Network-based formulation for assignment of lines-of-flight to tails

In this formulation, the primary decision variable u_{lt} specifies whether line-of-flight l is assigned to tail t. In addition, we introduce a secondary variable w_{eds}, which represents the decision of whether maintenance event e is assigned to station s on day d of the planning horizon. In this formulation, the primary decision variable contains the least amount of information compared to the other formulations that will follow in this section. Here, the primary decision variable specifies only which line-of-flight is assigned to a particular tail. However, any other information, such as the formation of the flight network or the association of maintenance events to lines-of-flight must be accomplished through constraints. We define the following sets, parameters and variables for this particular formulation.

Sets

T	The set of tails.
L	The set of lines-of-flight.
$L(t) \subseteq L$	The set of lines-of-flight that can be flown by tail t, $\forall t \in T$.
D	The set of days in the planning horizon.
S	The set of stations in the network.
$M \subseteq S$	The set of maintenance stations in the network.
E	The set of all maintenance events.
$S(e)$	The set of stations at which maintenance event e can take place, $\forall e \in E$.
$D(e)$	The set of days on which maintenance event e can take place, $\forall d \in D$.
$E(t) \subseteq E$	The set of maintenance events for tail t, $\forall t \in T$.
$H(s)$	The set of sequentially ordered time nodes at station s, i.e. departures or arrival lines-of-flight, $\forall s \in S$.
$I(t,s,h) \subset L(t)$	The set of lines-of-flight that arrive (inbound) at station s at event h that can be flown by tail t, $\forall t \in T, \forall s \in S, \forall h \in H$.
$O(t,s,h) \subset L(t)$	The set of lines-of-flight that depart (outbound) from station s at event h that can be flown by tail t, $\forall t \in T, \forall s \in S, \forall h \in H$.

Parameters

$t_D(l)$	The minimum of the scheduled departure time and end-of-maintenance shift time of line l, $\forall l \in L$.
$t_A(l)$	The maximum of the scheduled arrival time or begin-of-maintenance shift time of line l, $\forall l \in L$.
$a(l) \in D$	The day of the planning horizon at which line-of-flight l terminates, $\forall l \in L$.
$l(e) \in D$	The deadline on which event e must be performed, $\forall t \in T, \forall e \in E(t)$.
$s_A(l) \in S$	The specific station at which line-of-flight l arrives, $\forall l \in L$.
$s_D(l) \in S$	The specific station at which line-of-flight l departs, $\forall l \in L$.
γ_e	The required capacity (man-hours) of maintenance event e, $\forall e \in E$.
δ_e	The required duration (man-hours) of maintenance event e, $\forall e \in E$.
ρ_{sd}	The available capacity (man-hours) at station s on day d of the planning horizon, $\forall s \in M, \forall d \in D$.
κ_{cl}	The contribution of line-of-flight l to maintenance counter c, $\forall l \in L$.
α_{ct}	The initial value for maintenance counter c on tail t, $\forall t \in T, \forall c \in C$.
δ_{cte}	The limit for maintenance counter c on tail t with its respective maintenance event e, $\forall c \in C, \forall t \in T, \forall e \in E(t)$.

Variables

w_{eds} a binary variable that is 1 if maintenance event e is performed on day d at station s, and 0 otherwise, $\forall t \in T, \forall e \in E(t), \forall d \in D(e), \forall s \in S(e)$.

u_{lt} A binary variable that is 1 if line-of-flight l is assigned to tail t and is 0 otherwise, $\forall t \in T, \forall l \in L(t)$.

v_{tsh} A binary variable that is 1 if tail t is on the ground at station s immediately preceding event h and is 0 otherwise, $\forall t \in T, \forall s \in S, \forall h \in H$.

Objective:

$$\max \sum_{t \in T} \sum_{e \in E(t)} \sum_{d \in D(e)} \sum_{s \in S(e)} w_{eds} \tag{5.29}$$

Subject to:

$$\sum_{t\in T}\sum_{\substack{e\in E(t):\\ d\in D(e)\\ s\in S(e)}}\gamma_e w_{eds}\le \rho_{sd} \qquad \forall s\in M,\forall d\in D \tag{5.30}$$

$$w_{eds}-\sum_{\substack{l\in L(t):\\ s_A(l)=s\\ a(l)=d}} u_{lt}\le 0 \tag{5.31}$$ [1]

$$\sum_{d\in D(e)}\sum_{s\in S(e)} w_{eds}\le 1 \qquad \forall t\in T,\forall e\in E(t) \tag{5.32}$$

$$\alpha_{ct}+\sum_{\substack{l\in L(t):\\ a(l)\le l(e)}}\kappa_{cl}u_{lt}\le \delta_{cte} \qquad \forall c\in C,\forall t\in T,\forall e\in E(t) \tag{5.33}$$

$$\sum_{\substack{l\in L(t):\\ s_A(l)=s\\ a(l)=d}} t_A(l)u_{lt}+\delta_e w_{eds}-\sum_{\substack{l\in L(t):\\ s_D(l)=s\\ a(l)=d+1}} t_D(l)u_{lt}\le 0 \tag{5.34}$$ [2]

$$v_{tsh}+\sum_{l\in I(t,s,h)} u_{lt}-v_{ts(h+1)}-\sum_{l\in O(t,s,h)} u_{lt}=0 \qquad \forall t\in T,\forall s\in S,\forall h\in H(s) \tag{5.35}$$

$$\sum_{t\in T:l\in L(t)} u_{lt}\le 1 \qquad \forall l\in L \tag{5.36}$$

$$w_{eds}\in\{0,1\} \tag{5.37}$$ [3]

$$u_{lt}\in\{0,1\} \qquad \forall t\in T,\forall l\in L(t) \tag{5.38}$$

$$v_{tsh}\in\{0,1\} \qquad \forall t\in T,\forall s\in S,\forall h\in H(s) \tag{5.39}$$

[1] $\forall t\in T,\forall e\in E(t),\forall d\in D,\forall s\in S$

[2] $\forall t\in T,\forall e\in E(t),\forall d\in D(e),\forall s\in S(e)$

[3] $\forall t\in T,\forall e\in E(t),\forall d\in D(e),\forall s\in S(e)$

Similar to the MRP formulation, in the objective in equation (5.29) we maximize the total number of maintenance events that are covered. Constraint (5.30) ensures

that the capacity of a maintenance station, measured in available man-hours, is not exceeded. This constraint could be modified to handle other forms of capacity constraints, such as the maximum number of aircraft serviced. In constraint (5.31), we ensure that a maintenance event is only assigned if a particular line l actually terminates at a particular maintenance opportunity that matches the maintenance event e. In other words, we can only assign maintenance event e if line l terminates at the particular day and station where maintenance event e is destined to take place. Constraint (5.32) ensures that each maintenance event is assigned at most once. Next in constraint (5.33) we ensure that only a set of lines-of-flight can be assigned that do not violate the original maintenance event's respective counters. In constraint (5.34) we ensure that a maintenance event e is only assigned if adequate time to perform maintenance exists. That is, the maintenance event must fit within the time window as determined by the difference between the maximum of the arrival of the last line-of-flight and the beginning of the maintenance shift, and the minimum of the departure time and the end of the maintenance shift. Constraint (5.35) ensures continuity within the network. More specifically, at each point in time, an aircraft is either on the ground already or arriving, while immediately following this event, it must take-off or remain on the ground. Such a constraint ensures that aircraft follow a logical path throughout the flight network. In constraint (5.36), we require that each line-of-flight is assigned at most once. Finally, constraints (5.37), (5.38), (5.39) provide integer constraints on each of the variables.

Based on the formulation presented, the line-of-flight-based approach contains little information in the primary decision variable u_{lt} and thus uses constraints to perform the maintenance event assignment, as well as the network formation. As such, this formulation contains a large number of constraints. In the next section,

we explore the connection-based approach. Here we store a connection between two-lines of flight in the primary decision variable, which allows us to remove constraints (5.34) and (5.35) from the line-of-flight-based formulation.

Connection-based Formulations

Perhaps the most intuitive model that can be used to represent the overnight swaps is the connection-based formulation similar to the approach presented by [1]. In this particular model, we define a binary variable x_{ijt} that represents the decision as to whether the connection between line-of-flight i and line-of-flight j is performed by tail t. This primary decision variable forms the basis for two separate yet equivalent formulations presented in this section.

It should be noted that the connection-based formulation stores additional information in the actual variable as compared to the network-based formulation presented previously. In this formulation, the connections between line-of-flight i and line-of-flight j is only possible if these lines can indeed be connected given the flight schedule. As such, network construction constraints are not necessary in this formulation, because an instance of the variable x_{ijt} is only created if such a connection is indeed feasible.

The connection-based model can be divided two different formulations. In these two variations, we represent the maintenance decision variable using two different approaches. We prove, however, that these two formulations are indeed equivalent in their objective in the appendix.

Overnight-Swap Connection-based Model - Decision Variables: x_{ijt}, b_{ije}

In this formulation, the decision of performing a maintenance event is similar to the

decision of connecting two LOFs using variable x_{ijt}. That is, we define a variable b_{ije} that represents the connection of line i with line j and maintenance event e which takes place during the overnight stop between the two lines. In addition to this variable, we require the following sets, parameters and variables for this particular formulation.

Sets

C	The set of maintenance counters, including flying-hours, cycles and calendar days.
$U_T(l,t) \subseteq L(t)$	The set of lines-of-flight upstream to line-of-flight l restricted for the fleet type of tail t, $\forall t \in T, \forall l \in L(t)$. In this case, we refer to upstream lines-of-flights as those lines that directly precede line l in terms of departure station and day in the planning horizon, e.g. $\forall l' \in L(t) : s_A(l') = s_D(l), a(l) = a(l') - 1$.
$D_T(l,t) \subseteq L(t)$	The set of lines-of-flight downstream from line-of-flight l restricted for the fleet type of tail t, $\forall t \in T, \forall l \in L(t)$. In this case, we refer to downstream lines-of-flights as those lines that directly follow line l in terms of arrival station and day in the planning horizon, e.g. $\forall l' \in L(t) : s_D(l') = s_A(l), a(l') = a(l) - 1$.
$U_E(l,e) \subseteq L(t)$	The set of lines-of-flight upstream to line-of-flight l that can accommodate maintenance event e, $\forall t \in T, \forall l \in L(t), \forall e \in E(t)$ and $\forall l' \in L(t) : s_A(l') = s_D(l)$.
$D_E(l,e) \subseteq L(t)$	The set of lines-of-flight downstream from line-of-flight l that can accommodate maintenance event e, $\forall t \in T, \forall l \in L(t), \forall e \in E(t)$ and $\forall l' \in L(t) : s_D(l') = s_A(l)$.

Parameters

$s_A(l) \in S$	The specific station at which line-of-flight l arrives, $\forall l \in L$.
$s_D(l) \in S$	The specific station at which line-of-flight l departs, $\forall l \in L$.

Variables

b_{ije}	a binary variable that is 1 if the maintenance opportunity between lines-of-flight i and j is assigned to maintenance event e, and 0 otherwise, $\forall t \in T, e \in E(t), s \in S(e), d \in D(e), i \in L(t) : a(i) = d, s_A(i) = s, j \in D(i,e)$.
x_{ijt}	a binary variable that is 1 if line i is followed by line j on tail t, $\forall t \in T, \forall i \in L(t), \forall j \in D(i,t)$.
y_{it}	a binary variable that is 1 if line i is the first line assigned to tail t during the planning horizon, $\forall t \in T, \forall i \in L(t)$.
z_{it}	a binary variable that is 1 if line i is the last line assigned to tail t during the planning horizon, $\forall t \in T, \forall i \in L(t)$.

Objective:

$$\max \sum_{t \in T} \sum_{e \in E(t)} \sum_{s \in S(e)} \sum_{d \in D(e)} \sum_{\substack{i \in L(t): \\ d(i)=d \\ s_A(i)=s}} \sum_{j \in D_E(i,e)} b_{ije} \tag{5.40}$$

Subject to:

$$\sum_{t \in T} \sum_{e \in E(t)} \sum_{s \in S(e)} \sum_{d \in D(e)} \sum_{\substack{i \in L(t): \\ a(i)=d \\ s(i)=s}} \sum_{j \in D_E(i,e)} \gamma_e b_{ije} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \tag{5.41}$$

$$b_{ije} - x_{ijt} \leq 0 \tag{5.42}$$

[1]

$$\alpha_{ct} + \sum_{\substack{i \in L(t): \\ a(i) \leq l(e)}} \kappa_{ci} y_{it} + \sum_{i \in L(t)} \sum_{\substack{j \in L(t): \\ a(j) \leq l(e)}} \kappa_{cj} x_{ijt} \leq \delta_{cte} \qquad \forall c \in C, \forall t \in T, \forall e \in E(t) \tag{5.43}$$

$$\sum_{s \in S(e)} \sum_{d \in D(e)} \sum_{\substack{i \in L(t): \\ a(i)=d \\ s(i)=s}} \sum_{j \in D_E(i,e)} b_{ije} \leq 1 \qquad \forall t \in T, \forall e \in E(t) \tag{5.44}$$

$$\sum_{j \in U_T(i,t)} x_{jit} + y_{it} - \sum_{j \in D_T(i,t)} x_{ijt} - z_{it} = 0 \qquad \forall t \in T, \forall i \in L(t) \tag{5.45}$$

$$\sum_{i \in L} y_{it} \leq 1 \qquad \forall t \in T \tag{5.46}$$

$$\sum_{t \in T} \left(y_{it} + \sum_{j \in U_T(i,t)} x_{jit} \right) = 1 \qquad \forall i \in L \tag{5.47}$$

$$b_{ije} \in \{0, 1\} \tag{5.48}$$

[2]

$$x_{ijt} \in \{0, 1\} \tag{5.49}$$

[3]

$$y_{it} \in \{0, 1\} \qquad \forall t \in T, \forall i \in L(t) \tag{5.50}$$

$$z_{it} \in \{0, 1\} \qquad \forall t \in T, \forall i \in L(t) \tag{5.51}$$

[1] $\forall t \in T, e \in E(t), s \in S(e), d \in D(e), i \in L(t) : a(i) = d, s_A(i) = s, j \in D(i, e)$

[2] $\forall t \in T, e \in E(t), s \in S(e), d \in D(e), i \in L(t) : a(i) = d, s_A(i) = s, j \in D(i, e)$

[3]$\forall t \in T, \forall i \in L(t), \forall j \in D(i,t)$

Constraint (5.41) ensures that the capacity of a maintenance station, measured in available man-hours, is not exceeded. In constraint (5.42), we ensure that a maintenance event is only assigned if a particular set of lines i, j are actually paired together to form a maintenance opportunity for maintenance event e. Next in constraint (5.43) we ensure that only a set of lines-of-flight can be assigned that do not violate the original maintenance event's respective counters. Constraint (5.44) ensures that each maintenance event is assigned at most once. Constraint (5.45) requires if a line-of-flight occurred on the first day of the planning horizon or a proceeding line-of-flight, then it is followed by another or it is the last in the planning horizon. We ensure that each tail can only have a single starting line-of-flight as defined in constraint (5.46). Constraint (5.47) requires each line-of-flight to be assigned exactly once. Finally, constraints (5.48), (5.49), (5.50) and (5.51) provide integer constraints on each of the variables.

In contrast to using the variable b_{ije} to represent the assignment of a maintenance event, we can use a more aggregate variable, similar to the approach in the line-of-flight-based formulation. Next, we present an identical formulation using this alternate variable representation. The equivalence between these models is proven in the appendix.

Overnight-Swap Connection-based Model - Decision Variables: x_{ijt}, w_{eds}

Instead of using the variable b_{ije} to represent the assignment of maintenance event e between line-of-flight i and line-of-flight j, we can reduce the overall number of variables by introducing a new variable w_{eds}. This binary variable represents the fact

that maintenance event e takes place on day d of the planning horizon at maintenance station s. Using this variable definition, we no longer require a total of $(i \times j)$ number of variables for each of the maintenance opportunities and their respective assignments, but can rather replace this with $(s \times d)$, the number of maintenance stations multiplied by the number of days in the planning horizon. This reduction also applies to constraint (5.42) in the previous model. Previously, we required a total of $(e \times i \times j)$ constraints of this type. This number is now reduced to $(e \times d \times s)$, while the remaining size of the formulation remains the same. The formulation using this type of variable definition is provided below.

Variables

w_{eds} a binary variable that is 1 if maintenance event e is performed on day d at station s, and 0 otherwise, $\forall t \in T, \forall e \in E(t), \forall d \in D(e), \forall s \in S(e)$.

Objective:

$$\max \sum_{t \in T} \sum_{e \in E(t)} \sum_{d \in D(e)} \sum_{s \in S(e)} w_{eds} \tag{5.52}$$

Subject to:

$$\sum_{t\in T}\sum_{\substack{e\in E(t):\\ d\in D(e)\\ s\in S(e)}}\gamma_e w_{eds}\le \rho_{sd} \qquad \forall s\in M,\forall d\in D \tag{5.53}$$

$$w_{eds}-\sum_{\substack{i\in L(t):\\ s_A(i)=s\\ a(i)=d}}\sum_{j\in D_E(i,e)} x_{ijt}\le 0 \qquad [1] \tag{5.54}$$

$$\sum_{s\in S(e)}\sum_{d\in D(e)} w_{eds}\le 1 \qquad \forall t\in T,\forall e\in E(t) \tag{5.55}$$

$$\alpha_{ct}+\sum_{\substack{i\in L(t):\\ a(i)\le l(e)}}\kappa_{ci}y_{it}+\sum_{i\in L(t)}\sum_{\substack{j\in L(t):\\ a(j)\le l(e)}}\kappa_{cj}x_{ijt}\le \delta_{cte} \qquad \forall c\in C,\forall t\in T,\forall e\in E(t) \tag{5.56}$$

$$\sum_{j\in U_T(i,t)} x_{jit}+y_{it}-\sum_{j\in D_T(i,t)} x_{ijt}-z_{it}=0 \qquad \forall t\in T,\forall i\in L(t) \tag{5.57}$$

$$\sum_{i\in L} y_{it}\le 1 \qquad \forall t\in T \tag{5.58}$$

$$\sum_{t\in T}\left(y_{it}+\sum_{j\in U_T(i,t)} x_{jit}\right)=1 \qquad \forall i\in L \tag{5.59}$$

$$w_{eds}\in\{0,1\} \qquad [2] \tag{5.60}$$

$$x_{ijt}\in\{0,1\} \qquad [3] \tag{5.61}$$

$$y_{it}\in\{0,1\} \qquad \forall t\in T,\forall i\in L(t) \tag{5.62}$$

$$z_{it}\in\{0,1\} \qquad \forall t\in T,\forall i\in L(t) \tag{5.63}$$

[1] $\forall t\in T,\forall e\in E(t),\forall d\in D(e),\forall s\in S(e)$

[2] $\forall t\in T,\forall e\in E(t),\forall d\in D(e),\forall s\in S(e)$

[3] $\forall t\in T,\forall i\in L(t),\forall j\in D(i,t)$

Constraint (5.53) ensures that the capacity of a maintenance station, measured in available man-hours, is not exceeded. In constraint (5.54), we ensure that a

maintenance event is only assigned if a particular line i actually terminates at a particular maintenance opportunity that matches the maintenance event e. It should be noted that within this constraint we ensure the compatibility between the event type e and the tail t that is assigned to the line. Constraint (5.55) ensures that each maintenance event is assigned at most once. Next in constraint (5.56) we require that only a set of lines-of-flight can be assigned that do not violate the original maintenance event's respective counters. The second set of constraints (5.57 - 5.59) are network constraints. That is, constraint (5.57) requires for each particular tail that if a line-of-flight was the first or a proceeding line-of-flight, then it is followed by another or it is the last in the planning horizon. We ensure that each tail can only have a single starting line-of-flight as defined in constraint (5.58). Constraint (5.59) requires each line-of-flight to be assigned exactly once. As before, constraint (5.60), (5.61), (5.62) and (5.63) provide integrality for each of the variables.

Rotation-based Formulations

The rotation-based approach is commonly found in many airline-related planning problems. For example, the authors in [34] use a rotation-based approach (commonly also known as a string-based approach) to generate crew rotations. In our approach, lines-of-flight are connected to form rotations that span several days of the planning horizon. In contrast to the connection-based approach, in this formulation, we capture additional information within a single variable.

Furthermore, we present two rotation-based formulations. The first formulation uses the x_{tr} variable to represent the assignment of a rotation to a tail, but is combined with the variable w_{eds} to indicate whether a maintenance assignment is performed on a particular day and at a particular station. This approach is similar to the

line-of-flight and connection-based formulations presented in the previous section. In addition, we present a rotation-based formulation in which the maintenance events are already contained within rotations (called *augmented rotations*), thus eliminating the need for the w_{eds} variable. Both formulations will be solved using a similar column-generation approach and the difference in the model size and performance is detailed in §5.6.3.

Rotation-based Model - Decision Variables: w_{eds}, x_{tr}

In this section we present the rotation-based approach using the w_{eds} supplemental variable. We define additional sets, parameters and variables in the table below.

Sets

R	The set of all rotations.
$R(t) \subseteq R$	The set of all rotations that can be assigned to tail t, $\forall t \in T$.

Parameters

κ_{reds}	A binary parameter that defines if line-of-flight r can be associated with event e on day d at station s, $\forall t \in T, \forall r \in R(t), \forall e \in E, \forall d \in D, \forall s \in S$.
δ_{lr}	A binary parameter that indicates if line-of-flight l is contained in rotation r, $\forall t \in T, \forall r \in R(t), \forall l \in L(t)$.

Variables

x_{tr}	a binary variable that is 1 if rotation r is assigned to tail t, $\forall t \in T, \forall r \in R(t)$.

Objective:

$$\max \sum_{e \in E} \sum_{d \in D} \sum_{s \in S} w_{eds} \tag{5.64}$$

Subject to:

$$\sum_{t \in T} \sum_{\substack{e \in E(t): \\ d \in D(e) \\ s \in S(e)}} \gamma_e w_{eds} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \tag{5.65}$$

$$w_{eds} - \sum_{t \in T} \sum_{r \in R(t)} \kappa_{resd} x_{tr} \leq 0 \qquad \forall e \in E, \forall s \in S, \forall d \in D \tag{5.66}$$

$$\sum_{t \in T} \sum_{r \in R(t)} \delta_{lr} x_{tr} = 1 \qquad \forall l \in L \tag{5.67}$$

$$\sum_{r \in R(t)} x_{tr} = 1 \qquad \forall t \in T \tag{5.68}$$

$$x_{tr} \in \{0, 1\} \qquad \forall t \in T, \forall r \in R(t) \tag{5.69}$$

In this formulation, constraint (5.65) ensures that maintenance station capacity is not exceeded. Constraint (5.66) connects a rotation to a possible maintenance event. Constraint (5.67) ensures that each line-of-flight is covered in the solution through an assignment. Constraint (5.68) requires each tail to be assigned a particular rotations, including a possible null rotation. Finally constraint (5.69) requires variable integrality.

Augmented Rotation-based Model - Decision Variable: v_{tr}

In addition to the rotation-based approach presented above, we can also formulate this problem using an augmented rotation-based approach. That is, we have a single decision variable v_{tr}, which contains the required maintenance events within each rotation r. As such, we no longer require any additional maintenance event-specific variables in this formulation. On the other hand, by moving the maintenance events into the objective function, we increase the total number of variables.

Sets

R^A	The set of all augmented rotations.
$R^A(t) \subseteq R^A$	The set of all augmented rotations that can be assigned to tail t, $\forall t \in T$.

Parameters

κ_r	The number of maintenance events that are contained within rotation r, $\forall r \in R^A$.
γ_e	The required capacity of maintenance event e, $\forall t \in T, \forall e \in E(t)$.
ξ_{resd}	A binary parameter that indicates whether rotation r containing maintenance event e is located at station s on day d of the planning horizon, $\forall t \in T, \forall e \in E(t), \forall r \in R^A(t), \forall s \in M, \forall d \in D$.

Variables

v_{tr}	a binary variable that is 1 if augmented rotation r is assigned to tail t, $\forall t \in T, \forall r \in R^A(t)$.

Objective:

$$\max \sum_{t \in T} \sum_{r \in R^A(t)} \kappa_r v_{tr} \tag{5.70}$$

Subject to:

$$\sum_{t \in T} \sum_{e \in E(t)} \sum_{r \in R^A(t)} \gamma_e \xi_{resd} v_{rt} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \tag{5.71}$$

$$\sum_{t \in T} \sum_{r \in R^A(t)} \delta_{lr} v_{tr} = 1 \qquad \forall l \in L \tag{5.72}$$

$$\sum_{r \in R^A(t)} v_{tr} = 1 \qquad \forall t \in T \tag{5.73}$$

$$v_{tr} \in \{0, 1\} \qquad \forall t \in T, \forall r \in R^A(t) \tag{5.74}$$

Objective function (5.70) minimizes the total number of maintenance checks that are scheduled. Constraint (5.71) ensures that the capacity of a maintenance station, measured in available man-hours, is not exceeded. In constraint (5.72) we ensure that all lines-of-flight of the planning horizon are covered. In constraint (5.73) we

require that each tail is assigned exactly one rotation, while constraint (5.74) ensures integrality for all variables.

As evident from both formulations above, we require a set $R(t)$ and $R^A(t)$ respectively that enumerate all possible rotations that can be flown by each tail t. The size of this set is an exponential number based on the number of viable connections that can be formed.

Column-Generation based Solution Approach

The column-generation algorithm used to solve both the pure- and augmented rotation-based approach detailed in this section follows the structure provided in [19] and [98], and is outlined in Figure (5.14). We solve these optimization problems using an LP-relaxation with a subset of all possible rotations, which we will refer to as the *active-set*, denoted as Ω_t^A for a particular tail t. With each iteration, we include additional variables that must be evaluated. In typical column generation, we would solve a pricing-problem that provides additional variables (columns), which are stored in the *passive-set*, denoted as Ω_t^P for a particular tail t. In our particular approach, we implement a tail-recursive algorithm to generate all possible rotations a priori, which are then stored in memory and subsequently priced. The utility of our approach depends on the size of the input data. As we demonstrate in our computational experiments, generating these rotations a priori allows for faster completion of the pricing stage in the column-generation-based approach.

For both the pure- and augmented rotation-based approaches, we begin this process by first determining a set of feasible rotations that can be used in the initial formulation. That is, we include the set of original rotations based on the input LOF

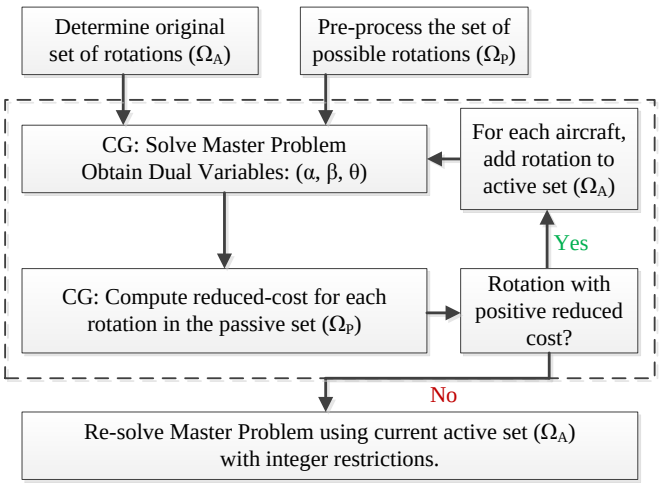


Figure 5.14: Column-generation algorithm overview

assignment. In addition, we include a set of *null* rotations, which allow aircraft to remain on the ground at a particular station, if all rotations are assigned. Although this combination of rotations is undesirable (since we are trying to maximize the number of maintenance events), it will however, lead to a feasible initial solution, which can be used to jump-start the column generation algorithm.

For the augmented rotation-based approach, once we have obtained the set of rotations, we insert maintenance events. More specifically, for each rotation, we determine the set of possible maintenance events that could be completed, if this rotation were assigned. We provide the actual algorithm to generate augmented rotations later in this section.

Next, for either the pure- or augmented rotation-based approach, we solve the first-iteration of the above-mentioned optimization problem under its LP-relaxation. Once we have obtained such a solution, we evaluate the optimality condition for each of the rotations in the passive-set (Ω_t^P) of potential variables. This is often referred to as the pricing problem, i.e. we are pricing the passive-set variables to determine if any rotations would lead to an improvement in the objective function. In this case,

we are solving a maximization problem, which implies that at the optimal solution, the reduced-cost of each variable is 0 or negative. To compute this reduced-cost for each of the passive variables, we require the dual-variables from the constraints of our master problem. For each of the constraints we define a dual-variable.

The dual-variables for the pure rotation-based approach with decision variables w_{eds} and x_{tr} are shown below.

$$w_{eds} - \sum_{t \in T} \sum_{r \in R(t)} \kappa_{resd} x_{tr} \leq 0 \qquad \forall e \in E, \forall s \in S, \forall d \in D \qquad (\alpha_{esd}) \tag{5.75}$$

$$\sum_{t \in T} \sum_{r \in R(t)} \delta_{lr} x_{tr} = 1 \qquad \forall l \in L \qquad (\beta_l) \tag{5.76}$$

$$\sum_{r \in R} x_{rt} = 1 \qquad \forall t \in T \qquad (\theta_t) \tag{5.77}$$

The dual-variables for the augmented rotation-based approach with decision variable v_{tr} are shown below.

$$\sum_{t \in T} \sum_{e \in E(t)} \sum_{r \in R(t)} \gamma_e \xi_{resd} v_{rt} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \qquad (\alpha_{sd}) \tag{5.78}$$

$$\sum_{t \in T} \sum_{r \in R(t)} \delta_{lr} v_{tr} = 1 \qquad \forall l \in L \qquad (\beta_l) \tag{5.79}$$

$$\sum_{r \in R} v_{rt} = 1 \qquad \forall t \in T \qquad (\theta_t) \tag{5.80}$$

With these dual-variables, we can write down the problem-specific reduced-cost equation, shown in equation (5.81) for the rotation-based problem including the w_{eds} variable and in equation (5.82) for the augmented rotation-based problem. Each of these respective equations will be used to price the passive rotations.

$$\overline{c_{rt}^1} = 0 - \sum_{e \in E(t)} \sum_{s \in S} \sum_{d \in D} \kappa_{resd} \alpha_{esd} - \sum_{l \in L} \delta_{lr} \beta_l - \theta_t \tag{5.81}$$

$$\overline{c_{rt}^2} = \kappa_r - \sum_{e \in E(t)} \sum_{s \in S} \sum_{d \in D} \xi_{resd} \alpha_{sd} - \sum_{l \in L} \delta_{lr} \beta_l - \theta_t \tag{5.82}$$

Formulating a Pricing Problem

So far, we have assumed that a set of all possible rotations already exists, and that we compute the reduced-cost based on the equations above. Alternatively (for example, when the set of all possible rotations is too large to enumerate explicitly) we can formulate the problem of finding a new rotation with positive reduced-cost as an optimization problem. That is, to determine a new rotation, we formulate an optimization problem that maximizes the reduced cost (since we are solving a maximization problem). We solve this optimization problem *for each* tail in the set T.

Once we solve this set of optimization problems, we have two possible outcomes for each tail. In the case where we find a variable with a positive reduced-cost, we must bring this variable into the active-set and re-solve the restricted master problem. Since we are solving this problem on per-tail basis, during each iteration, we may include several new variables in the master problem. In the second outcome, the optimization for each of the tail's sub-problems yields only negative (or 0) reduced-cost. In that case, we have reached an optimality condition and no further processing is required. To determine a positive reduced cost, we can formulate the following optimization problem.

Objective (for each tail):

(5.83) $$\max \overline{c_{rt}}$$

Subject to:

(5.84) $$\text{feasible rotation} \qquad r \in R(t)$$

This sub-problem can be formulated as a standard negative min-cost flow formulation with an added side constraint. That is, to determine a feasible rotation for a particular tail t, we can start with this tail at its current location and then compute each possible connection to other lines-of-flight which this tail could actually perform. We show this flow graphically in Figure (5.15) below. In this diagram, we label lines-of-flight with the letter l while overnight maintenance events are denoted by e. In this formulation, we also include a *null* arc that flows directly from the starting node to the terminal node. In this case, this arc represents a ground-stay for the particular tail. As such, we are always guaranteed a solution to the max-cost-flow problem.

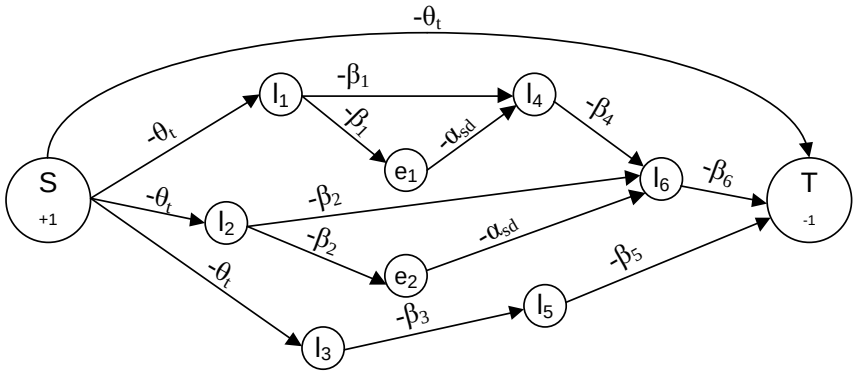


Figure 5.15: Max-cost flow sub-problem

The network for the min-cost flow formulation is created by evaluating the starting position of tail t. From there, we enumerate all possible flights (in time & space) that

may follow. For each subsequent node, we again enumerate all possible originating events emanating at the arrival station. We continue this process until we have reached the planning horizon for which this problem is being solved. We note that in the formulation that follows, no connection between a node and itself exists, allowing the flow across the network to always be positive and without the possibility of cycles. We use the following notation for the max-cost flow sub-problem.

Sets

N	the set of nodes representing the flights and events that must be performed.
L	the set of nodes representing lines-of-flight, $L \subseteq N$.
E	the set of nodes representing maintenance events, $E \subseteq N$.

Parameters

$a(l) \in D$	The day of the planning horizon at which line-of-flight l terminates, $\forall l \in L$.
$l(e) \in D$	The deadline on which event e must be performed, $\forall t \in T, \forall e \in E(t)$.
c_{ij}	the cost of going from line-offlight or event i to flight or event j. This cost is defined by the dual variables as defined in the figure above, $\forall i, j \in F$.
b_f	the supply or demand at a node in the event. This is 0 except for the beginning and ending node, $\forall f \in F$.
δ_{cte}	The limit for maintenance counter c on tail t with its respective maintenance event e, $\forall c \in C, \forall t \in T, \forall e \in E(t)$.
κ_{cl}	The contribution of line-of-flight l to maintenance counter c, $\forall l \in L$.
α_{ct}	The initial value for maintenance counter c on tail t, $\forall t \in T, \forall c \in C$.

Decision Variable

x_{ij}	a binary variable that is 1 if line-of-flight or event i is followed by flight-of-flight or event j and is 0 otherwise, $\forall i, j \in F$.

Objective:

$$\max \sum_{i \in F} \sum_{j \in F} c_{ij} x_{ij} \tag{5.85}$$

Subject to:

$$\sum_{j \in F} x_{ij} - \sum_{j \in F} x_{ji} = b_i \qquad \forall i \in F \tag{5.86}$$

$$\alpha_{ct} + \sum_{\substack{i \in L: \\ a(i) \leq l(e)}} \kappa_{ci} x_{ij} \leq \delta_{cte} \qquad \forall c \in C, \forall e \in E \tag{5.87}$$

$$0 \leq x_{ij} \leq 1 \qquad \forall i \in F, \forall j \in F \tag{5.88}$$

In the objective (5.85), we seek to maximize the total cost of creating a rotation of flights and maintenance events in the network. In addition, in constraint set (5.86), we ensure that the demand/supply is met. As seen in in the diagram in Figure (5.15), the only supply ($+1$) is at the first node, and the only demand (-1) is located at the last (terminal) node. Finally, constraint (5.87) is required to ensure that only LOFs are assigned that are indeed within the counter limits of the maintenance events.

If the solution to the max-cost flow problem either produces a negative or 0 objective, we proceed to the next tail t. In the case where the objective of the max-cost flow problem is positive, we bring the rotation r that provides this positive reduced-cost into our restricted master problem and solve the problem yet again. This process then continues until all max-flow sub-problems for all tails return a negative (or 0) reduced cost, which acts as our certificate for optimality.

Generating Maintenance Rotations

For our computational experiments, instead of solving an optimization problem to

generate rotations for the passive-set of variables, we are able to generate rotations using a tail-recursive approach prior to starting the optimization process. Using data from our medium-size carrier, we are able to pre-process the set of all possible rotations for each tail that exist over the planning horizon. We emphasize that this may not always be possible and is dependent on the size of the input data. However, when it is feasible, it does offer considerable solution speed advantages, as the max-cost flow problem does not need to be built and solved for each tail during each iteration.

This generation is detailed in three algorithms in the appendix. First, in Algorithm (1) we determine all of the initial lines-of-flight on which all subsequent rotations will be built. This subset of the entire set of LOFs then provides the input required for Algorithm (2). Both of these algorithms are required for the pure- and augmented rotation-based models.

Using the set of initial lines-of-flight as the building blocks for all further rotations, we now generate actual rotations. Detailed in Algorithm (2), we begin by evaluating each of the initial lines-of-flight that were created earlier and recursively evaluate each possible connection opportunity. When solving the pure rotation-based approach which uses the w_{eds} to represent maintenance events, we can stop with Algorithm (2). However, when solving the augmented-rotation-based model, we also need to insert maintenance events into these rotations. This process is completed in Algorithm (3).

For each tail, we analyze the set of particular rotations and determine which of the three check types can be completed during this particular rotation. For each possible check combination, we make a copy of the rotation without maintenance events and then insert the events accordingly. For example, if a rotation allows for

both the A-check and the B-check, then we generate a total of 4 rotations:

1. The rotation with no maintenance events.
2. The rotation containing only the A-check.
3. The rotation containing only the B-check.
4. The rotation with both the A- and B-checks.

It should be noted that we do include the rotation without any maintenance events to allow for capacity management within the mathematical model.

The advantage of this recursive rotation-generation algorithm is two-fold. First, all possible rotations can be quickly pre-processed and stored in memory. That is, we store all rotations in the passive set (Ω_t^P) of variables. Second, once we solve the master problem under its LP-relaxation, we use the dual-variables to price each of these rotations. In the next section, we provide computation results that illustrate the advantage of including overnight swaps in contrast to the original MRP problem. In addition, we provide a comparison regarding the advantages/disadvantages for each of the formulations above. While equivalent in capturing the same objective, the formulations differ in size and solve-ability.

5.6.3 Computational Results

The data used to evaluate the effectiveness of the MRP and MRP-OS models stem from a major US carrier. As detailed in Table (5.6), this carrier is of medium size and features a mixture in fleet types across a total of 71 tails. In Table (5.6), we provide the total number of maintenance opportunities given the respective length of the planning horizon. This number reflects the total opportunities that exist during

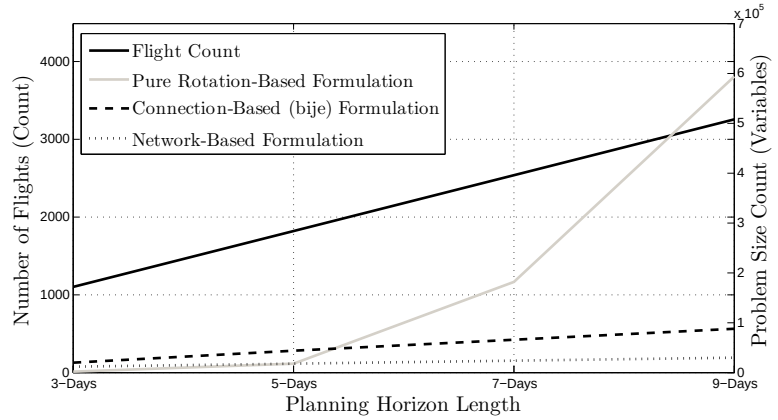
this time frame, i.e. whenever an aircraft remains overnight at a maintenance station and has adequate time along with enough man-hour capacity to perform a particular maintenance event.

Horizon Length	Flights Count	Duties Count	Stations Count	Fleet Types	Tails Count	Event Count	Opportunities
3-Day	1,105	198	17	6	71	213	571
5-Day	1,815	328	17	6	71	213	1,644
7-Day	2,543	456	17	6	71	213	2,499
9-Day	3,263	586	17	6	71	213	3,386

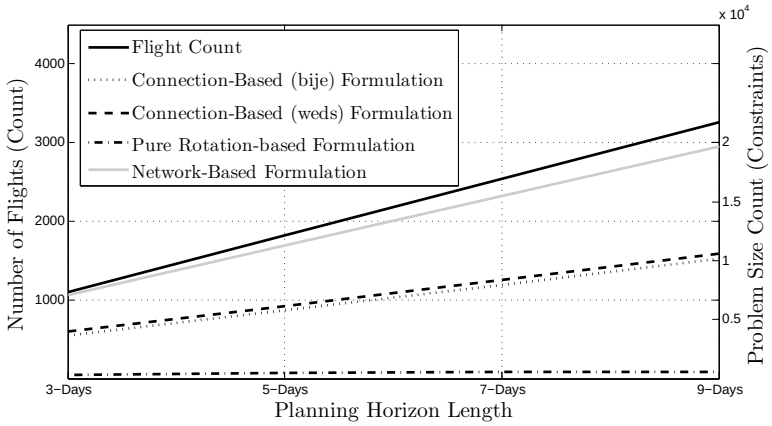
Table 5.6: Flight data from major US carrier

In §5.6.2, we presented five mathematical formulations that can be used to represent this particular problem. The first formulation, a line-of-flight-based representation, contained the least amount of information within a single decision variable. On the opposite end, the rotation-based formulation contained the most information within a single decision variable. Here, the primary decision focused on the assignment of an entire rotation (lines-of-flight spanning several days) to a single tail. Upon implementation these models will have substantially different sizes in terms of both then number of variables and the number of constraints.

Based on the carrier data from Table (5.6), we can evaluate the effectiveness of each of these models and their solve-ability. In Figure (5.16), we compare the actual size of the different formulations as a function of the total number of flights during each respective planning horizon length. In terms of the variables, we note that each model linearly increases with the length of planning horizon, except for the rotation-based approach. The rotation-based approach displays an exponential relationship between the length of the planning horizon and the number of variables. In terms of the constraints, the rotation-based model clearly dominates all other formulations



(a) Variable Count



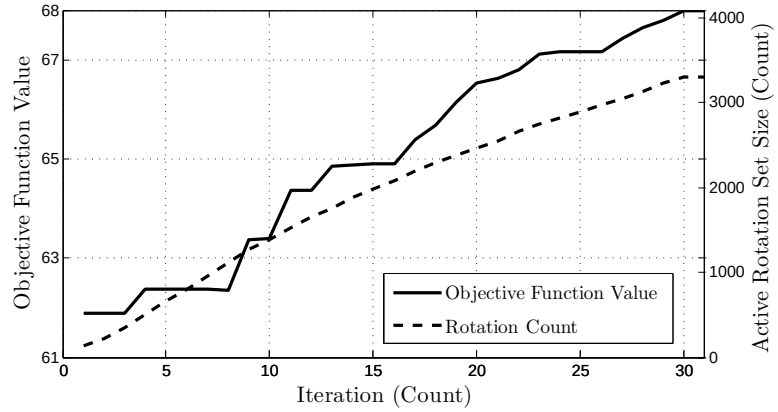
(b) Constraint Count

Figure 5.16: Problem size comparison

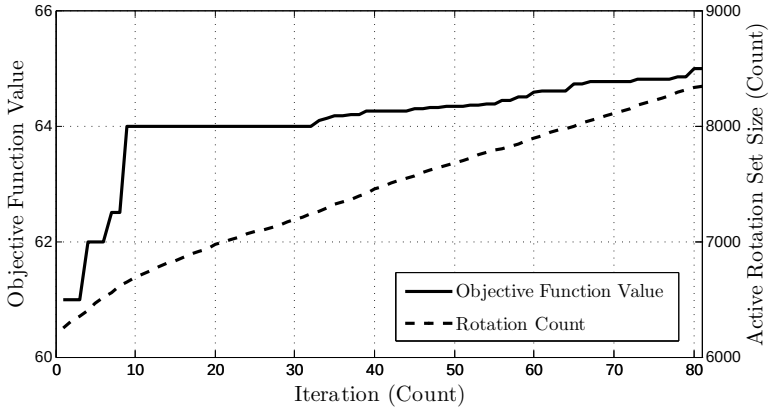
with the lowest number of constraints. This is due to the fact that the number of constraints only increases by the additional number of days of the planning horizon, as a direct result of the capacity constraint.

Based on the variable count, it may appear that the rotation-based approach is not very effective, especially given that it requires a large number of variables. It should be noted however that while the rotation-based formulation features over $600,000$ potential rotations for the 9-day planning horizon, these variables are stored primarily in the passive set (Ω_t^P) of variables, i.e. and thus are not included directly in the solution approach. In Figure (5.17), we show the progression of the column-

generation approach and the number of variables actually required. We note that each of the models can solve this problem for a 9-day planning horizon and do so effectively.



(a) Rotation-based model using w_{eds}



(b) Rotation-based model using v_{tr}

Figure 5.17: Problem size comparison for rotation-based approach

In addition, we note that the pure rotation-based model using the w_{eds} variable solves in one-third of the number of iterations required to solve the augmented-rotation-based model: 32 iterations compared to 81 rotations. However, the augmented-rotation-based model features a tighter solution bound. More specifically, the pure rotation-based approach terminates the restricted master problem with an LP-optimal solution of 68 maintenance events. Once we require integrality on both the v_{tr} and

w_{eds} variables, we obtain a final solution of 65 maintenance events. On the other hand, the augmented rotation-based approach reaches an LP-optimal solution of 65 maintenance events, which does not change once we require integrality on the v_{tr} variables. Given this result, using the augmented rotation approach appears preferable in terms of overall solution quality and ability to reach an IP equivalent solution.

Comparison of MRP-OS to MRP Model

In the previous section, we introduced the Maintenance Recovery Problem (MRP). In the MRP formulation, we used the current tail assignment as input and subsequently assigned as many maintenance events as possible. In contrast to the MRP, we now allow overnight swaps between tails within the same fleet type. As a direct result of these overnight swaps, the total number of maintenance events that can be completed increases significantly.

The effectiveness of the MRP-OS is driven by two factors. First, overnight swaps allows those maintenance events to take place that were capacity constrained before. Second, overnight swaps provide access to maintenance opportunities that did not exist previously. To compare the effectiveness of the MRP-OS versus MRP for each of these factors, we need to eliminate the effects of capacity constraints. That is, we want to quantify how often maintenance capacity restrictions influenced the objective (of maximizing the number of maintenance events) and how often the flexibility due to overnight swaps contributed to additional maintenance coverage.

In Figure (5.18), we compare the total number of maintenance events that can be completed between the original Maintenance Routing Problem (MRP), and our formulation of the Maintenance Routing Problem with Overnight Swaps (MRP-OS) as a function of the available capacity at each of the maintenance stations. While

not surprising, as more capacity becomes available, more maintenance events can be completed. More interestingly, it should also be noted that maintenance capacity at stations is a tight constraint, but only up to 80% of the original capacity. That is, once we have reached 80% of this input capacity, adding additional capacity has no effect on the ability to cover additional maintenance events. By implication, we can conclude that once this capacity limit has been reached, the network design itself prevents maintenance events from being covered, i.e. a tail is required to perform maintenance, but cannot do so because no timely path to a maintenance station exists.

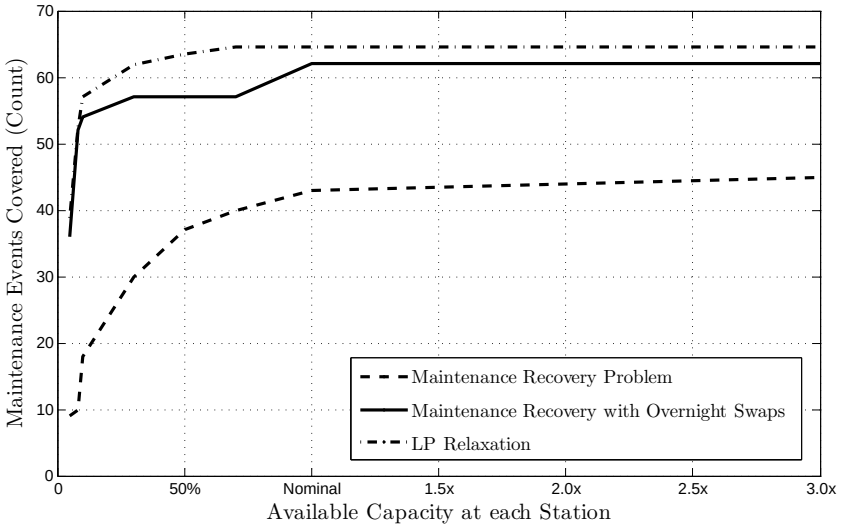


Figure 5.18: Problem comparison to the Maintenance Routing Problem

Moreover, as is evident from the Figure (5.18), allowing overnight swaps significantly increases the total number of maintenance events that can be completed. It should be noted that in this case, the planning horizon is a 5-day horizon, during which a maximum number of 74 events are required, but the LP relaxation (and by implication the network structure) only allows 65. Allowing overnight swaps provides for coverage of 64 of these events, while the traditional MRP only covers 45 events.

5.7 Conclusions

Recovering rotation assignments and creating maintenance feasibility from unexpected disruptions are important problems in airline operations. Once rotation assignments have been recovered, a maintenance assignment must be generated to ensure continued and safe operation. Up to this point, we have focused on the problem in which we maximize the number of maintenance events that can be performed given the set of recovered rotations. In addition, as we have shown, we can consider alternative objective functions that mirror airline business processes. One alternative is to minimize event earliness (relative to the deadline) of all maintenance events. The other is to consider the maintenance recovery problem from a man-power scheduling perspective where the objective is to spread capacity utilization as evenly throughout the week as possible.

In our work, we demonstrated that the Maintenance Recovery Problem (MRP), while easy to explain, can be difficult to solve, especially if the objective function and constraints create underlying symmetry in the problem structure, as evidenced in the even maintenance assignment objective. Furthermore, as we have shown, different solution and decomposition approaches must be applied to determine the optimal solution. In the case of solving the even maintenance assignment problem, a Lagrangian relaxation approach provided an optimal solution when solving a MIP using the standard branch-and-bound technique failed to converge within a timely fashion.

Lastly, in this chapter we considered the maintenance routing problem with overnight swaps (MRP-OS). In this variation, we change the aircraft assignment and thus decide the final destination of each aircraft at the end of each day (in-

sofar possible given the lines-of-flight available) during the planning horizon. This additional freedom increases the complexity of the problem, but also improves the possible maintenance event coverage. The solution to the MRP-OS increased maintenance event coverage by as much as 42% (45 events to 64 events) in our test instances when compared to the MRP solution.

To model these overnight swaps, we presented five different approaches, including: a line-of-flight- (or network-based) based formulation, two connection-based (b_{ije} and w_{eds}) approaches and two rotation-based (pure and augmented) approaches. As shown in §5.6.3, these different formulations and structures imply different problem sizes.

In the next chapter, we extend the maintenance recovery problem further in two ways. First, we allow the deadline of the maintenance check to depend on the new rotations, rather than forcing rotations to conform to the pre-processed deadlines. These maintenance checks then become part of the decision of the optimization model. This approach stands in contrast to the overnight-swap model presented in this chapter, where we removed any line-of-flight assignment that did not meet the respective counter limits. As we will show, removing these LOFs from the solution space may be sub-optimal.

In the second extension, we do not just consider the upcoming maintenance check, but rather all subsequent events. More specifically, maintenance checks are recurring, because the next check influences the deadline of its following check. This in turn affects the deadline of the third subsequent check and so on. Solving the maintenance recovery problem with all subsequent checks creates additional complexity, but also realizes a more realistic instance of the problem. In addition, we will demon-

strate that only a rotation-based formulation, similar to the rotation-based approach presented in this chapter, allows us to solve this particular problem effectively.

CHAPTER VI

Recurrent Maintenance Scheduling

6.1 Introduction

In Chapter V we presented the Maintenance Recovery Problem (MRP) and the Maintenance Recovery Problem with Overnight Swaps (MRP-OS). In both of these approaches, we used the recovered set of assigned aircraft rotations as input to pre-compute the set of upcoming, required maintenance events. As noted in Chapter V, the maintenance events were pre-processed based on the *current* set of assigned rotations and their respective effects on maintenance counters. Thus, for the MRP-OS, any overnight swap performed was required to meet the maintenance check limits of the original rotation. (For more information about this limitation, we refer the reader to §5.2.3 in Chapter V). With these pre-computed maintenance events, the MRP and MRP-OS maximized maintenance coverage, i.e. each model covered as many of the maintenance events as possible subject to the underlying structure of the flight network and its overnight swap possibilities.

As noted in Chapter V, the solution to the MRP and MRP-OS model may not be maintenance feasible and will thus require costly over-the-day swaps to rectify. This infeasibility exists due to two primary reasons. First, the underlying network may not contain adequate flexibility to allow all maintenance events to be covered. Sec-

ond, because all maintenance events are pre-processed, the effective solution space for the MRP and MRP-OS model is constrained. In this chapter, we address this second cause of infeasibility and provide greater flexibility to expand the solution space, thereby increasing the likelihood of feasibility. It turns out that even with all combinations of lines-of-flight, a maintenance feasible assignment may not exist. In this case, airlines use other measures to reconcile the rotation assignments with maintenance feasibility. Such measures often create crew and passenger complexity, in addition to generating financial implications. As such, we do not evaluate such additional recovery measures, but rather demonstrate in our computational experiments, that reconfiguring lines-of-flight is sufficient to attain feasibility.

In the solution approaches presented in this chapter, we no longer pre-define maintenance events and their deadlines. Given that these events are no longer predefined, we will no longer refer to them as *maintenance events*, but rather as instances of *maintenance checks*. We now allow deadline decisions themselves to be part of the optimization process and require that maintenance feasibility be maintained.

In §6.2 we begin by continuing to focus on only scheduling the first of each maintenance check for each tail, but we now allow greater flexibility in changing the rotation. No longer must the new rotation be compatible with the prior rotation's maintenance deadlines. Instead, we focus on ensuring compatibility between the new rotation assignment and its corresponding maintenance assignment. Next, we expand our scope to consider multiple (recurring) maintenance checks. Specifically, in §6.3, we realize that maintenance checks are cascading decisions, in which the timing, location and assigned rotation of the first check has an impact on the second, which has an impact on the third and so on. As we will show, solving a recurring model becomes

significantly more difficult due to the increase in both the number of variables and constraints. In §6.4 we develop our approach further to allow for extended horizon recovery, using a rolling horizon approach. Finally, in §6.5 we conclude our findings and motivate several extensions to this problem, including an approach to use our rolling horizon model to perform maintenance capacity planning.

6.2 The First Maintenance Check Problem (FMCP)

6.2.1 Overview

In this section, we focus on maintenance recovery from the perspective of the *first* maintenance check. That is, similar to the approach in the MRP and MRP-OS models, only one check is scheduled for each type over the planning horizon. In contrast to the MRP and MRP-OS model, however, the maintenance events are no longer pre-processed, but are rather decisions within the optimization model. The advantage of creating instances of checks as necessary (rather than pre-processing) is illustrated in Figure (6.1).

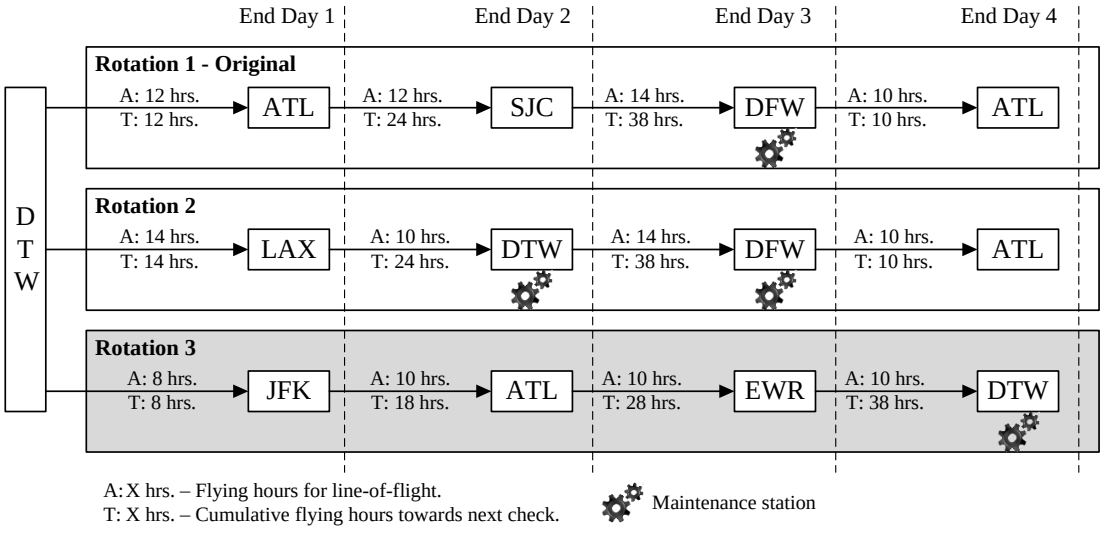


Figure 6.1: Rotation choice for lines-of-flight leaving DTW

In Figure (6.1), we feature three possible rotations that can be assigned to a tail starting on day 1 at station DTW. The first rotation is the original assignment and requires an A-check at the end of day 3 because the 40-hour flying-hours limit has been reached. In the MRP-OS model, the second rotation is also a possible assignment, because the tail is able to reach a maintenance station prior to the 40-hour limit on day 3. However, in the MRP-OS model, rotation 3 is excluded from the possible solution space despite its validity. That is, the rotation does not overnight at a maintenance station on or before day 3. In reality, it is a feasible rotation because the required maintenance check could be performed on day 4 at DFW and still remain within the maintenance limits, because this rotations incurs fewer flight hours over the first three days relative to the other two rotations.

As illustrated by this example, the MRP and MRP-OS problems do not capture the full solution space because they ignore possible rotations in which a required maintenance check can legally be completed after the preprocessed deadline. We now present an optimization model that is able to capture these additional choices and solve the *First Maintenance Check Problem* (FMCP).

6.2.2 Formulating the First Maintenance Check Problem

We now formulate the First Maintenance Check Problem. The formulation shown here is similar to our solution approach presented in §5.6.2 in Chapter V. We employ the following notation for our formulation. This formulation and notation then is extended in the remainder of this chapter.

Sets

T	The set of tails.
D	The set of days in the planning horizon.
M	The set of maintenance stations in the network.
R	The set of all rotations.
$R(t) \subseteq R$	The set of all rotations that can be assigned to tail t, $\forall t \in T$.
L	The set of lines-of-flight.
$L(t) \subseteq L$	The set of lines-of-flight that can be flown by tail t, $\forall t \in T$.
C	The set of check types.

Parameters

θ_c	The amount of maintenance man-hour capacity required for maintenance check c, $\forall c \in C$.
$m(r,d) \in M$	An binary parameter that provides the station where rotation r overnights on day d of the planning horizon, $\forall r \in R, \forall d \in D$.
δ_{lr}	A binary parameter that indicates if line-of-flight l is contained in rotation r, $\forall t \in T, \forall r \in R(t), \forall l \in L(t)$.
ρ_{md}	The available capacity at station m on day d in the planning horizon, $\forall m \in M, \forall d \in D$.
$r(d,t,c,r)$	An binary parameter that is 1 if rotation r triggers the first check of type c to be completed by day d for tail t, $\forall d \in D, \forall t \in T, \forall c \in C, \forall r \in R(t)$.

Variables

x_{tr}	a binary variable that is 1 if rotation r is assigned to tail t, $\forall t \in T, \forall r \in R(t)$.
w_{tcmd}	a binary variable that is 1 if tail t receives the *first* check of type c on day d of the planning horizon at station m, $\forall t \in T, \forall d \in D, \forall c \in C, \forall m \in M$.

Objective:

$$\min \sum_{t \in T} \sum_{c \in C} \sum_{m \in M} \sum_{d \in D} w_{tcmd} \tag{6.1}$$

Subject to:

$$\sum_{t \in T} \sum_{r \in R(t)} \delta_{lr} x_{tr} = 1 \qquad \forall l \in L \tag{6.2}$$

$$\sum_{r \in R(t)} x_{tr} = 1 \qquad \forall t \in T \tag{6.3}$$

$$\sum_{m \in M} \sum_{\substack{d_1 \in D: \\ d_1 \leq d}} w_{tcmd_1} - \sum_{r \in R(t)} r(d,t,c,r) x_{tr} \geq 0 \quad \forall t \in T, \forall d \in D, \forall m \in M, \forall c \in C \tag{6.4}$$

$$w_{tcmd} - \sum_{r \in R(t)} m(r,d) x_{tr} \leq 0 \quad \forall t \in T, \forall m \in M, \forall d \in D, \forall c \in C \tag{6.5}$$

$$\sum_{c \in C} \sum_{t \in T} \theta_c w_{tcmd} - \rho_{md} \leq 0 \qquad \forall m \in M, \forall d \in D \tag{6.6}$$

$$x_{tr} \in \{0,1\} \qquad \forall t \in T, \forall r \in R(t) \tag{6.7}$$

$$w_{tcmd} \in \{0,1\} \qquad \forall t \in T, \forall c \in C, \forall m \in M, \forall d \in D \tag{6.8}$$

In this approach, the objective function in equation (6.1) minimizes the total number of maintenance checks assigned. This is in contrast to maximizing maintenance coverage in the MRP and MRP-OS model. Here, we are given underlying rotations and must assign maintenance checks. As long as a feasible maintenance assignment exists and because the day and station of a maintenance check is a decision (rather than being preprocessed), this formulation will lead to solution in which we may complete a single "first" check. That is given a particular rotation, it could be possible that the first required check falls outside of the assignment window. Finally, in this approach we are inserting maintenance checks as necessary and thus minimizing the total number (of checks) ensures that at most one is scheduled for each tail.

Similar to the MRP problem in Chapter V, the FMCP is generally a feasibility problem and thus can benefit from additional (secondary) objective functions. As before, two possible secondary objective functions could include minimizing the ear-

liness of the particular check or the even maintenance event assignment as illustrated in Chapter V.

In constraint (6.2), we require that all lines-of-flight in the schedule are covered. Next, in constraint (6.3) all tails are permitted to perform exactly one rotation. Constraint (6.4) requires that if a rotation requires a maintenance check by a particular deadline, then this check must be scheduled on (or before) this particular deadline. Constraint (6.5) requires that a maintenance check can only be scheduled on a given day at a given station when a rotation places the corresponding tail at that station on that day. Finally, constraint (6.6) requires that station capacity constraints are observed. Constraint (6.7) and (6.8) require variable integrality.

6.2.3 Solving the First Maintenance Check Problem

To solve the FMCP, we employ a column generation framework. As before in Chapter V, we solve the problem using our active-set (Ω_A) of variables (the recovered set of rotations) and then price additional rotations in a second, pricing step in our solution algorithm.

To begin the column generation approach, we first denote the following dual variables for the linear program presented in the previous section.

$$\sum_{t \in T} \sum_{r \in R(t)} \delta_{lr} x_{tr} = 1 \qquad (\gamma_l) \tag{6.9}$$

$$\sum_{r \in R} x_{tr} = 1 \qquad (\theta_t) \tag{6.10}$$

$$\sum_{m \in M} \sum_{\substack{d_1 \in D: \\ d_1 \leq d}} w_{tcmd_1} - \sum_{r \in R(t)} r(d,t,c,r) x_{tr} \geq 0 \qquad (\alpha_{tdmc}) \tag{6.11}$$

$$w_{tcmd} - \sum_{r \in R(t)} m(r,d) x_{tr} \leq 0 \qquad (\beta_{tmdc}) \tag{6.12}$$

Using these dual-variables we can determine the reduced-cost for all additional rotations. In our approach, we are pricing additional rotations, i.e. additional x_{tr} variables. Thus, to compute a reduced cost for a new rotation, we must consider those constraints in which x_{tr} actually appears, including constraints (6.2), (6.3), (6.4), and (6.5). The reduced-cost equation used to price these additional variables is provided in equation (6.13).

$$\overline{c_{rt}} = 0 - \sum_{l \in L} \delta_{lr} \gamma_l - \theta_t + \sum_{t \in T} \sum_{m \in M} \sum_{d \in D} \sum_{c \in C} r(d,t,c,r) \alpha_{tcmd} + \sum_{t \in T} \sum_{m \in M} \sum_{d \in D} \sum_{c \in C} m(r,d) \beta_{tmdc} \tag{6.13}$$

As noted in Chapter V, the pricing problem is solved using an enumerative tail-recursive approach given the size of our data set. However, if enumeration is not possible, generation of new rotations is possible through the same min-cost flow problem described in Chapter V.

6.2.4 Computational Results

In our solution approach, we begin by solving FMCP using the active-set (Ω_A) of rotations from the recovered set of assigned rotations. We solve this problem to optimality and then price out each of the additional rotations that still exist in our set of passive rotations (Ω_t^P). Any rotation that has a negative reduced cost (the objective of our problem is to minimize maintenance checks) will then be included in the restricted master LP problem, which is subsequently resolved to obtain new dual variables. This process continues until either no rotations provide a negative reduced cost or all rotations have been included in the master problem. At this point, we re-solve the restricted master problem as an integer program using the current set of active rotations, Ω_A.

We apply this solution algorithm to the same US carrier data that was used in Chapter V. We feature a data set for a medium size airline with a 5-day planning window, 71 aircraft and 2,176 flights. Based on these data, Figure (6.2) demonstrates the effectiveness of our approach. As seen in the figure, a total of 104 iterations are required to solve our instance of the FMCP. Furthermore, for comparison purposes we illustrate both the LP and IP objective function value during each iteration. As noted in the figure, as the algorithm progresses, the objective function values continue to diverge, emphasizing the fact that the LP relaxation of this problem is a poor bound on the integer problem equivalent.

As the solution algorithm progresses, additional rotations are included in the restricted master problem. Figure (6.3) illustrates the number of rotations that must be included to attain a solution. For our medium size airline, at the optimal solution, we include $15,048$ rotations out of a possible $856,537$ rotations.

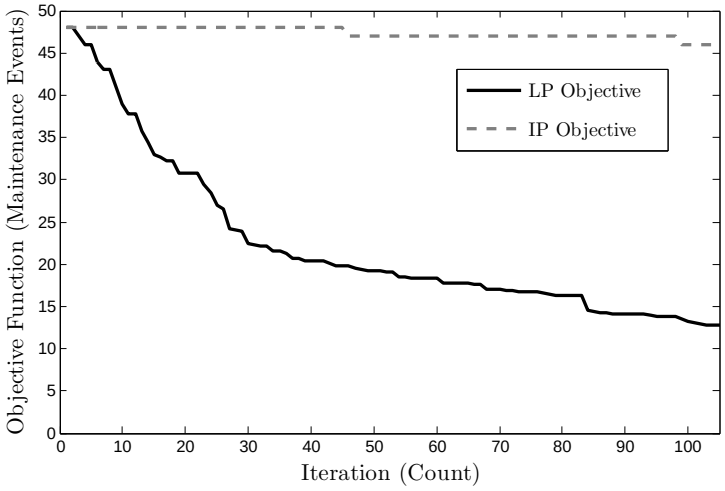


Figure 6.2: Objective function value comparison

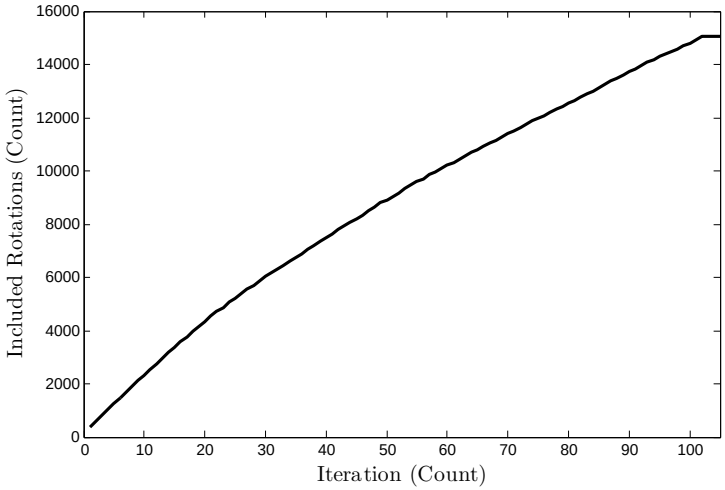


Figure 6.3: Required rotations in solution

In Figure (6.4), we contrast each of these iterations with the time taken to solve the actual problem. As noted, each iteration for our instance of the problem solves in a matter of seconds. For comparison purposes, we also solve the IP-equivalent solution during each iteration. Similar to the rotation-based approach of Chapter V, we solve the FMCP using the LP-relaxation until we reach optimality. We then branch on the included variables to obtain an integer solution.

As noted in the figure, the solution to the IP is also relatively fast and requires

at most 45 seconds. Due to the random starting nature of the branching process, we note some variation in the solution time, however, on average, it trends upward with iteration count. For comparison purposes, we also provide a 3-period moving average for the IP solution time. The moving average removes some of the variance and illustrates the upward moving trend of the solution time. These results are promising, because for our data instance we can solve the the FMCP in about two minutes (81 seconds for the LP column-generation process and 45 seconds for the IP branching process) using this modified column-generation approach.

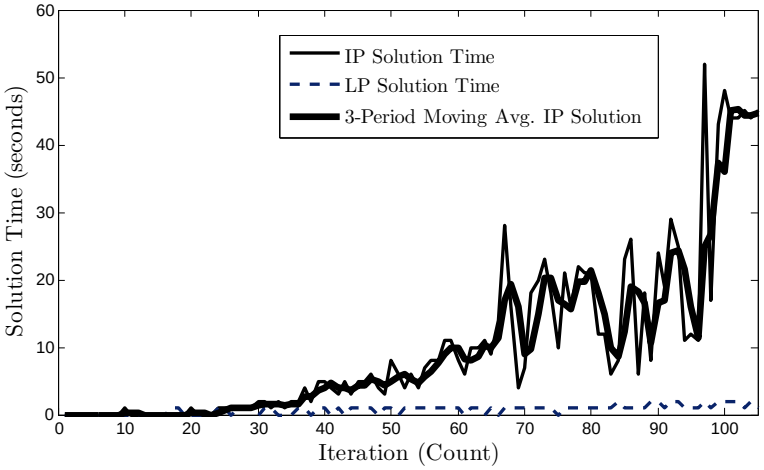


Figure 6.4: Required solution time per iteration of column generation algorithm

In this section, we solved the problem of scheduling the *first* maintenance check, while choosing a particular rotation for each tail. We solve this problem quickly (2 minutes) and provide a maintenance feasible assignment for a 5-day horizon. However, scheduling only the next check ignores the underlying recurrence of a maintenance check. That is, the decision of scheduling the first A-check impacts the decision on the second A-check, which in turn affects the third A-check and so on. In the next section, we consider the problem in which we not only consider the next check, but all subsequent checks that fall into the planning horizon.

6.3 Recurring Maintenance Problem (RMP)

6.3.1 Recurring Maintenance Events

The recovery problem we have discussed up to this point is a simplification of the real-world problem, in that we only consider the first (i.e. next due) maintenance check of each type, ignoring any future checks that may be required. Ignoring such future checks, however, poses several difficulties. First, omitting all future checks may lead to invalid, maintenance infeasible tail assignments. More specifically, if we only consider the next check, there is no guarantee that a future maintenance check can be reached, because the underlying network structure may not allow the aircraft to reach a subsequent maintenance station when required. Second, by disregarding future checks, we also ignore maintenance station capacity. That is, without considering maintenance check recurrence, it is difficult to understand the capacity requirements at each of the maintenance bases.

Our work so far has provided us with insights which we will now apply to solve the more realistic (and more challenging) problem in which we consider not only the first, but subsequent maintenance checks as well. Scheduling subsequent checks increases the problem complexity, not only because the total number of maintenance checks increases, but also because the decision regarding the deadline of one check determines the deadline of the subsequent check, which in turn affects its subsequent check. We will refer to solving a problem with recurring maintenance checks as the Recurring Maintenance Problem or *RMP*.

To illustrate the increased complexity of scheduling recurring maintenance checks, we consider the following example. Figure (6.5) shows the current state of the maintenance counters on a particular tail, Tail #1, which just completed an A-check.

Given this status, we can assign several possible lines-of-flight, each of which feature a different time as to when the next, and all subsequent, A-checks will take place over a five day planning horizon.

Tail #1		Current	Limit
A-check	Hours (H):	0	40
	Cycles (C):	0	10

Figure 6.5: Tail 1: Beginning A-check status

As noted in Figure (6.6), the first maintenance plan features an A-check on day 3, as required by the limit on the number of flight hours (40 hrs.). The decision to schedule the event on day 3 then forces another A-check to take place on day 5. In contrast, the second maintenance plan performs the first A-check one day early. Performing this check early also forces the second subsequent check to be performed a day earlier on day 4. There are several additional plans (not-illustrated) in this scenario, where checks are performed earlier than specified by the deadline. Note, however, that in this case, plan 1 may appear preferable because maintenance checks are pushed further into the future; however, the second plan may be desirable in cases of limited maintenance capacity given that multiple aircraft may have a similar assignment.

6.3.2 Connection-based Formulations

To formulate the RMP, we now extend the connection-based approach that was introduced in Chapter V. The underlying network structure is formed by the binary decision variable x_{ijt}, which is similar to the previous formulation in that this variable is one if line-of-flight i is connected to line-of-flight j and assigned to tail

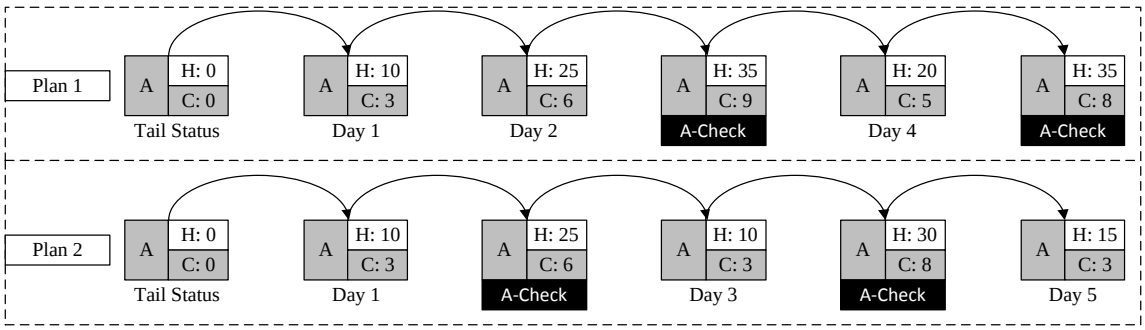


Figure 6.6: Maintenance A-check assignments with recurrence

t. In addition, we now add variables for each of the maintenance checks and their respective counters.

We define a variable c^B_{dtpq} as a continuous variable that tracks the counter of type q for a particular maintenance check p for tail t on the beginning (B) of the day d of the planning horizon. This variable, for example, may index the number of flying-hours (q) for an A-check (p) for a given tail at the start of a given day. This variable is either equal to the previous end-of-the-day counter, or 0 if maintenance was performed. In addition, we have a similar variable that represents the counter status at the end of the day, c^E_{dtpq}. This variable will be equal to the respective beginning of the day variable c^B_{dtpq} plus any counters as a result of the day's assigned LOF. In order to enforce maintenance checks, we require that the value of each counter may never exceed its respective limit, unless a maintenance check is scheduled, which effectively resets the particular counter back to 0.

It should also be noted that in RMP, much like in the Next Maintenance Check Model, the objective function minimizes the total number of maintenance checks assigned. In the MRP and MRP-OS models, maintenance feasible rotations were not guaranteed as we performed as many maintenance events as possible. In addition, some feasible maintenance plans were removed from the possible solution space

in the MRP and MRP-OS that can now be considered. Because we are assigning lines-of-flight and deciding the required maintenance checks, we have a greatly expanded solution space and thus are more likely to find a feasible solution. Given this feasibility, from the airline perspective, we now seek to minimize the total count of maintenance checks that are actually assigned. As such, for the remainder of the mathematical models presented in this chapter, the objective minimizes the total count of maintenance checks that have to be completed.

We now present the connection-based model for RMP. This formulation mimics the connection-based approach presented in Chapter V. We begin by defining the sets, parameters and decision variables we require for this formulation.

Sets

T	The set of tails.
D	The set of days in the planning horizon.
M	The set of maintenance stations in the network.
L	The set of lines-of-flight.
$L(t) \subseteq L$	The set of lines-of-flight that can be flown by tail t, $\forall t \in T$.
P	The set of maintenance checks, i.e. A-check, B-check, etc.
$Q(p)$	The set of maintenance counters, i.e. flying hours, cycles, and calendar days for a particular check p, $\forall p \in P$.
$U_T(l,t) \subseteq L(t)$	The set of lines-of-flight upstream to line-of-flight l allows for fleet type of tail t, $\forall t \in T, \forall l \in L(t)$. In this case, we refer to upstream lines-of-flights as those lines that directly precede line l in terms of departure station and day in the planning horizon.
$D_T(l,t) \subseteq L(t)$	The set of lines-of-flight downstream from line-of-flight l for fleet type of tail t, $\forall t \in T, \forall l \in L(t)$. In this case, we refer to downstream lines-of-flights as those lines that directly follow line l in terms of arrival station and day in the planning horizon.

Parameters

N	A large (big M) numeric constant.
$a(l) \in D$	The day of the planning horizon at which line-of-flight i terminates, $\forall l \in L$.
$s_A(l) \in M$	The specific station at which line-of-flight l terminates, $\forall l \in L$.
$s_D(l) \in M$	The specific station at which line-of-flight l originates, $\forall l \in L$.
γ_p	The required capacity (man-hours) to perform a maintenance check of type p, $\forall p \in P$.
σ_{lp}	The counter values incurred towards the check limit before a maintenance check of type p is required for line-of-flight l, $\forall l \in L, \forall p \in P$.
ρ_{sd}	The available capacity (man-hours) at station s on day d of the planning horizon, $\forall s \in M, \forall d \in D$.
Ω_{pq}	The limit of maintenance check type p for counter q, $\forall p \in P, \forall q \in Q(p)$.

Variables

w_{tdsp}	a binary variable that is 1 if tail t receives maintenance on day d at station s for maintenance check type p, and 0 otherwise, $\forall t \in T, \forall d \in D, \forall s \in M, \forall p \in P$.
c^B_{dtpq}	a continuous variable that represents the accumulation of counter type q for maintenance check p at the *beginning* of day d for tail t, $\forall d \in D, \forall t \in T, \forall p \in P, \forall q \in Q(p)$.
c^E_{dtpq}	a continuous variable that represents the accumulation of counter type q for maintenance check p at the *end* of day d for tail t, $\forall d \in D, \forall t \in T, \forall p \in P, \forall q \in Q(p)$.
x_{ijt}	a binary variable that is 1 if line i is followed by line j on tail t, $\forall t \in T, \forall i \in L(t), \forall j \in D(i,t)$.
y_{it}	a binary variable that is 1 if line i is the first line assigned to tail t during the planning horizon, $\forall t \in T, \forall i \in L(t)$.

Objective:

$$\min \sum_{t \in T} \sum_{d \in D} \sum_{s \in S} \sum_{p \in P} w_{tdsp} \tag{6.14}$$

Subject to:

$$\sum_{t\in T}\sum_{p\in P}\gamma_p w_{tdsp} \le \rho_{sd} \qquad \forall d\in D, \forall s\in M \tag{6.15}$$

$$w_{tdsp} - \sum_{\substack{i\in L(t):\\ s_A(i)=s\\ a(i)=d}} \left(y_{it} + \sum_{j\in D_T(i,t)} x_{ijt} \right) \le 0 \tag{6.16}$$ [1]

$$c^B_{dtpq} - \left(c^E_{d-1tpq} - N\left(\sum_{s\in M} w_{t(d-1)sp} \right) \right) \ge 0 \tag{6.17}$$ [2]

$$c^E_{dtpq} - \left(c^B_{dtpq} + \sum_{\substack{i\in L(t):\\ a(i)=d}} \sigma_{ip} \left(y_{it} + \sum_{j\in D_T(i,t)} x_{ijt} \right) \right) = 0 \tag{6.18}$$ [3]

$$c^E_{dtpq} - \Omega_{pq} \le 0 \tag{6.19}$$ [4]

$$\sum_{j\in U_T(i,t)} x_{jit} + y_{it} - \sum_{j\in D_T(i,t)} x_{ijt} = 0 \qquad \forall t\in T, \forall i\in L(t) \tag{6.20}$$

$$\sum_{i\in L} y_{it} \le 1 \qquad \forall t\in T \tag{6.21}$$

$$\sum_{t\in T} \left(y_{it} + \sum_{j\in U_T(i,t)} x_{jit} \right) = 1 \qquad \forall i\in L \tag{6.22}$$

$$c^B_{0tpq} = 0 \tag{6.23}$$ [5]

$$c^B_{dtpq}, c^E_{dtpq} \ge 0 \tag{6.24}$$ [6]

$$w_{tdsp} \in \{0,1\} \tag{6.25}$$ [7]

$$x_{ijt} \in \{0,1\} \tag{6.26}$$ [8]

$$y_{it} \in \{0,1\} \quad \forall t\in T, \forall i\in L(t) \tag{6.27}$$

[1] $\forall t\in T, \forall d\in D, \forall s\in S, \forall p\in P$

[2] $\forall t\in T, \forall d\in D, \forall p\in P, \forall q\in Q(p)$

[3] $\forall t\in T, \forall d\in D, \forall p\in P, \forall q\in Q(p)$

[4] $\forall t\in T, \forall d\in D, \forall p\in P, \forall q\in Q(p)$

[5]$\forall t \in T, \forall p \in P, \forall q \in Q(p)$

[6]$\forall t \in T, \forall d \in D, \forall p \in P, \forall q \in Q(p)$

[7]$\forall t \in T, \forall d \in D, \forall s \in S, \forall p \in P$

[8]$\forall t \in T, \forall i \in L(t), \forall j \in D(i,t)$

The objective function in equation (6.14) minimizes the total number of maintenance checks that are completed during the planning horizon. Constraint (6.15) ensures that the capacity at a maintenance station, measured in available man-hours, is not exceeded. In constraint (6.16), we ensure that a maintenance check is only assigned if a particular line i actually terminates at a particular maintenance opportunity on a given day d and a maintenance station s. In constraint (6.17), we set the maintenance counter for the beginning of the day c^B_{dtpq} to either the previous day's counter, or 0 if we performed maintenance over the course of the night. Constraint (6.18) performs the summation of the maintenance counter. More specifically, if a particular line-of-flight is covered by a tail t, then the flying hours for this line-of-flight are added to the total count for this particular tail. The last maintenance constraint, constraint (6.19) ensures that each maintenance counter never exceeds the global limit for a particular number of flying hours.

The second set of constraints (6.20 - 6.22) are network constraints. That is, constraint (6.20) requires for each tail that if a line-of-flight was the first or a proceeding line-of-flight, then it is followed by the another or it is the last in the planning horizon. We ensure that each tail can only have a single starting line-of-flight as defined in constraint (6.21). Finally, constraint (6.22) requires each line-of-flight to be assigned exactly once.

Solving the Connection-based Approach

The connection-based approach proved successful at solving the MRP and MRP-OS models in Chapter V. However, to represent recurring maintenance checks, we add a significant number of additional variables (c^B_{dtpq} and c^E_{dtpq}) and constraints (6.17 - 6.19). These additional variables and constraints increase the size of the formulation significantly, such that even short planning horizons, on the order for 2-days, cannot be solved using a traditional MIP solution approach. Thus, an approach that was successful at solving at MRP cannot be used to solve the RMP.

6.3.3 Pure Rotation-based Formulation

Given the fact that the connection-based formulation cannot be solved due to its problem size, we revisit the rotation-based approach that successfully solved the MRP-OS problem in Chapter V. As with other large-scale mathematical solution approaches, the rotation-based (column generation) approach has the advantage of not requiring all variables to be included during the solution process simultaneously.

In the "pure" rotation-based approach, the primary decision variable, x_{tr}, represents the assignment of a rotation r to a tail t. As the name suggests, the particular rotation r is a sequence of lines-of-flight that covers the length of the planning horizon. For recurring maintenance events, we use an additional variable, w^n_{tdc}, to represent the fact that the n^{th} check of type c is completed on a particular day for a particular tail. To formulate the pure rotation-based approach, we introduce the following (additional) notation.

Sets

$N(c)$	The maximum number of checks of a given type c required over a planning horizon, $\forall c \in C$. This is computed by the total days in the horizon divided by shortest maintenance counter.

Parameters

θ_c	The amount of maintenance man-power capacity required for maintenance check c, $\forall c \in C$.
$m(r,d) \in M$	A binary parameter that is 1 if rotation r terminates at station m on day d of the planning horizon, $\forall r \in R, \forall d \in D$.
δ_{lr}	A binary parameter that indicates if line-of-flight l is contained in rotation r, $\forall t \in T, \forall r \in R(t), \forall l \in L(t)$.
ρ_{md}	The available capacity at station m on day d in the planning horizon, $\forall m \in M, \forall d \in D$.
$r(d_e, d_s, t, c, r)$	This parameter is 1 if tail t (to which this rotation will be assigned) last received maintenance on d_s and will require subsequent check by day d_e for this particular rotation r, $\forall d_e, d_s \in D, \forall t \in T, \forall c \in C, \forall r \in R(t)$.

Variables

x_{tr}	a binary variable that is 1 if rotation r is assigned to tail t, $\forall t \in T, \forall r \in R(t)$.
w^n_{tdc}	a binary variable that is 1 if tail t receives the n-th check of type c on day d of the planning horizon, $\forall t \in T, \forall d \in D, \forall c \in C, \forall n \in N$.
v_{tmdc}	a binary variable that is 1 if maintenance on tail t is performed at station m on day d of check type c, $\forall t \in T, \forall m \in M, \forall d \in D, \forall c \in C$.

Objective:

$$\min \sum_{t \in T} \sum_{m \in M} \sum_{d \in D} \sum_{c \in C} v_{tmdc} \tag{6.28}$$

Subject to:

$$\sum_{t\in T}\sum_{r\in R(t)}\delta_{lr}x_{tr}=1 \qquad \forall l\in L \tag{6.29}$$

$$\sum_{r\in R(t)}x_{tr}=1 \qquad \forall t\in T \tag{6.30}$$

$$\sum_{\substack{d_1\in D:\\ d_1\le d}} w^1_{td_1c}-\sum_{r\in R(t)}r(d,0,t,c,r)x_{tr}\ge 0 \tag{6.31}$$ [1]

$$\sum_{\substack{d_2\in D:\\ d_2>d_1\\ d_2\le d}} w^n_{tcd_2}-\left(w^{n-1}_{tcd_1}+\sum_{r\in R(t)}r(d,d_1,t,c,r)x_{tr}-1\right)\ge 0 \tag{6.32}$$ [2]

$$v_{tmdc}-\left(\sum_{n\in N}w^n_{tdc}+\sum_{\substack{r\in R(t)\\ m(r,d)=m}}x_{tr}-1\right)\ge 0 \tag{6.33}$$ [3]

$$\sum_{c\in C}\sum_{t\in T}\theta_c v_{tmdc}-\rho_{md}\le 0 \qquad \forall m\in M,\forall d\in D \tag{6.34}$$

$$x_{tr}\in\{0,1\} \qquad \forall t\in T,\forall r\in R(t) \tag{6.35}$$

$$v_{tmdc}\in\{0,1\} \tag{6.36}$$ [4]

$$w^n_{tdc}\in\{0,1\} \tag{6.37}$$ [5]

[1] $\forall t\in T,\forall d\in D,\forall c\in C$

[2] $\forall t\in T,\forall d\in D,{}^{\forall d_1\in D}_{d_1<d},\forall c\in C,\forall n\in N(c):n>1$

[3] $\forall t\in T,\forall m\in M,\forall d\in D,\forall c\in C$

[4] $\forall t\in T,\forall m\in M,\forall d\in D,\forall c\in C$

[5] $\forall t\in T,\forall n\in N,\forall d\in D,\forall c\in C$

The objective function in equation (6.28) minimizes the total number of maintenance checks that are completed during the planning horizon. In constraint (6.29), we ensure line-of-flight coverage across all days in the planning horizon. Furthermore,

in constraint (6.30) we require that each tail is assigned to a rotation. Constraint (6.31) requires a maintenance check to take place if the rotation assigned requires this event to be completed. For example, if a particular rotation is assigned that requires a maintenance check to be completed on day 2, then the left-hand-side of this constraint ensures that a maintenance check is scheduled anywhere between the beginning of the planning horizon and day 2. Constraint (6.32) is similar to constraint (6.31) as it requires maintenance checks to be completed depending on the rotation selected, in this case, only those maintenance checks since the first event. Next, constraint (6.33) connects the day and station of a maintenance event to the rotation that is selected. Constraint (6.34) enforces station capacity limits. Finally, constraint (6.35) to (6.37) require variable integrality.

Column-Generation-based Solution Approach

We solve the pure rotation-based approach using a column-generation approach. That is, we include a subset of all possible rotations to the problem, solve the master problem and price out any additional rotations and include those with a negative reduced cost. To begin the column generation approach, we first denote the following dual variables for the linear program presented in the previous section.

$$\sum_{t \in T} \sum_{r \in R(t)} \delta_{lr} x_{tr} = 1 \qquad (\delta_l) \tag{6.38}$$

$$\sum_{r \in R} x_{tr} = 1 \qquad (\theta_t) \tag{6.39}$$

$$\sum_{\substack{d_1 \in D: \\ d_1 \leq d}} w^1_{td_1c} - \sum_{r \in R(t)} r(d, 0, t, c, r) x_{tr} \geq 0 \qquad (\xi_{tdc}) \tag{6.40}$$

$$\sum_{\substack{d_2 \in D: \\ d_2 > d_1 \\ d_2 \leq d}} w^n_{tcd} - \left(w^{n-1}_{tcd_1} + \sum_{r \in R(t)} r(d, d_1, t, c, r) x_{tr} - 1 \right) \geq 0 \qquad (\kappa_{tdd_1cn}) \tag{6.41}$$

$$v_{tmdc} - \left(\sum_{n \in N} w^n_{tdc} + \sum_{r \in R(t)} m(r, d) x_{tr} - 1 \right) \geq 0 \qquad (\alpha_{tmdc}) \tag{6.42}$$

Using these dual-variables we can determine the reduced-cost for all additional rotations as follows.

$$\begin{aligned} \bar{c_{rt}} = \quad & 0 - \sum_{l \in L} \delta_{lr} \beta_l - \theta_t + \sum_{d \in D} \sum_{c \in C} r(d, 0, t, c, r) \xi_{tdc} \\ & + \sum_{d \in D} \sum_{d_1 < d} \sum_{c \in C} \sum_{n \in N} r(d, d_1, t, c, r) k_{tdd_1cn} + \sum_{m \in M} \sum_{d \in D} \sum_{c \in C} m(r, d) \alpha_{tmdc} \end{aligned} \tag{6.43}$$

We can solve this particular model using a column-generation approach similar to that presented in Chapter V. We first pre-compute the set of all possible rotations, Ω^P_t. Next, we solve the mathematical model above using a linear-programming solver. Subsequently, we price all rotations in the passive set of additional rotations Ω^P_t and include those with negative reduced cost. Finally, we take the final LP solution of the restricted master problem and branch on the x_{tr} variables to obtain an integer solution.

As we will show in §6.3.5, it turns out that the LP-relaxation of this model is easily solved, while the IP equivalent is significantly more difficult to solve due to the variable branching that occurs on the w^n_{tdc} variables. To address this issue, we now contrast the pure rotation-based approach to one in which the maintenance checks are embedded within the actual rotation. We refer to this approach as the *augmented rotation-based* formulation.

6.3.4 Augmented Rotation-based Formulation

In the pure rotation-based approach, sequences of lines-of-flight are connected and additional variables and constraints are used to enforce maintenance checks. In this augmented rotation-based approach, we build the maintenance checks directly into the rotation. Similar to the pure-rotation-based approach, we use the variable v_{tr} to determine which rotation is assigned to a particular tail t. However, in this case, a rotation consists not only of a set of lines-of-flight, but also includes a set of required maintenance checks, including their respective scheduled day and station, that will be performed if such a rotation is selected. In effect, this variable representation stores more information within a single variable. As a result, our formulation requires a larger number for variables, but the number of constraints is reduced.

Each rotation in the active set Ω^A_t must be maintenance feasible and thus, as argued in the previous two sections, the objective of this optimization problem also minimizes the total number of maintenance checks that must be completed.

Sets

R^A	The set of all augmented rotations.
$R^A(t) \subseteq R^A$	The set of all augmented rotations that can be assigned to tail t, $\forall t \in T$.
C	The set of maintenance checks, e.g. A-Check, B-Check, etc.

Parameters

κ_r	The number of maintenance checks that are contained within rotation r, $\forall r \in R$.
γ_c	The required capacity of maintenance check c, $\forall c \in C$.
ξ_{crsd}	A binary parameter that indicates whether rotation r contains a check of type c at station s on day d of the planning horizon, $\forall c \in C, \forall r \in R^A(t), \forall s \in M, \forall d \in D$.
δ_{lr}	A binary parameter that indicates if line-of-flight l is contained in rotation r, $\forall t \in T, \forall r \in R(t), \forall l \in L(t)$.

Variables

v_{tr}	a binary variable that is 1 if rotation r is assigned to tail t, $\forall t \in T, \forall r \in R(t)$.

Objective:

$$\min \sum_{t \in T} \sum_{r \in R^A(t)} \kappa_r v_{tr} \tag{6.44}$$

Subject to:

$$\sum_{t \in T} \sum_{r \in R^A(t)} \sum_{c \in C} \gamma_c \xi_{crsd} v_{rt} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \tag{6.45}$$

$$\sum_{t \in T} \sum_{r \in R^A(t)} \delta_{lr} v_{tr} = 1 \qquad \forall l \in L \tag{6.46}$$

$$\sum_{r \in R^A(t)} v_{rt} = 1 \qquad \forall t \in T \tag{6.47}$$

$$v_{tr} \in \{0, 1\} \qquad \forall t \in T, \forall r \in R^A(t) \tag{6.48}$$

The objective in equation (6.44) minimizes the total number of maintenance checks performed over the planning horizon. Constraint (6.45) ensures that the capacity of a maintenance block, measured in available man-hours, is not exceeded. Constraint (6.46) requires each line-of-flight over the course of the planning horizon

to be covered. Finally constraint (6.47) ensures that each tail is assigned a particular rotation, even if it is the null rotation.

As discussed in Chapter V, solving the rotation-based formulation is a two-phased approach. First, we solve the master problem followed by a pricing sub-problem. As done in Chapter V, we use a tail-recursive approach to generate the set of rotations a priori. Given the set of rotations, we then apply the reduced-cost equation during the pricing step. The advantage of this approach is the solution speed with which we are able to evaluate the pricing problem and thus move new columns from the passive set (Ω_t^P) to the active set (Ω_t^A) for inclusion during the next iteration of the master problem. This pricing approach however depends on the size of the data instance — above a certain threshold, it will be faster to solve a pricing optimization problem rather than using rotation-enumeration. We now provide an outline of the rotation generation algorithm used in our approach.

Generating Rotations

The rotation algorithm used in our approach depends on several parameters that can be used to control the size and thus the speed of the algorithm. Note that these parameters reflect an airline decision maker's desire for control over the check frequency and capacity utilization. As such, a decision maker should determine the trade-off between the solution speed of the recovery problem and the inclusion of additional maintenance checks during earlier days in the planning horizon. For our particular algorithm, we define the following three parameters that must be set prior to execution.

PERFORM-OPTIONAL-MTN-END

MAX-DAY-EARLY$_c$

SAME-DAY-CHECKS-ALLOWED

The first parameter, PERFORM-OPTIONAL-MTN-END defines whether or not a maintenance check will be scheduled at the end of the planning horizon. If this parameter is set to *true*, then a maintenance event is generated at the end of the planning horizon, even if maintenance is not necessary for the aircraft. This parameter is less concerned with the solution space, but rather the characteristics of the solution. That is, this feature is useful as it resets the maintenance counters in the subsequent planning period and thus can preempt excessive maintenance requirements at the beginning of the following horizon.

In addition, the parameter $\texttt{MAX-DAY-EARLY}_c$ restricts maintenance checks to be performed early by a limited number of days. In other words, suppose that a rotation requires a maintenance check to take place by the end of day 3 in the planning horizon, then this parameter, if set to 1, will allow this maintenance check to be performed at most one day early (i.e. day 2). From an algorithmic standpoint, this parameter effectively creates a copy of a particular rotation, one in which the particular check occurs on day 3, while in the copy, the maintenance check is scheduled for day 2 (if possible). Note that this parameter applies to cascading maintenance checks. That is, if one check is scheduled a day early, the previous check may have to move a day early as well. Furthermore, this parameter is check-type specific. For example, an A-check, which occurs rather frequently, can be restricted to be performed at most one day early. On the other hand, for checks that happen less-frequently, e.g. every 90 days, we may allow such checks to be performed earlier. Such a restriction is typically useful for long-haul lines-of-flight, which rotate through maintenance hubs less frequently.

Finally, we define a parameter SAME-DAY-CHECKS-ALLOWED that controls whether two maintenance checks can be scheduled on the same day. That is, if a rotation commands an A-check and a B-check on the same day, then this parameter can be used to determine if such a condition will be allowed, or if the second-check has to occur a day earlier or later.

To form the set of rotations, we use two algorithms in conjunction. First, in Algorithm (4), we create a new rotation for each tail with a possible lines-of-flight that begins from the tail's starting location at the beginning of the planning horizon. This initial rotation serves as the starting point from which we will spawn additional rotations. As noted in the algorithm, we also initiate a potential A-check and B-check through the newACheck and newBCheck functions. These counters track the LOFs that are assigned in the rotation and trigger the required check when necessary.

Given our starting positions, we now add LOFs over the planning horizon by recursively traversing the flight network of LOFs. This approach is shown in Algorithm (5). In this algorithm, we check whether a check limit has been reached each time a LOF is added to the rotation. If such a limit is reached, we spawn a new rotation (a copy). In one case we insert the maintenance check (if possible) and continue traversing the flight network. In the other case, we follow the spawned copy and attempt to schedule the maintenance check early, for each of the days as defined by $\texttt{MAX-DAY-EARLY}_c$. This process continues until a rotation reaches the end of the planning horizon at which point it is added to the set of passive variables, Ω_t^P.

Example Rotation Generation

In Figure (6.7), we provide a possible rotation to demonstrate our rotation generation algorithm. In this rotation, an aircraft flies five lines-of-flight over the course of

a five day horizon. As noted on the diagram, each line-of-flight has a certain number of hours (H) and cycles (C) associated with it. For simplicity of this example, we ignore the calendar days maintenance counter. Moreover, in this example, each station at the end of the day features the opportunity to perform aircraft maintenance.

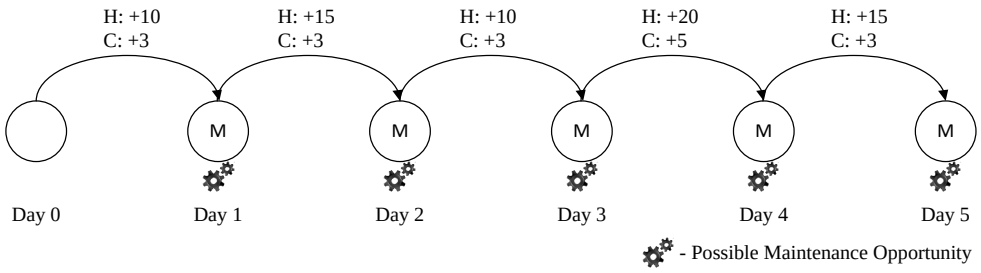


Figure 6.7: Sample conjunction of five lines-of-flight

As defined by our algorithm, we have three parameters that are defined as follows:

```
PERFORM-OPTIONAL-MTN-END = 1

MAX-DAY-EARLY = 1

SAME-DAY-CHECKS-ALLOWED = 0
```

To determine the set of possible rotations for a particular tail, we begin by defining the tail properties as seen in Figure (6.8) below. This indicates that we are building rotations for Tail #1, which just completed an A-Check, and thus its maintenance counters for this check are set to zero. In addition, this tail features 50 flight hours and 30 cycles towards it next B-Check.

In terms of maintenance limits, in the following example, we assume that the aircraft assigned to these rotations will require an A-check at 40 flight hours or 10 cycles, whichever comes first. In addition, we assume that a B-Check must be

performed at 80 hours of flying or 40 cycles, again whichever occurs earlier.

Tail #1		Current	Limit
A-check	Hours (H):	0	40
	Cycles (C):	0	10
B-check	Hours (H):	50	80
	Cycles (C):	30	40

Figure 6.8: Tail information used for generation

Based on our algorithm, we can use the rotation from Figure (6.7) to generate the set of specific recurring maintenance rotations. In Figure (6.9) we provide all four possible routes that can be generated based on this single set of lines-of-flight, each one indicating when a specific maintenance check is scheduled to occur.

Figure 6.9: Set of possible rotations including maintenance

Rotation 1 features the first maintenance check on day 2, as this is the farthest the aircraft can go without a B-check given the characteristics (flying hours and cycles) of the lines-of-flight. In addition, an A-check follows the next day, because of the flight hour and cycle count. In addition, because we set `PERFORM-OPTIONAL-MTN-END = 1`, a maintenance event is assigned on day 5. For rotation 2, given that we are allowed to perform maintenance at most one day early (set by $\texttt{MAX-DAY-EARLY}_B$ `= 1`), the B-check is now scheduled for day 1, while the A-checks remain the same. For rotation 3, not only is the B-check moved one day forward, but now the A-check could also be performed one day earlier than required. However, performing this A-check early requires another A-check to be added on day 4. Finally, in rotation 4, the A-check that was added on day 4 for rotation 3 can also be performed a day earlier. As such, in this rotation, the A-checks follow each other directly. As the application of the problem expands to larger data sets, one could argue that rotation 4, while feasible, performs two successive A-checks, which may be undesirable from a capacity planning perspective and therefore should be eliminated. In our approach, however, we do not eliminate such rotations.

Solving the Augmented Rotation-based Formulation

The augmented rotation-based approach contains a large number of variables, even for relatively short planning horizons. In Figure (6.10), we graphically demonstrate the impact of an additional day of the planning horizon on the number of possible augmented rotations. However, in our solution approach, we will not solve a problem containing all variables simultaneously. Instead, we use a similar approach as to the one posed in Chapter V, whereby we use a column-generation approach to price out only those variables that improve the objective.

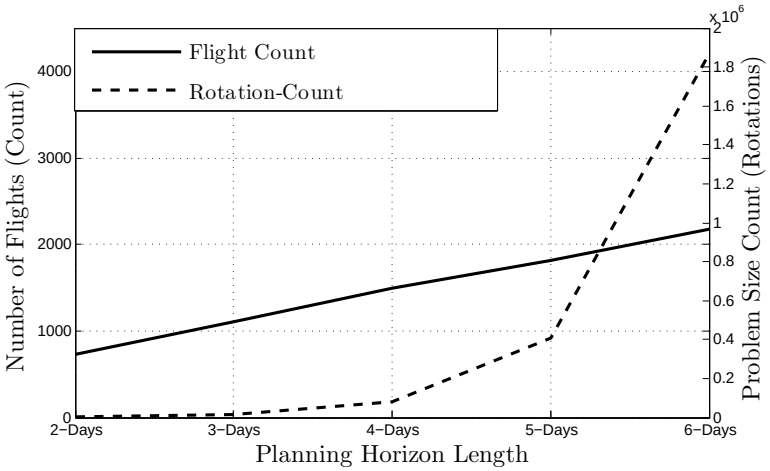


Figure 6.10: Passive-set (Ω_t^P) variable count

We begin by solving the optimization problem with a subset of all possible augmented rotations, the active-set (Ω_t^A). We describe these starting conditions in the next section. Once we solve this particular problem, we can extract the dual-variables for each of the constraints, namely $\alpha_{sd}, \beta_l, \theta_t$ for the respective constraints (6.49) through (6.51) below.

$$\sum_{t \in T} \sum_{r \in R^A(t)} \sum_{c \in C} \gamma_c \xi_{crsd} v_{rt} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \qquad (\alpha_{sd}) \tag{6.49}$$

$$\sum_{t \in T} \sum_{r \in R^A(t)} \delta_{lr} v_{tr} = 1 \qquad \forall l \in L \qquad (\beta_l) \tag{6.50}$$

$$\sum_{r \in R^A(t)} v_{tr} = 1 \qquad \forall t \in T \qquad (\theta_t) \tag{6.51}$$

With these dual-variables, we note the reduced-cost equation that will be used to price each of the additional rotations from passive set. This reduced-cost equation is shown in (6.52). Note that the dual-variable θ_t has an implied 0/1 multiplier. That is, the dual-variable θ_t is only applied if the proposed rotation can actually be flown by tail t for which it is being considered. Otherwise, this variable is not considered in

the reduced-cost calculation. As noted earlier, if a $\overline{c_{rt}}$ is negative, then this variable is included in the active-set and the master problem is resolved. We iterate between solving the optimization problem (master problem) and the pricing problem until we cannot find a rotation that has a negative reduced-cost.

$$\overline{c_{rt}} = \kappa_r - \sum_{c \in C} \sum_{s \in M} \sum_{d \in D} \xi_{crsd} \alpha_{sd} - \sum_{l \in L(t)} \delta_{lr} \beta_l - \theta_t \tag{6.52}$$

Generating Starting Conditions

To begin the column-generation process, we must start with a feasible solution, such that dual-variables can be obtained. To do so, we begin with the set of original rotations that are assigned to each tail. For each of these rotations, we apply our maintenance check insertion algorithm detailed in Algorithm (5). The goal of this approach is to generate the initial set of maintenance feasible augmented rotations that will be included in the restricted master, followed by the pricing problem. However, in certain cases, the initial rotations are not maintenance feasible. In the event that the initial rotation turns out to be infeasible, we then include all possible rotations for this particular tail. For each of these rotations, we apply our maintenance check insertion algorithm to attain maintenance feasible augmented rotations. We include all possible augmented rotations for the particular tail based on this generation Algorithm (4).

Despite this approach, situations can arise where total line-of-flight coverage, constraint (6.50), is not met or the total capacity of a maintenance station is inadequate given the current set of rotation. If it turns out that by using this approach we are

unable to find an initial feasible solution, then we add the original rotation (even though it is maintenance infeasible) with a significant penalty cost to the set of active-variables (Ω_t^A). As the optimization process begins, these rotations provide a significant disincentive to the objective function value and will be excluded as soon as possible. Given this starting condition, we are able to find an initial feasible solution to the problem and then begin the modified column-generation algorithm.

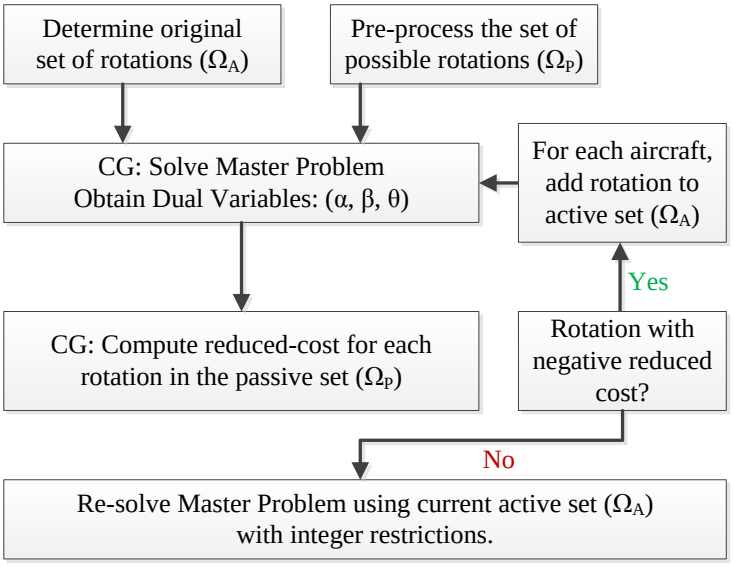


Figure 6.11: Column generation approach using pre-processed rotations

6.3.5 Computational Results

Our models and algorithms are implemented using the C++ programming language with CPLEX as a commercial solver product for linear optimization. System specifications for our computational experiments include an Intel Core i7 processor with a 2.8GHz clock speed and 8GB of main memory. Our implementation is based on serial execution and does not take advantage of multiple processors (or processor cores). However, that the pricing problem, as illustrated in our approach, is an independent problem. More specifically, solving a pricing problem for a particular tail

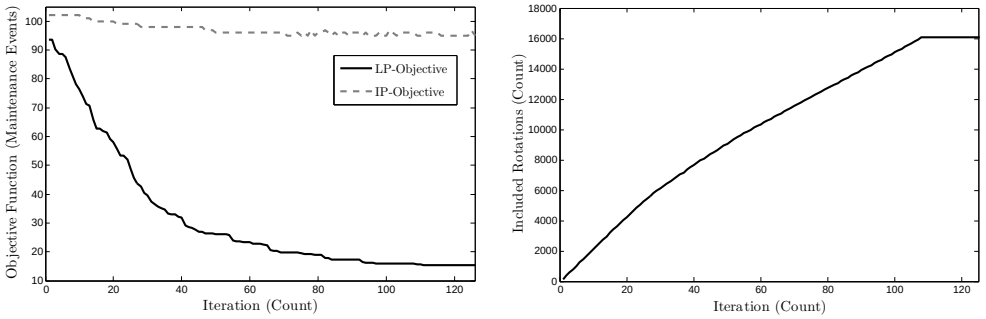
does not involve any cross-tail constraints. As such, solving such a problem lends itself to parallelization for improved computational performance.

We now provide computational results for both the pure rotation-based and augmented rotation-based problems illustrated earlier in this section. Furthermore, we show that only the augmented rotation-based approach provides a solution to the RMP in a reasonable amount of computation time.

Pure Rotation-based Approach Results

The pure rotation-based approach is solved using a column-generation approach in which rotations are pre-processed. These pre-processed rotations are the same set of rotations used in Chapter V. We start the column-generation process with the original set of rotations from the tail assignment. As outlined in Figure (6.11), we continue to price additional rotations with each iteration. The number of rotations added per iteration is a function of the number of tails in the data set. We price one rotation for each tail. As such, for each iteration, a maximum of 71 new rotations (for our data set) can be added to the restricted master problem.

The results from the pure rotation-based approach are shown in Figure (6.12) where we solve the restricted master problem (LP) and for comparison purposes, the corresponding IP. We emphasize that these results are time constrained. That is, each iteration was permitted to run for a maximum of 600 seconds of CPU time. To determine at which iteration the IP solution reaches this time constraint, we solve both the LP and its corresponding IP simultaneously. As seen in Figure (6.13), this time-limit was reached rather quickly, after iteration 41. In other words, after iteration 41, we cannot solve the pure-rotation-based approach in a reasonable amount of time.



(a) Objective function value progression (b) Rotations included in restricted master

Figure 6.12: Solution approach to pure-rotation based model

We make several additional observations about these results. First, when comparing the IP solution to the LP solution, we note a growing divergence in objective function value as the number of iterations increase. In other words, solving the LP relaxation for the pure rotation-based approach is not an effective bound on the total number of maintenance checks that must be completed from the IP solution. Second, the total number of maintenance events required over the 5-day horizon decreases only slightly (from 102 to 95) as more iterations of the column-generation approach are solved.

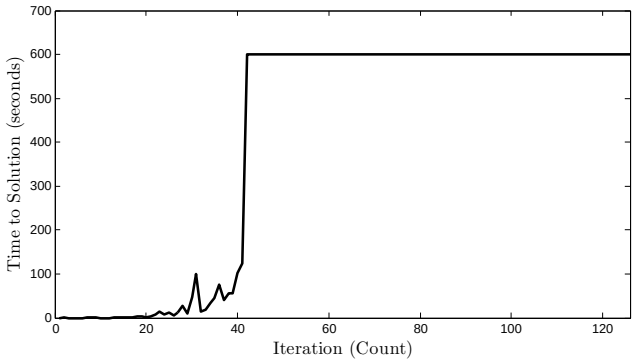


Figure 6.13: Pure rotation-based solution time results for IP solution.

Based on these results, we note that while the pure rotation-based approach can be formulated rather elegantly, it is does not satisfy our overall objective of solving a

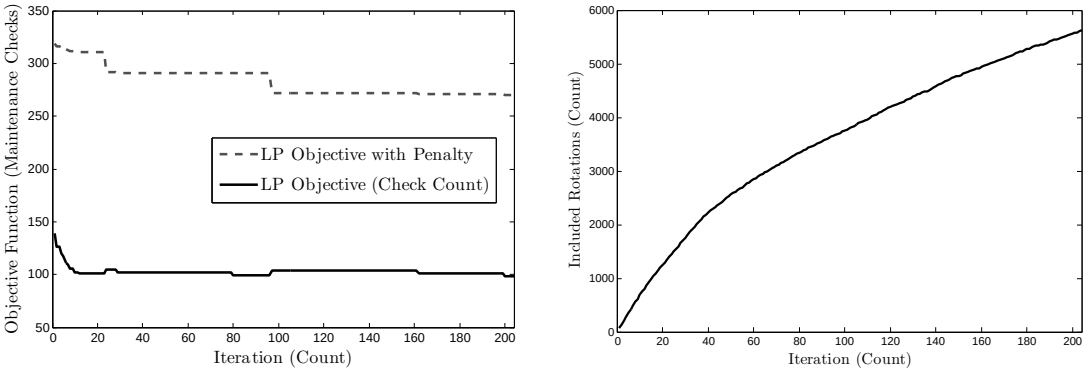
maintenance recovery problem with recurring maintenance events within a relatively short time window. It turns out, however, the augmented rotation-based model, which combines all rotation information into a single variable, is able to solve the recurring maintenance problem rather quickly.

Augmented Rotation-based Approach Results

We use a column-generation-based approach to solve the Recurring Maintenance Problem (RMP) with augmented rotations. Using this approach, we obtain a maintenance feasible recovery plan quickly and can assign this plan to aircraft to complete the tail assignment recovery process. In Figure (6.14), we illustrate the effectiveness of our approach. That is, this figure illustrates the convergence of the algorithm to the optimal solution.

Solving the five-day horizon problem is done over the course of 204 iterations of the modified column-generation solution algorithms. The objective function shown here is the total number of checks that are included in the final set of assigned rotations. Given the fact that some non-feasible rotations are included as part of the solution approach, the objective function can actually *increase* during the column-generation process. That is, we find a rotation in the passive set that, when included, reduces the overall objective function value (by removing an infeasible rotation with a high penalty cost), however, given this rotation the total number of maintenance checks actually increases.

Thus far we solved the maintenance recovery problem with recurring events. Based on our results, the augmented-rotation-based approach appears most successful in solving RMP both from a feasibility and speed-to-solution perspective.



(a) Objective function value progression

(b) Required rotations in solution

Figure 6.14: Solution approach to augmented-rotation-based model

However, the augmented-rotation-based approach is only effective for a 5-day planning horizon after which the number of variables increases such that longer planning horizons cannot be evaluated. We now develop RMP further to include longer planning horizons, on the order for three weeks, which assists airlines in longer-term station capacity planning.

6.4 Rolling Horizon Approach

6.4.1 Algorithm for Solving Longer Planning Horizons

The augmented rotation-based approach presented in §6.3.4 solves the recurring maintenance problem for a five-day horizon. To perform long-term maintenance recovery and understand capacity utilization patterns, airlines prefer a solution to the maintenance recovery problem for time horizons on the order of multiple weeks. Such a solution provides a prediction on the long-term capacity utilization at maintenance bases. This length of time frame, however, will lead to a problem size that cannot be solved using our augmented rotation, column-generation approach, because the number of rotations would increase exponentially, nor would an optimization-based pricing problem approach likely converge to a solution within a reasonable amount of time.

In this section, we propose an alternative algorithm that can be used to solve the recurrent maintenance recovery problem over longer time horizons. Our rolling horizon approach can be used to solve an arbitrary length horizon problem, while only increasing linearly in the amount of time required to obtain a solution. We combine the 5-day augmented rotation-based approach from §6.3.4 with an iterative (waterfall) method. This is approach illustrated in Figure (6.15) below.

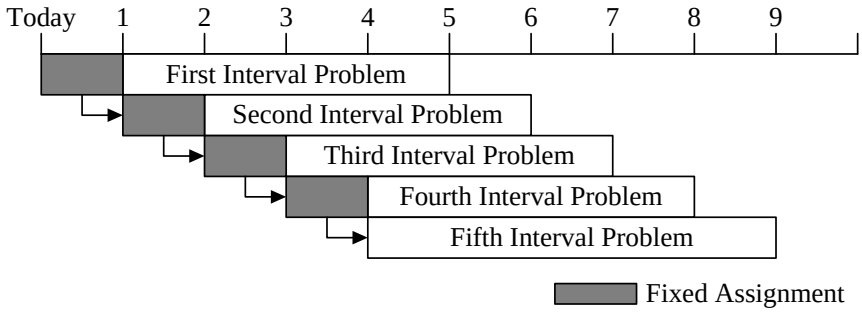


Figure 6.15: Rolling horizon overview

As discussed in [9], the rolling horizon approach is commonly used during long-horizon and infinite horizon models. In their work, the authors successful bound the optimality gap of using the rolling horizon approach for a non-homogeneous Markov decision process. One of the earlier examples of solving long-term horizon scheduling problems comes from [12]. In this work, the authors focus on production planning problems and demonstrate that a rolling horizon approach offers an efficient solution for a long-term planning problem.

In our approach, the first interval problem is solved using the augmented rotation-based algorithm. Once the first interval problem (days 1 through 5) is solved to optimality, it provides the input for the second interval problem. That is, the solution to the first interval problem fixes the lines-of-flight and any corresponding maintenance checks that are assigned on the first day. Thus, the second interval problem receives input in the form of updated maintenance counters and aircraft locations from the solution of the first interval problem. This process then repeats whereby the second interval problem provides input for the third interval problem and so on.

Our complete solution algorithm, graphically depicted in Figure (6.16), continues and solves each of the interval problems given the input from the previous interval. In each iteration another day's lines-of-flights are fixed and maintenance checks are set. This process can be repeated to solve time horizons of arbitrary length and will only require another iteration of the augmented rotation-based algorithm for each day that is added.

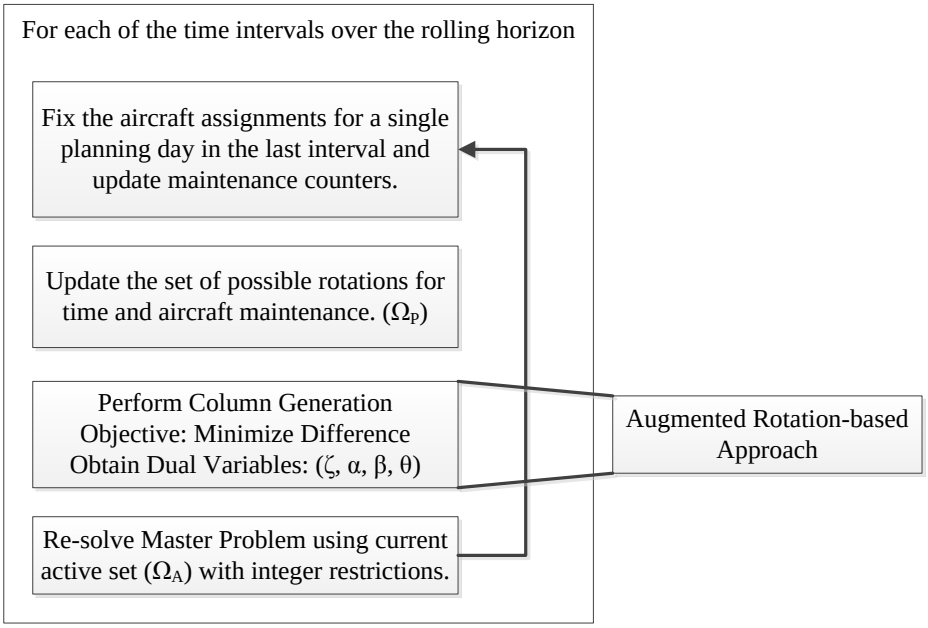


Figure 6.16: Rolling horizon algorithm overview

6.4.2 Alternate Objective Function

We solve each of the interval problems using the augmented rotation-based approach. The objective function of this algorithm minimizes the total number of maintenance checks over the specified time interval. As we will demonstrate later in this chapter in our computational results, minimizing the total number of maintenance checks across each time interval creates significant variation in station man-power utilization from one solution iteration to the next. From an airline's perspective, large fluctuations in the assigned maintenance man-power causes additional difficulties for station capacity planning. In this section, similar to the even maintenance problem in MRP in Chapter V, we optimize for a relatively stable station utilization over the extended planning horizon.

Instead of minimizing the total number of checks, we now seek a solution that stabilizes maintenance capacity utilization at each station within the network. More specifically, we minimize the total capacity deviation that occurs at a station on vari-

ous days in the planning horizon. We implement this stabilization by first solving the augmented rotation-based approach for a 5-day planning horizon with the objective to minimize the total number of maintenance checks. From this solution, we obtain the *daily* average station utilization for this 5-day horizon. Subsequently, for each of the interval problems, we change the objective to minimize the capacity deviation from this daily average.

In each of the interval problems, we minimize the difference between assigned capacity (through rotations that contain maintenance checks) and the daily station average as computed during the first interval problem. To achieve this objective, we introduce a new variable y_{sd} which indicates the absolute difference between the amount of maintenance, measured in man-hours, that is assigned to station s on day d and the average that was assigned to this station/day during the first time interval, represented by ω_s. The additional input parameters and variables are provided in the table below.

Parameters

ω_s A continuous parameter that represents the average daily amount of capacity allocated during the first interval problem to station s, $\forall s \in M$. This parameter becomes available once the optimal solution to the first interval problem is reached.

Variables

y_{sd} a continuous variable that represents the man-hour capacity difference between the parameter ω_s and the amount currently assigned to station s on day d of the planning horizon, $\forall s \in M, \forall d \in D$.

Objective:

$$\min \sum_{s \in M} \sum_{d \in D} y_{sd} \tag{6.53}$$

Subject to:

(6.54)
$$y_{sd} - \left(\omega_s - \sum_{c \in C} \sum_{t \in T} \sum_{r \in R^A(t)} \gamma_c \xi_{crsd} v_{tr} \right) \geq 0 \qquad \forall s \in M, \forall d \in D \qquad (\nu^P_{sd})$$

(6.55)
$$y_{sd} - \left(\sum_{c \in C} \sum_{t \in T} \sum_{r \in R^A(t)} \gamma_c \xi_{crsd} v_{tr} - \omega_s \right) \geq 0 \qquad \forall s \in M, \forall d \in D \qquad (\nu^N_{sd})$$

(6.56)
$$\sum_{t \in T} \sum_{r \in R^A(t)} \sum_{c \in C} \gamma_c \xi_{crsd} v_{tr} \leq \rho_{sd} \qquad \forall s \in M, \forall d \in D \qquad (\alpha_{sd})$$

(6.57)
$$\sum_{t \in T} \sum_{r \in R^A(t)} \delta_{lr} v_{tr} = 1 \qquad \forall l \in L \qquad (\beta_l)$$

(6.58)
$$\sum_{r \in R^A(t)} v_{tr} = 1 \qquad \forall t \in T \qquad (\theta_t)$$

(6.59)
$$v_{tr} \in \{0, 1\} \qquad \forall t \in T, \forall r \in R^A(t)$$

The formulation of this problem is used for every interval problem starting with the second and is similar to the rotation-based maintenance recovery problem, but with two added constraints (6.54) and (6.55). These constraints form an absolute value constraint for the capacity difference that is assigned to station s on day d of the planning horizon with the station average, ω_s. This difference is also the target of the objective function shown in equation (6.53).

Given this altered objective function and additional constraints, the reduced-cost equation employed during the pricing state of our algorithm has also changed. As seen in equation (6.60) below, the reduced cost equation for a new rotation includes two additional terms for the positive and negative difference dual (ν^P_{sd} and ν^N_{sd}).

$$\overline{c_{rt}} = 0 - \sum_{c \in C} \sum_{s \in M} \sum_{d \in D} \xi_{crsd} \left(\nu_{sd}^{N} + \nu_{sd}^{P} \right) - \sum_{c \in C} \sum_{s \in M} \sum_{d \in D} \xi_{crsd} \alpha_{sd} - \sum_{l \in L} \delta_{lr} \beta_{l} - \theta_{t} \tag{6.60}$$

Using this new reduced-cost equation, we are able to solve the rolling-horizon problem using the same solution algorithm as described in §6.3.4. In the next section we present the computational results from this approach.

6.4.3 Computational Results

In this section, we solve the previously-described algorithm and realize the impact of the rolling-horizon recurring maintenance recovery problem. For each horizon, we obtain a total of five days worth of maintenance capacity assignments. In Table (6.1) we display the results from solving the rolling horizon approach using the objective of minimizing the total number of maintenance checks during each iteration. To contrast, in Table (6.2) we show the maintenance check assignment when all but the first interval problem are solved using the alternative objective approach described in §6.4.2.

In both result tables, the rows refer to the different time intervals that are solved. For example, Interval 1-5 represents the initial five-day horizon, while Interval 2-6 represents the next set of days 2 through 6, etc. For each day in each interval, the daily capacity utilization is reported. For example in Table (6.2), for day 2 of Interval 0, the total amount of maintenance capacity required is 90.4 man-hours. The first number of any 5-day sequence is the actual capacity allocated for that particular day. For example, the capacity allocation on day 1 of Interval 1-5 (man-hour allocation of 139) is fixed when the optimization is performed for days 2 through 6. Likewise,

the capacity utilization of 82.6 on day 2 is fixed once the optimization proceeds to Interval 3-7 of the rolling horizon.

In addition, both tables show the standard deviation (σ), as well as the actual assignment for a particular day of the planning horizon. The standard deviation is indicative of the changes in the expected man-hour workload on a particular day. A large standard deviation indicates that between the five interval problems that covered a particular day, the assigned man-hours changed more-so than if the standard deviation was small.

Int.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1-5	139	90.4	95.2	83	65.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2-6	0	79.8	65.8	85.4	58.8	54.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3-7	0	0	84	88.2	85.4	89.6	71.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4-8	0	0	0	81.2	61.6	82.6	54.6	50.4	0	0	0	0	0	0	0	0	0	0	0	0	0
5-9	0	0	0	0	78.4	64.4	74.2	61.6	42	0	0	0	0	0	0	0	0	0	0	0	0
6-10	0	0	0	0	0	77	56	81.2	54.6	50.8	0	0	0	0	0	0	0	0	0	0	0
7-11	0	0	0	0	0	0	74.2	54.6	74.2	50.4	37.8	0	0	0	0	0	0	0	0	0	0
8-12	0	0	0	0	0	0	0	81.2	57.4	75.6	56	43.4	0	0	0	0	0	0	0	0	0
9-13	0	0	0	0	0	0	0	0	71.4	61.6	75.6	60.2	43.4	0	0	0	0	0	0	0	0
10-14	0	0	0	0	0	0	0	0	0	84	57.4	82.6	60.2	51.8	0	0	0	0	0	0	0
11-15	0	0	0	0	0	0	0	0	0	0	82.6	88.2	82.6	85.4	74.2	0	0	0	0	0	0
12-16	0	0	0	0	0	0	0	0	0	0	0	81.2	56	78.4	56	43.4	0	0	0	0	0
13-17	0	0	0	0	0	0	0	0	0	0	0	0	84	84	89.6	79.8	78.4	0	0	0	0
14-18	0	0	0	0	0	0	0	0	0	0	0	0	0	84	89.6	79.8	93.8	74.2	0	0	0
15-19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85.4	49	86.8	50.4	44.8	0	0
16-20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78.4	93.8	85.4	91	78.4	0
17-21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	91	63	81.2	71.4	54.6
σ	0	7.5	14.84	3.04	11.42	14.1	9.92	14.62	13.15	14.99	17.7	18.8	17.62	14.19	14.3	18.27	6.47	15.01	24.35	4.95	0
Act.	139	79.8	84	81.2	78.4	77	74.2	81.2	71.4	84	82.6	81.2	84	84	85.4	78.4	91	63	81.2	71.4	54.6

Table 6.1: Minimizing maintenance events across all intervals

Int.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1-5	139	90.4	95.2	83	65.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2-6	0	82.6	79.8	86.8	77	68.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3-7	0	0	81.2	79.8	82.6	81.2	61.6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4-8	0	0	0	81.2	84	88.2	77	67.2	0	0	0	0	0	0	0	0	0	0	0	0	0
5-9	0	0	0	0	82.6	81.2	79.8	78.4	64.4	0	0	0	0	0	0	0	0	0	0	0	0
6-10	0	0	0	0	0	82.6	75.6	84	79.8	70.4	0	0	0	0	0	0	0	0	0	0	0
7-11	0	0	0	0	0	0	79.8	78.4	81.2	78.4	64.4	0	0	0	0	0	0	0	0	0	0
8-12	0	0	0	0	0	0	0	84	78.4	81.2	75.6	67.2	0	0	0	0	0	0	0	0	0
9-13	0	0	0	0	0	0	0	0	81.2	79.8	79.8	85.4	67.2	0	0	0	0	0	0	0	0
10-14	0	0	0	0	0	0	0	0	0	85.4	75.6	86.8	77	67.2	0	0	0	0	0	0	0
11-15	0	0	0	0	0	0	0	0	0	0	77	77	82.6	72.8	61.6	0	0	0	0	0	0
12-16	0	0	0	0	0	0	0	0	0	0	0	85.4	72.8	82.6	74.2	65.8	0	0	0	0	0
13-17	0	0	0	0	0	0	0	0	0	0	0	0	84	81.2	89.6	77	75.6	0	0	0	0
14-18	0	0	0	0	0	0	0	0	0	0	0	0	0	82.6	81.2	78.4	82.6	67.2	0	0	0
15-19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86.8	71.4	89.6	77	65.8	0	0
16-20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	75.6	85.4	82.6	78.4	71.4	0
17-21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	92.4	77	86.8	85.4	68.6
σ	0	5.52	8.52	3.04	7.54	7.18	7.58	6.86	7.14	5.5	5.89	8.32	6.96	6.96	11.22	5.11	6.53	6.41	10.57	9.9	0
Act.	139	82.6	81.2	81.2	82.6	82.6	79.8	84	81.2	85.4	77	85.4	84	82.6	86.8	75.6	92.4	77	86.8	85.4	68.6

Table 6.2: Minimize deviation from station capacity utilization

For our data instance, the solution time varies across the various interval problems. As seen in Figure (6.17), some iterations solve more quickly than others, but none of them exceed the solution time of the augmented rotation-based solution approach. In some cases, by fixing the LOFs for the previous day, the next interval problem has fewer choices as to where to place a maintenance check and as such, the problems can solve quicker. In other words, because some maintenance checks are already fixed, fewer decisions are available leading to a faster solution time in subsequent time intervals.

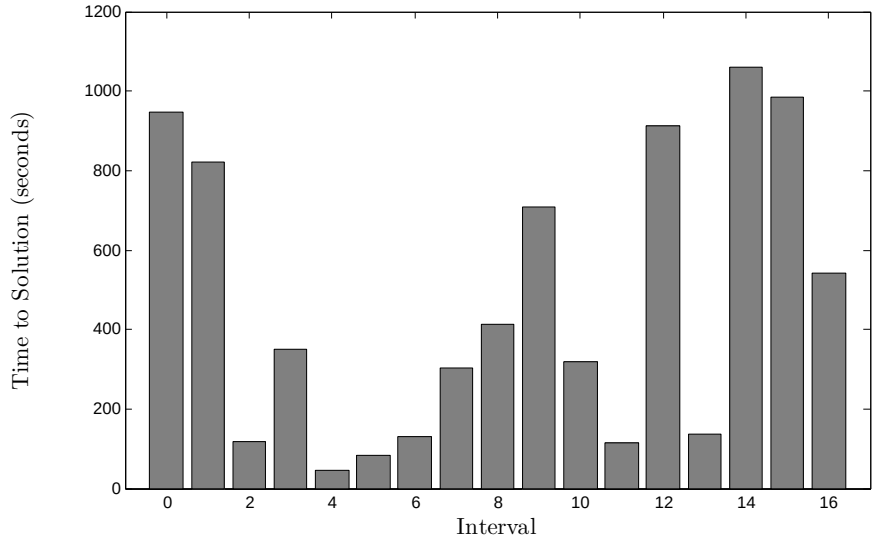


Figure 6.17: Interval solution time for rolling horizon approach for minimizing maintenance checks

To compare the objective of minimizing the total number of maintenance checks and minimizing the deviation of capacity, we provide a graphical comparison in Figure (6.18). As is evident from the figure, the standard deviation between the allocated man-hour capacity is significantly higher when minimizing the total number of checks during each interval problem. In contrast, in Figure (6.19) we illustrate the total number of maintenance checks completed during each time interval. We only incur a slight increase when changing the objective to minimize station capacity

deviation. This result demonstrates the trade-off that exists between minimizing the number of maintenance checks (higher standard deviation) and minimizing the capacity deviation (higher number of maintenance checks).

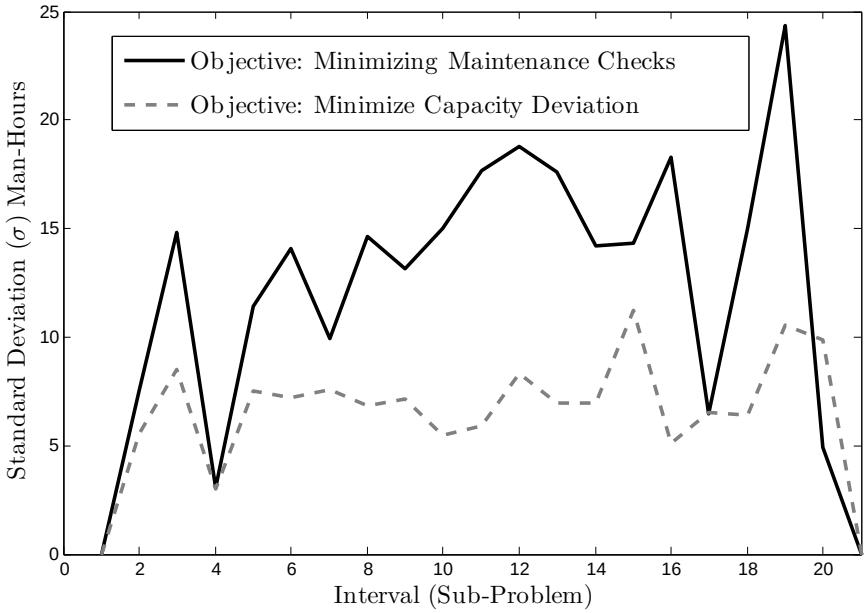


Figure 6.18: Rolling horizon standard deviation comparison

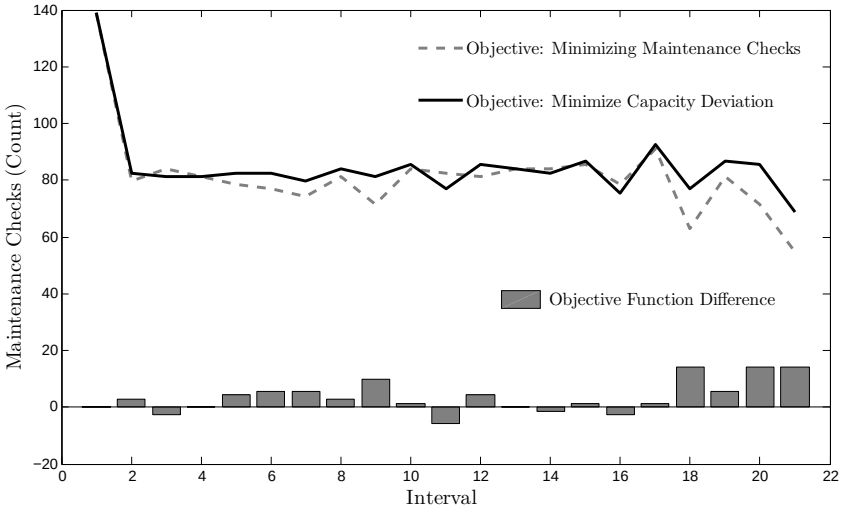


Figure 6.19: Rolling horizon objective function comparison

We realize that this approach does not provide a truly optimal solution to the

recurring maintenance problem over a longer time horizon, because we make myopic (5-day) decisions and subsequently fix the first day in the horizon. We then roll the time horizon one iteration forward and add-on a subsequent day. While the globally optimal solution would encompass a complete solution for the entire planning horizon, we believe that the rolling horizon approach is not a significant shortcoming, for several reasons.

First, when solving the single 5-day horizon model with an objective function of minimizing the total number of maintenance checks, we incur an "end-of-horizon" effect. That is, when performing aircraft maintenance, if a maintenance check can be pushed to day-6 of the planning, then this check will not be included in the solution as this minimizes the total events performed during the five day horizon. However, this myoptic optimal solution ignores the fact that once the day-of-operations occurs, maintenance plans will need to include all checks that were "pushed" beyond the five day horizon. The rolling horizon approach solves this problem. More specifically, once the second interval problem is solved to include day 6, all checks will be re-evaluated and spread across both days 5 and 6 or perhaps even earlier. Furthermore, if the objective function is to minimize the deviation from daily capacity utilization, maintenance checks will not be pushed into furture time horizons. Arguably, this makes the rolling horizon approach useful for long-term maintenance capacity planning. We discuss this extension in §6.5.

Another argument as to why our rolling horizon model still provides a valuable solution is the fact that the airline industry incurs a large number of planning changes daily. For example, airline experts routinely find that 60% or more of planning tail assignments are swapped every day. As such, if the distant horizon is not globally

optimal, we argue that this is not of significant importance as airline plans will most likely change in the interim. However, it is still worth planning for this time frame to assess potential capacity utilization and ensure overall maintenance feasible plans.

6.5 Conclusions & Future Work

In this chapter we extended the maintenance recovery problem (MRP) in three ways. First, instead of pre-processing maintenance events, maintenance checks are now considered part of the decision space and assigned based on the underlying rotation that has been built. Second, instead of considering only the next check of a particular type, we now include the underlying recurrence of maintenance checks. Finally, in addition to considering recurrence, we also include an extended (or rolling) horizon model to solve maintenance recovery problems that exceed a 5-day recovery horizon.

As we have shown, each of these problems can be effectively solved using a column-generation approach in which the rotations are augmented to include maintenance checks within each actual rotation. Furthermore, we can solve the rolling horizon model using two different objectives to assist in the maintenance man-power planning process, which illustrates an extension to this particular problem.

The output of the rolling horizon model can be used to perform the actual maintenance man-power planning process for a long-term horizon for recurring checks. More specifically, as we solve the rolling horizon model, we realize the work that is performed by all aircraft over the course of the planning period. These line-of-flight assignments imply the checks that must be realized, much like the maintenance recovery problem. However, if solved for a longer horizon, the maintenance capacity

required at each station can be noted, as seen in Figure (6.20). Based on these requirements, maintenance man-power can be accurately predicted despite changing airline plans. This is because our rolling horizon approach reveals the cyclical nature of maintenance checks. That is, we emphasize less the *specific* tail that rotates through a maintenance station, but rather that *a* tail rotates through a maintenance station.

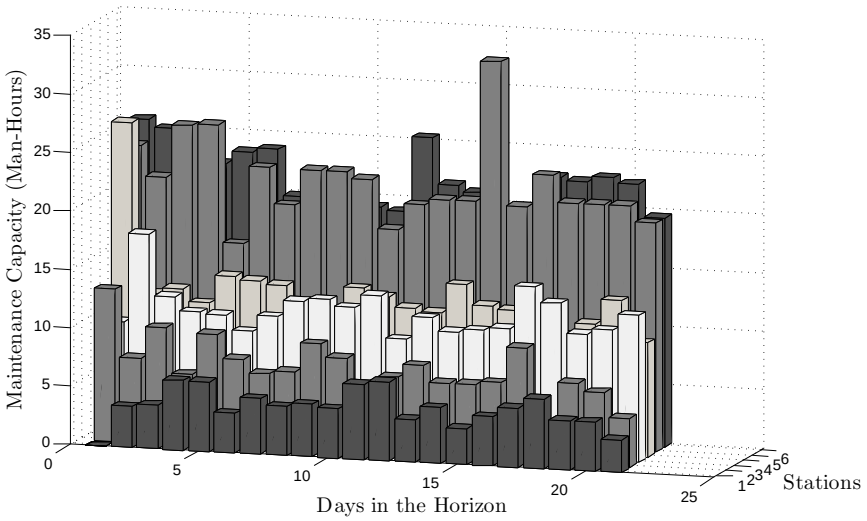


Figure 6.20: Maintenance capacity planning based on rolling horizon approach

CHAPTER VII

Conclusions & Future Work

In this dissertation, we developed new methods for improving robustness and recovery in aviation planning. These problems, sourced from the airline industry, featured a progression beginning with robust flight planning to robust maintenance planning and finally to maintenance recovery.

We first examined the airline planning process in great detail. This background knowledge was required to understand the interconnectedness of airline processes, which set the foundation for the remaining chapters.

In Chapter III we looked at airline schedule design and the importance of interconnectedness among resources, such as aircraft, crew and passengers. In this chapter we developed a simulation model that when paired with the mathematical optimization model in [8] evaluates airline schedule robustness by analyzing the allocation of slack-time within a flight network.

One way to extend this simulation further is to incorporate recovery actions. Depending on the depth and severity of the delay, airlines generally do intervene manually to remove excessive delay propagation. While, such decisions can be incorporated into the simulation model to make it more realistic, it is difficult to determine

the set of possible actions, as each airline has its own set of policies to mitigate delay. Understanding and building a set of recovery actions can improve the results of this simulation framework further.

Next, in Chapter IV, we extended the idea of robustness in airline schedules to the process of maintenance planning. More specifically, we presented a new metric, *maintenance reachability (MR)*, which measures the robustness of a planned set of lines-of-flight, and developed a mathematical programming approach to improving the MR of a given set of lines-of-flight. Through this approach, we demonstrated improvements in maintenance robustness that can be achieved by reallocating lines-of-flight to aircraft, such that the possibility of reaching a maintenance station when required is maximized.

We can extend this approach to capture additional complexities that may arise at a particular airline. First, in our approach, we do not consider maintenance capacity constraints, however, each station in the flight network features a finite amount of man-power capacity to perform maintenance. Our model could be extended to include a capacity constraint, which would then enforce such a restriction. A further extension involves expanding the set of line-splicing opportunities. In our approach, we only looked at simple splices, but deeply nested splicing opportunities could provide additional robustness, especially for airlines which do not feature a comprehensive set of intersecting lines-of-flight.

In Chapters V and VI we transitioned from maintenance planning to maintenance recovery. In this case, we investigated airline maintenance recovery strategies. In Chapter V, we focused on maximizing maintenance coverage, i.e. performing the maximum number of maintenance events possible, assuming fixed, pre-determined

rotations. We later relaxed the pre-defined rotations by allowing overnight swaps, which significantly improved maintenance event coverage.

Chapter VI extended our maintenance recovery problem further by including recurring maintenance checks, i.e. checks where scheduling the first check has implications on the deadline of all future events. In addition, Chapter VI also focused on solving the maintenance recovery problem for an extended horizon, such as a three week time horizon. Solving such an extended horizon provided insights not only into the iterative solution approach, but was also informative for the ground maintenance capacity planning problem.

As noted in Chapter VI, the extended horizon maintenance recovery problem solution approach can also be implemented for maintenance capacity planning problems. More specifically, when we solve the extended horizon problem, we forecast future demand for all maintenance checks and aircraft will undergo in the near-term. Due to the cyclic nature of flight schedules, once all immediate checks are covered, the extended horizon problem provides information into the future man-power requirements at various stations through the flight network. This information, in turn, can be used to develop work schedules for man-power planning departments with respect to maintenance capacity.

In addition to examining and improving the maintenance planning and recovery process, the contributions of this dissertation also featured an in-depth analysis of several mathematical modeling approaches and proof of their structural equivalence. Furthermore, we analyzed several decomposition approaches and developed solution algorithms to solve computationally difficult problems. This insight allowed us to select the most appropriate formulation for a particular problem structure.

We believe that this dissertation has provided insight not only into the aviation maintenance process, but also into algorithms and approaches that can be used to solve recurring event planning problems in general, and that these algorithms and approaches provide insights for scheduling problems in other problem domains.

APPENDICES

APPENDIX A

Maintenance LOF Assignment

A.1 Discrete Convexity Proof

Theorem A.1. *The day-6 aircraft expectation coefficients used in the objective function are discrete-convex when the number of LOFs are equal for each station.*

Proof. To show convexity of the objective function coefficients, a discrete function, we must show that:

$$f(n-1) + f(n+1) \geq 2f(n)$$

We begin with the probability coefficient as seen in the equation below.

$$f(n) = \sum_{i=n+1}^{L_s} \binom{L_s}{i} \left(\frac{1}{7}\right)^i \left(\frac{6}{7}\right)^{L_s - i} (i-n)$$

Replacing the left-hand side, we obtain the following expression:

$$\begin{aligned} \binom{L_s}{i-1} \cdots \times (1) + \binom{L_s}{i} \cdots \times (2) + \binom{L_s}{i+1} \cdots \times (3) + \cdots + \binom{L_s}{L_s} \cdots \times (L_s - n - 1) \\ \binom{L_s}{i+1} \cdots \times (1) + \cdots + \binom{L_s}{L_s} \cdots \times (L_s - n + 1) \end{aligned}$$

Combining terms from the equation above yields the equation below.

$$\binom{L_s}{i-1} \cdots \times (1) + \binom{L_s}{i} \cdots \times (2) + \binom{L_s}{i+1} \cdots \times (3+1) + \cdots + \binom{L_s}{L_s} \cdots \times ((L_s - n + 1) + (L_s - n - 1))$$

Now, replacing the right-hand side, we obtain the following expression:

$$2\left[\binom{L_s}{i} \cdots \times (1) + \binom{L_s}{i+1} \cdots \times (2) + \cdots + \binom{L_s}{L_s} \cdots \times (L_s - n)\right]$$

Combining the equations above, eliminating common terms and re-writing, we obtain the following expression

$$\binom{L_s}{i-1} \left(\frac{1}{7}\right)^{i-1} \left(\frac{6}{7}\right)^{L_s - i - 1} \times (1) \geq 0$$

which is trivially true. From this, we have shown that our probability coefficient function is indeed discrete-convex. □

A.2 Equal MLOF Assignment Proof

Corollary A.2. *Given a situation where n MLOFs are to be assigned to m stations with the fact that n is evenly divisible by m, then the optimal allocation is one where an equal number of MLOFs is assigned to each station.*

Proof. We will prove this corollary by contradiction. Given an assignment whereby all MLOFs outlets are equally distributed, then changing this assignment will result in a *lower* number of day-6 aircraft that will be unable to reach maintenance. In other words, changing a station by increasing its number of MLOFs by one and therefore decreasing another station by one will result in a lower number of expected day-6 aircraft requiring maintenance.

In mathematical terms:

$$f(n+1) + f(n-1) \leq f(n) + f(n)$$

This result is clearly false as it contradicts the convexity argument in the previous proof. Therefore, it must be the case that the optimal allocation is one where an equal number of MLOFs are assigned to each station. □

APPENDIX B

Lagrangian Equivalence Proof

B.1 One-Constraint Lagrangian Duality

We begin with a standard linear-program.

$$\begin{aligned} &\min y \\ &\text{s.t.} \quad Ax = b \end{aligned}$$

We relax the constraint $y = gx$ and move it into the objective function with a penalty coefficient λ.

$$\max_{\lambda} \left[\begin{aligned} &\min y - \lambda \left[y - gx \right] \\ &\text{s.t.} \quad Ax = b \end{aligned} \right]$$

We recombine the terms in the objective function as follows.

$$\max_{\lambda} \left[\begin{aligned} &\min (1 - \lambda) y - \lambda g x \\ &\text{s.t.} \quad Ax = b \end{aligned} \right]$$

We can evaluate this objective under two conditions of λ.

$$\lambda = \begin{cases} (1-\lambda) < 0, y \to \infty, \text{obj} \to -\infty, & \therefore \lambda \leq 1 \\ (1-\lambda) > 0, y \to -\infty, \text{obj} \to -\infty & \therefore \lambda \geq 1 \end{cases}$$

Based on this result, $\lambda = 1$ and thus the optimization problem posed is equivalent to the original problem.

$$\begin{aligned} & \min gx \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

B.2 Two-Constraints Lagrangian Duality

We now consider a standard linear-program with two constraints that define the decision variable y.

$$\begin{aligned} & \min y \\ \text{s.t.} \quad & Ax = b \\ & y \geq gx \\ & y \geq hx \end{aligned}$$

We can again apply the Lagrangian relaxation to this problem, dualize both constraints that pertain to the variable y and add two respective penalty coefficient to the objective.

$$\max_{\lambda_1, \lambda_2} \left[\begin{array}{l} \min y - \lambda_1[y - gx] - \lambda_2[y - hx] \\ \text{s.t.} \quad Ax = b \end{array} \right]$$

We recombine the terms of the objective function as follows.

$$\max_{\lambda_1,\lambda_2} \left[\begin{array}{l} \min(1-\lambda_1-\lambda_2)y - \lambda_1 gx - \lambda_2 hx \\ \text{s.t.} \quad Ax = b \end{array} \right]$$

We can evaluate this objective under two conditions for λ_1 and λ_2.

$$\lambda_1 + \lambda_2 = \begin{cases} (1-\lambda_1-\lambda_2) > 0, y \to \infty, \text{obj} \to -\infty, & \therefore \lambda_1 + \lambda_2 \geq 1 \\ (1-\lambda_1-\lambda_2) < 0, y \to \infty, \text{obj} \to -\infty & \therefore \lambda_1 + \lambda_2 \leq 1 \end{cases}$$

By this derivation, the following relationship between the penalty coefficients must hold: $\lambda_1 + \lambda_2 = 1$. We can rewrite the objective as follows.

$$\max_{\lambda_1,\lambda_2} \left[\begin{array}{l} \min \lambda_1 gx + \lambda_2 hx \\ \text{s.t.} \quad Ax = b \end{array} \right]$$

In this maximization problem, either λ_1 or λ_2 will be 1 and the other respective variable will be set to 0.

B.3 Two-Variable, Two-Constraints Lagrangian Duality

We now consider a standard linear-program with two constraints and two variables. Each constraint defines a particular variable.

$$\begin{array}{rl} & \min y_1 + y_2 \\ \text{s.t.} & Ax = b \\ & y_1 \geq gx \\ & y_2 \geq hx \end{array}$$

We can again apply the Lagrangian relaxation to this problem, dualize both constraints for each corresponding variable and add two respective penalty coefficient to the objective.

$$\max_{\lambda_1,\lambda_2} \left[\begin{array}{l} \min y_1 + y_2 - \lambda_1[y_1 - gx] - \lambda_2[y_2 - hx] \\ \text{s.t.} \quad Ax = b \end{array} \right]$$

We recombine the terms of the objective function as follows.

$$\max_{\lambda_1,\lambda_2} \left[\begin{array}{l} \min(1-\lambda_1)y_1 + (1-\lambda_2)y_2 + \lambda_1 gx + \lambda_2 hx \\ \text{s.t.} \quad Ax = b \end{array} \right]$$

We can evaluate this objective under several conditions.

$$\lambda_1 = \begin{cases} (1-\lambda_1) < 0, y_1 \to \infty, \text{obj} \to -\infty, & \therefore \lambda \leq 1 \\ (1-\lambda_1) > 0, y_1 \to -\infty, \text{obj} \to -\infty & \therefore \lambda \geq 1 \end{cases}$$

$$\lambda_2 = \begin{cases} (1-\lambda_2) < 0, y_2 \to \infty, \text{obj} \to -\infty, & \therefore \lambda \leq 1 \\ (1-\lambda_2) > 0, y_2 \to -\infty, \text{obj} \to -\infty & \therefore \lambda \geq 1 \end{cases}$$

Based on this result, $\lambda_1 = 1, \lambda_2 = 1$ and thus the optimization problem posed is equivalent to the original problem.

$$\begin{array}{l} \min gx + hx \\ \text{s.t.} \quad Ax = b \end{array}$$

B.4 Proof of Strong Duality of the Lagrangian Relaxation

Theorem B.1. *The Lagrangian relaxation of the maintenance capacity allocation model is a strong dual.*

Proof. The Lagrangian relaxation of the maintenance capacity allocation model is presented in

$$\max_{\lambda} \mathcal{L}(\lambda) \left[\min \sum_{m \in M} \sum_{d_1 \in D} \sum_{d_2 \in D} t_{md_1d_2} - \lambda_{md_1d_2}[t_{md_1d_2} - (h^+_{md_1d_2} + h^-_{md_1d_2})] \right]$$

As noted in §5.5.2, the variable $h_{md_1d_2}$ forms the difference in maintenance capacity allocation at maintenance stations on all days of the planning horizon.

$$\sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(b)=d_1 \\ s(o)=m}} \gamma_e x_{eo} - \sum_{e \in E} \sum_{\substack{o \in O(e): \\ d(o)=d_2 \\ s(o)=m}} \gamma_e x_{eo} = (h^+_{md_1d_2} - h^-_{md_1d_2}) \qquad [1]$$

[1] $\forall m \in M, \forall d_1 \in D, \forall d_2 \in D : d_1 \neq d_2$

Rearranging the terms in the objective, we obtain the following equation.

$$\max_{\lambda} \mathcal{L}(\lambda)$$

where

$$\mathcal{L}(\lambda_{md_1d_2})\left[\min \sum_{m\in M}\sum_{d_1\in D}\sum_{d_2\in D} t_{md_1d_2} - \lambda_{md_1d_2}[t_{md_1d_2} - (h^+_{md_1d_2} + h^-_{md_1d_2})]\right]$$

$$\mathcal{L}(\lambda_{md_1d_2})\left[\min \sum_{m\in M}\sum_{d_1\in D}\sum_{d_2\in D} (1-\lambda_{md_1d_2})t_{md_1d_2} + \lambda_{md_1d_2}(h^+_{md_1d_2} + h^-_{md_1d_2})\right]$$

In this new objective, if $\lambda \geq 1$ then the obvious choice is to make $t \to \infty$, which would make the objective $-\infty$. Since we seek a maximum solution, it must be the case that $\lambda \leq 1$.

Based on this result, the variable t will always be set at 0 and thus we can remove it from the objective entirely.

$$\max_{\lambda_{md_1d_2}} \mathcal{L}(\lambda_{md_1d_2})\left[\min \sum_{m\in M}\sum_{d_1\in D}\sum_{d_2\in D} \lambda_{md_1d_2}(h^+_{md_1d_2} + h^-_{md_1d_2})\right]$$

Based on this final formulation shown in the equation above, we make several realizations. Recall that $(h^+_{md_1d_2} + h^-_{md_1d_2})$ represents the difference in terms of maintenance allocation between stations.

In addition, if $\lambda_{md_1d_2} = 1$ at the optimal solution, then we re-write the objective function as follows.

$$\max_{\lambda_{md_1d_2}} \mathcal{L}(\lambda_{md_1d_2})\left[\min \sum_{m\in M}\sum_{d_1\in D}\sum_{d_2\in D} (h^+_{md_1d_2} + h^-_{md_1d_2})\right]$$

This implies that the Lagrangian relaxation actually represents the true objective function as seen below.

$$\max_{\lambda_{md_1d_2}} \mathcal{L}(\lambda_{md_1d_2}) \left[\min \sum_{m \in M} \sum_{d_1 \in D} \sum_{d_2 \in D} t_{md_1d_2} \right]$$

As such, we have shown that the Lagrangian relaxation actually solves the *true* objective function of the integer program we posed from the beginning, which demonstrates that the Lagrangian approach does indeed provide the optimal solution to the original problem.

□

APPENDIX C

Theoretical Model Comparison

In this appendix, we provide a theoretical comparison between the various formulations that have been presented previously. More specifically, we compare the LP-relaxations of previous formulations and motivate their respective equivalence in terms of feasibility and optimality.

In this section, we compare the LP relaxation between two duty-based formulations. The first formulation uses a primary variable b_{ije}. For simplicity, we will refer to this formulation as **Model** B_1 throughout the remainder of this section. The second formulation uses the variable w_{eds} as its primary decision variable. We will refer this formulation as **Model** B_2 for the remainder of this section.

C.1 Importance of comparing the LP Relaxations

When solving linear (integer) optimization problems, a first step is to always solve the LP-relaxation. That is, we assume each variable is continuous and obtain the optimal. The advantage of this approach is that it is fast. However, in many cases, we arrive at optimal solutions that are fractional and thus cannot be easily translated into feasible solutions to the integer program.

In addition, the LP-relaxation provides a bound on the optimal solution of the

integer program. For example, in the case of the MRP-OS model, we seek to maximize the total number of maintenance events that are covered. Solving any of the formulations under their respective LP-relaxation provides an upper-bound on the maximum number of maintenance events that can be covered. As such, solving the LP-relaxation can quickly provide a bound on how effective a particular formulation can be, without solving an integer program to optimality.

C.2 Feasible solution x^{B_1} bounds solution to x^{B_2}

The goal of this section is to show that the LP relaxation of model B_1 can be directly translated into a solution of model B_2 with the same objective function value. This result, in turn, implies that the optimal solution to model B_1, which is an upper bound of the value of the IP, is as least as good as the solution provided by model B_2. In other words, solving model B_1 using an LP relaxation provides the same information as that of model B_2.

Lemma C.1. *For all solutions $\bar{x}$ feasible to model B_1, there exists an equivalent solution $\bar{q}$ feasible to B_2 such that $Z_{B_1}(\bar{x}) = Z_{B_2}(\bar{q})$.*

Proof. We begin with a solution to model B_1 and map its corresponding variables to those in model B_2. This mapping is shown in the equations below, and is common to both models.

$$\bar{x}_{ijt}^{B_1} = \bar{x}_{ijt}^{B_2}$$

$$\bar{y}_{it}^{B_1} = \bar{y}_{it}^{B_2}$$

$$\bar{z}_{it}^{B_1} = \bar{z}_{it}^{B_2}$$

$$\sum_{\substack{i \in L(t): \\ d(i)=d \\ s(i)=s}} \sum_{j \in D(i,e)} \bar{b}_{ije}^{B_1} = \bar{w}_{eds}^{B_2} \qquad \forall t \in T, \forall e \in E(t), \forall s \in S(e), \forall d \in D(e)$$

In addition to showing that the decision variables of model B_1 map directly to those of model B_2, we now present a similar construction of the constraints. It should be noted that the following set of constraints, shown below, are common to both models B_1 and B_2.

$$\sum_{j \in U(i,t)} \bar{x}_{jit} + y_{it} - \sum_{j \in D(i,t)} x_{ijt} - z_{it} = 0 \qquad \forall t \in T, \forall i \in L(t)$$

$$\sum_{i \in L} y_{it} \leq 1 \qquad \forall t \in T$$

$$\sum_{t \in T} \left(y_{it} + \sum_{j \in U(i,t)} x_{jit} \right) = 1 \qquad \forall i \in L$$

In addition, in both formulations, we require that each maintenance event is covered at most once using the equation below.

$$\sum_{s \in S(e)} \sum_{d \in D(e)} \sum_{\substack{i \in L(t): \\ d(i)=d \\ s(i)=s}} \sum_{j \in D(i,e)} b_{ije}^{B_1} = \sum_{s \in S(e)} \sum_{d \in D(e)} w_{eds}^{B_2} \leq 1 \qquad \forall t \in T, \forall e \in E(t)$$

Next, we use a similar approach to show equivalence between the capacity constraint in both models.

$$\sum_{t\in T}\sum_{e\in E(t)}\sum_{s\in S(e)}\sum_{d\in D(e)}\sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)}\gamma_e b^{B_1}_{ije} = \sum_{t\in T}\sum_{\substack{e\in E(t):\\ s\in S(e)\\ d\in D(e)}}\gamma_e w^{B_2}_{eds} \leq \rho_{sd} \quad \forall s\in M, \forall d\in D$$

The last constraint common to both models connects the assignment of duties using the variable x_{ijt} with actual maintenance events. This constraint is shown in the equations below, reproduced from model B_1 and model B_2 respectively.

$$b^{B_1}_{ije} - x^{B_1}_{ijt} \leq 0 \qquad \forall t\in T, \forall e\in E(t), \forall i\in L(t), \forall j\in D(i,e)$$

$$w^{B_2}_{eds} - \sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)} x^{B_2}_{ijt} \leq 0 \qquad \forall t\in T, \forall e\in E(t), \forall s\in S(e), \forall d\in D(e)$$

We replace the left-hand-side from the first equation with its equivalent $b^{B_1}_{ije}$ variable, and obtain the following constraints.

$$\sum_{s\in S(e)}\sum_{d\in D(e)}\sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)} b_{ije} - \sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)} x_{ijt} \leq 0 \qquad [1]$$

[1] $\forall t\in T, \forall e\in E(t), \forall s\in S(e), \forall d\in D(e)$

Based on these constraints, we note that a simple summation in both terms of the second constraint will result in the equivalent version of the previous constraint as show in the equation above. □

Corollary C.2. *If there exists an optimal solution x^* to model B_1, then there exists an optimal solution q^* such that $Z^*_{B_1} = Z^*_{B_2}$.*

This follows directly from Lemma (C.1).

C.3 Feasible solution x^{B_2} bounds solution to x^{B_1}

Lemma C.3. *For all solutions $\bar{q}$ feasible to model B_2, there exists an equivalent solution $\bar{x}$ feasible to B_1 such that $Z_{B_2}(\bar{q}) = Z_{B_1}(\bar{x})$.*

Proof. We show that for any feasible solution to the LP relaxation of B_2, we can find a feasible solution to the LP relaxation of B_1 with the same value.

$$\bar{x}^{B_2}_{ijt} = \bar{x}^{B_1}_{ijt}$$

$$\bar{y}^{B_2}_{it} = \bar{y}^{B_1}_{it}$$

$$\bar{z}^{B_2}_{it} = \bar{z}^{B_1}_{it}$$

$$\bar{w}^{B_2}_{eds} = \sum_{\substack{i \in L(t):\\ d(i)=d \\ s(i)=s}} \sum_{j \in D(i,e)} b^{B_1}_{ije} \qquad \forall t \in T, \forall e \in E(t), \forall s \in S(e), \forall d \in D(e)$$

As in the previous proof, the set of flow constraints, as shown in the equations above, are common to both. Furthermore, in both formulations we require that each maintenance event is covered at most once. Equivalence of these constraints is shown below.

$$\sum_{s \in S(e)} \sum_{d \in D(e)} w^{B_2}_{eds} = \sum_{s \in S(e)} \sum_{d \in D(e)} \sum_{\substack{i \in L(t):\\ d(i)=d \\ s(i)=s}} \sum_{j \in D(i,e)} b^{B_1}_{ije} \leq 1 \qquad \forall t \in T, \forall e \in E(t)$$

Capacity constraint equivalence is shown in the equation below.

$$\sum_{t\in T}\sum_{e\in E(t)}\sum_{s\in S(e)}\sum_{d\in D(e)}\sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)}\gamma_e b_{ije}^{B_1}=\sum_{t\in T}\sum_{\substack{e\in E(t):\\ s\in S(e)\\ d\in D(e)}}\gamma_e w_{eds}^{B_2}\leq \rho_{sd}\quad \forall s\in M,\forall d\in D$$

Finally, a set of constraints connects the variable x_{ijt} with actual maintenance events. This constraint, the equation below, can be summed over all possible lines-of-flight and still remain equivalent.

$$b_{ije}^{B_1}-x_{ijt}^{B_1}\leq 0 \qquad \forall t\in T,\forall e\in E(t),\forall i\in L(t),\forall j\in D(i,e)$$

$$\sum_{s\in S(e)}\sum_{d\in D(e)}\sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)} b_{ije}^{B_1}-\sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)} x_{ijt}^{B_1}\leq 0 \qquad [1]$$

[1] $\forall t\in T,\forall e\in E(t),\forall s\in S(e),\forall d\in D(e)$.

We replace the left-hand-side from the above-equation with its equivalent $w_{eds}^{B_2}$ variable, and obtain the following constraints.

$$w_{eds}^{B_2}-\sum_{\substack{i\in L(t):\\ d(i)=d\\ s(i)=s}}\sum_{j\in D(i,e)} x_{ijt}^{B_2}\leq 0 \qquad \forall t\in T,\forall e\in E(t),\forall s\in S(e),\forall d\in D(e)$$

We have shown equivalence between the solutions in both models which implies the same objective function value for an equivalent solution. □

Corollary C.4. *If there exists an optimal solution q^* to model B_2, then there exists an optimal solution x^* such that $Z^*_{B_2} = Z^*_{B_1}$.*

This follows directly from Lemma (C.3).

Corollary C.5. *An optimal solution to the LP relaxation of model B_1 has an equivalent optimal solution in model B_2.*

This follows directly from Lemma (C.1) and Lemma (C.3).

C.4 Solution Mapping

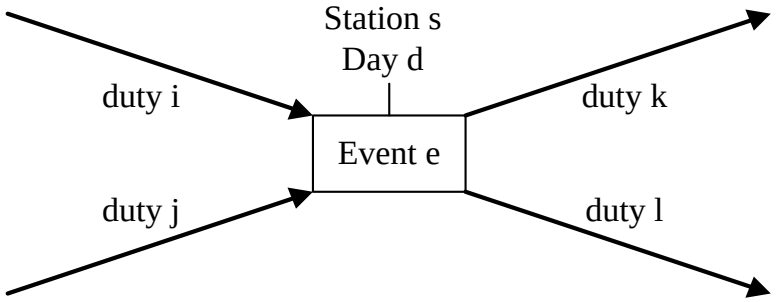
In the previous sections, we demonstrated the equivalence between the models and their respective LP-relaxations. In this particular section, we focus on the actual solution, and the translation of a solution from one model to another. The purpose of this section is to show the equivalence between actual solutions and their respective mappings, which we will then use to demonstrate that solving one model is computationally more efficient, while providing the same final objective.

First, we argue that a fractional solution to model B_1 will have a unique and direct solution in model B_2. In addition, we will provide a counter example that illustrates that a fractional solution to model B_2 can be translated into more than one solution in model B_1. Finally, we will argue that the integer solution to model B_1 has a direct map to a solution in model B_2, but could feature a different underlying flight plan.

Corollary C.6. *A fractional solution to model B_2 has a unique and direct mapping to a solution in model B_1.*

This follows directly from Lemma (C.3).

While the mapping of a solution from model B_1 has a direct and unique solution in model B_2, the reverse is not guaranteed. To illustrate this, we will provide a simple counter example that shows that various combinations of the underlying variable assignment in model B_1 can result in a single solution in model B_2, or in other words, a single solution to model B_2 will allow for several possible solution mappings in model B_1.



Suppose we have a situation as illustrated in the figure above. In this case, we have four duties which all meet to possibly cover a maintenance event e. Suppose that in the model B_2 formulation, the final solution for this situation is $w_{eds}^{B_2} = 0.5$ due to a capacity restriction. In other words, we able to cover a maintenance event e at station s on day d with a value of 0.5. On the other hand, the following solutions provide the equivalent objective function value, by Lemma (C.3), but are fundamentally different in their actual assignment of the fractional solution.

Solution 1		**Solution 2**
$b_{ile}^{B_1} = 0.1$		$b_{ile}^{B_1} = 0.1$
$b_{ike}^{B_1} = 0.1$		$b_{ike}^{B_1} = 0.2$
$b_{jle}^{B_1} = 0.2$		$b_{jle}^{B_1} = 0.1$
$b_{jke}^{B_1} = 0.1$		$b_{jke}^{B_1} = 0.1$

So far, we have shown that there exists a unique and direct mapping between a solution in model B_1 and a solution in model B_2. On the other hand, we have

also seen through a counter-example, that a one-to-many relationship exists between a solution in model B_2 and equivalent solutions in model B_1. However, in both of these cases, we argued that these relationships hold under the LP-relaxation in which variable assignments can be fractional. To conclude this section, we now analyze the integer case. That is, we argue that while there while there is one-to-one relationship between a solution to model B_1 and that of model B_2, depending on the underlying network structure, these solutions may not result result in the same, unique solution.

Lemma (C.1) provides the fact that a solution in model B_1 can be translated to a solution in model B_2. We now extend this framework to argue that a one-to-one solution exists between the two models.

Lemma C.7. *An integer solution in model B_1 has a direct mapping to a single integer solution in model B_2 and vice-versa.*

Proof. We have previously argued that the equation below holds. This equation illustrates the equivalence between between fractional solution in model B_1 and its mapping to a solution in model B_2. We now show that this relationship still holds under integer variable restrictions.

$$\sum_{\substack{i \in L(t): \\ d(i)=d \\ s(i)=s}} \sum_{j \in D(i,e)} b^A_{ije} = w^B_{eds} \qquad \forall t \in T, \forall e \in E(t), \forall s \in S(e), \forall d \in D(e)$$

The constraint below in both model B_1 and model B_2 restricts the flow across duties in the network. This flow constraint also requires the only one of the $x_{ijt}, \forall i \in L$ variables to be 1. In other words, a single duty can be associated with at most one other duty, through the $\sum_{j \in D(i,t)} x_{ijt} \leq 1$. This constraint along with the following

equation in model B_1 requires that the b_{ije} variables share the same property, i.e. only one of the b_{ije} variables is 1 for all each of the duties i.

$$\sum_{j \in U(i,t)} x_{jit} + y_{it} - \sum_{j \in D(i,t)} x_{ijt} - z_{it} = 0 \qquad \forall t \in T, \forall i \in L(t)$$

$$b^A_{ije} - x^A_{ijt} \leq 0 \quad \forall t \in T, \forall e \in E(t), \forall i \in L(t), \forall j \in D(i,e)$$

Given the fact that only of of the variables b_{ije} will be one in summation of the equation above, then the variable restrictions hold and the remainder of the constraints will hold true, as with Lemma (C.3). From this, we can conclude that there is a one-to-one mapping between a solution of model B_1 to model B_2 in the integer solution. □

It turns out that even though the solutions have a one-to-one mapping, the underlying network structure can result in a different duty assignment for these particular solutions. In other words, the variable b_{ije} requires that when a maintenance event is assigned, a particular line-of-flight connection is formed between line-of-flight i and line-of-flight j. On the other hand, in model B_2, the assignment of w_{eds} does specific a particular maintenance event, but it does not require a specific connection between the underlying model, allowing the solution process to determine the actual assignment of duties to tails.

APPENDIX D

Generation Algorithms

D.1 Generating Initial Rotations

Algorithm 1 Initiate Rotations Function

for all t in the set T **do**
  for all l in the set $L(t)$ **do**
    if $arp(l) == loc(t)$ and $day(l) == 0$ **then**
      $newRotation()$
      $buildFullRotation(t, newRotation)$
    end if
  end for
end for

D.2 Building Rotations

Algorithm 2 Build Rotations Function

{Stopping criteria}
if $size(outbounds$ for $newRotation) == 0$ **then**
  add $newRotation$ to the set R for tail t
  $remove$(last line l) from $newRotation$
  $return()$
end if
{Continue building}
for all l outbound to newRotation **do**
  add l to $newRotation$
  $buildRotations(t, newRotation)$
end for

D.3 Inserting Maintenance Events in Rotations

Algorithm 3 Insert Maintenance Events

for all t in the set T **do**
  for all r in the set $R(t)$ **do**
    Determine applicable checks C for rotation r
    for all c in the set C **do**
      Determine the set of opportunities $O(c)$
      for each combination m in $O(A) \times O(B) \times O(C)$ **do**
        Insert maintenance event(s) into rotation r
        Add rotation r to set of active rotations Ω_t^A for tail t.
      end for
    end for
  end for
end for

D.4 Building Initial Recurring Maintenance Rotations

Algorithm 4 Initiate Rotations Function

for all t in the set T **do**
  for all l in the set $L(t)$ **do**
    if $arp(l) == loc(t)$ and $day(l) == 0$ **then**
      $newRotation()$
      $newACheck(t.hours, t.cycles)$
      $newBCheck(t.hours, t.cycles)$
      $buildFullRotation(t, newRotation, newACheck, newBCheck)$
    end if
  end for
end for

D.5 Inserting Checks into Recurring Maintenance Rotations

Algorithm 5 Build Rotations Function

{Stopping criteria}
if $size(outbounds$ for $newRotation) == 0$ **then**
  if PERFORM-OPTIONAL-MTN-END $== true$ **then**
    $newRotation \Leftarrow newEvent(Check)$
  end if
  add $newRotation$ to the set R for tail t
  $remove$(last line l) from $newRotation$
  $return()$
end if
{Continue building}
for all l outbound to newRotation **do**
  if $((l.hours + aCheck.hours \geq aLimit.hours)$ **or** $(l.cycles + aCheck.cycles \geq aLimit.cycles))$
      and $isMaintenance(loc(newRotation))$ **and** $enoughTime$ **then**
    $newRotation \Leftarrow newEvent(ACheck)$
    $aCheck.hours \Leftarrow 0$
    $aCheck.cycles \Leftarrow 0$
    $CheckAddedA \Leftarrow true$
  else
    $delete(newRotation)$
  end if
  if $((l.hours + bCheck.hours \geq bLimit.hours)$ **or** $(l.cycles + bCheck.cycles \geq bLimit.cycles))$
      and $isMaintenance(loc(newRotation))$ **and** $enoughTime$ **then**
    $newRotation \Leftarrow newEvent(BCheck)$
    $bCheck.hours \Leftarrow 0$
    $bCheck.cycles \Leftarrow 0$
    $CheckAddedB \Leftarrow true$
  else
    $delete(newRotation)$
  end if
  if $checkAddedA$ or $checkAddedB$ **then**
    for each day up to MAX-DAY-EARLY **do**
      $newRotationEarly()$
      Copy lines to $newRotationEarly$ from $newRotation$ up to ($day-$ MAX-DAY-EARLY)
      If possible, add Check
      $Check.hours \Leftarrow 0$
      $Check.cycles \Leftarrow 0$
      $buildRotations(t, newRotationEarly, aCheck, bCheck)$
    end for
  end if
  $buildRotations(t, newRotation, aCheck, bCheck)$
end for

Bibliography

[1] Jeph Abara. Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4):20–28, 1989.

[2] Ahmed Abdelghany and Khaled Abdelghany. *Modeling Applications in the Airline Industry*. Ashgate, 2010.

[3] Ahmed Abdelghany, Khaled Abdelghany, and Ram Narasimhan. Scheduling baggage-handling facilities in congested airports. *Journal of Air Transport Management*, 12(2):76 – 81, 2006.

[4] Ahmed Abdelghany, Goutham Ekollu, Ram Narasimhan, and Khaled Abdelghany. A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research*, 127:309–331, 2004.

[5] Khaled F. Abdelghany, Ahmed F. Abdelghany, and Goutham Ekollu. An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, 185(2):825 – 848, 2008.

[6] Khaled F. Abdelghany, Sharmila S. Shah, Sidhartha Raina, and Ahmed F. Abdelghany. A model for projecting flight delays during irregular operation conditions. *Journal of Air Transport Management*, 10(6):385 – 394, 2004.

[7] Shervin AhmadBeygi, Amy Cohn, Yihan Guan, and Peter Belobaba. Analysis

of the potential for delay propagation in passenger airline networks. *Journal of Air Transport Management*, 14(5):221–236, 2008.

[8] Shervin Ahmadbeygi, Amy Cohn, and Marcial Lapp. Decreasing airline delay propagation by re-allocating scheduled slack. *IIE Transactions*, 2010.

[9] Jeffrey M. Alden and Robert L. Smith. Rolling horizon procedures in nonhomogeneous markov decision processes. *Operations Research*, 40:pp. S183–S194, 1992.

[10] Ranga Anbil, Rajan Tanga, and Ellis L. Johnson. A global approach to crew pairing. *IBM Systems Journal*, 31:71–78, 1992.

[11] Hojong Baik and Antonio A. Trani. Framework of a time-based simulation model for the analysis of airfield operations. *Journal of Transportation Engineering*, 134(10):397–413, 2008.

[12] Kenneth R. Baker. An experimental study of the effectiveness of rolling schedules in production planning. *Decision Sciences*, 8(1):19–27, 1977.

[13] Poornima Balakrishna, Rajesh Ganesan, and Lance Sherry. Airport taxi-out prediction using approximate dynamic programming: Intelligence-based paradigm. *Transportation Research Record: Journal of the Transportation Research Board*, 2052(1):53–61, 2008.

[14] Michael O. Ball, Lawrence M. Ausubel, Frank Berardino, Peter Cramton, George Donohue, Mark Hansen, and Karla Hoffman. Market-based alternatives for managing congestion at new york's laguardia airport. Technical report, University of Maryland, Department of Economics - Peter Cramton, 2007.

[15] James Finlay Barlow, Teresa Anna Maria Valdivieso, Anadi Gopal Risal, Vijay R. Shah, Sandeep Gangadhar Parmekar, Ramesh Venkata Perumal, Sarah Frances Davies, and Jeffrey Alan Peterson. Airline flight reservation system simulator for optimizing revenues, 1997.

[16] Cynthia Barnhart, Amy Cohn, Ellis L. Johnson, Diego Klabjan, George Nemhauser, and Pamela Vance. *Airline Crew Scheduling.* Kluwer Scientific Publishers, New York, NY, USA, 2003.

[17] Cynthia Barnhart, Amr Farahat, and Manoj Lohatepanont. Airline fleet assignment with enhanced revenue modeling. *Operations Research*, 57(1):231–244, 2009.

[18] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. *Integer multicommodity flow problems*, volume Volume 3011/2004. Springer Berlin / Heidelberg, 1996.

[19] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329, 1998.

[20] Cynthia Barnhart, Timothy S. Kniker, and Manoj Lohatepanont. Itinerary-Based Airline Fleet Assignment. *Transportation Science*, 36(2):199–217, 2002.

[21] Peter Belobaba, Amedeo Odoni, and Cynthia Barnhart. *Fundamentals of Pricing and Revenue Management.* John Wiley & Sons, Ltd., 2009.

[22] Peter Belobaba, Amedeo Odoni, and Cynthia Barnhart. *The Global Airline Industry.* John Wiley & Sons, Ltd., 2009.

[23] Peter P. Belobaba and John L. Wilson. Impacts of yield management in competitive airline markets. *Journal of Air Transport Management*, 3(1):3 – 9, 1997.

[24] Matthew E. Berge and Craig A. Hopperstad. Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms. *Operations Research*, 41(1):153–168, 1993.

[25] Richard A. Bihr. A conceptual solution to the aircraft gate assignment problem using 0, 1 linear programming. *Computers & Industrial Engineering*, 19(1-4):280 – 284, 1990.

[26] A. Bolat. Assigning arriving flights at an airport to the available gates. *The Journal of the Operational Research Society*, 50(1):23–34, 1999.

[27] Ralf Borndörfer, Ivan Dovica, Ivo Nowak, and Thomas Schickinger. Robust tail assignment. In *Proceedings of The Fiftieth Annual Symposium of AGIFORS*, 2010.

[28] Bureau of Transportation Statistics. *TranStats, Flights*, April 2010.

[29] Alberto Caprara, Paolo Toth, Daniele Vigo, and Matteo Fischetti. Modeling and solving the crew rostering problem. *Operations Research*, 46(6):820–830, 1998.

[30] Lloyd Clarke, Christopher Hane, Ellis Johnson, and George Nemhauser. Maintenance and Crew Considerations in Fleet Assignment. *Transportation Science*, 30(3):249–260, 1996.

[31] Lloyd Clarke, Ellis Johnson, George Nemhauser, and Zhongxi Zhu. The aircraft rotation problem. *Annals of Operations Research*, 69(1):33–46, 1997.

[32] Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J. Rezanova. Disruption management in the airline industry–concepts, models and methods. *Computers & Operations Research*, 37(5):809 – 821, 2010. Disruption Management.

[33] James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith. *Airline Resource Scheduling*. John Wiley & Sons, Inc., 2010.

[34] Amy M. Cohn and Cynthia Barnhart. Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, 51(3):387–396, 2003.

[35] Jean-Francois Cordeau, Goran Stojkovic, Francois Soumis, and Jacques Desrosiers. Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling. *Transportation Science*, 35(4):375–388, 2001.

[36] Mark S. Daskin and Nicholaos D. Panayotopoulos. A Lagrangian Relaxation Approach to Assigning Aircraft to Routes in Hub and Spoke Networks. *Transportation Science*, 23(2):91–99, 1989.

[37] Guy Desaulniers, Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and Franois Soumis. Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855, 1997.

[38] Sophie Dickson and Natashia Boland. Robust airline scheduling: Improving schedule robustness with flight re-timing and aircraft swapping. In *20th International Symposium on Mathematical Programming*, Chicago, August 2009. ISMP.

[39] Rigas Doganis. *Flying off course, the economics of international airlines*. Routledge, 2002.

[40] Ulrich Dorndorf, Andreas Drexl, Yury Nikulin, and Erwin Pesch. Flight gate scheduling: State-of-the-art and recent developments. *Omega*, 35(3):326 – 334, 2007.

[41] Jonathan Dumas, Fati Aithnard, and Franois Soumis. Improving the objective function of the fleet assignment problem. *Transportation Research Part B: Methodological*, 43(4):466 – 475, 2009.

[42] Jonathan Dumas and Francois Soumis. Passenger Flow Model for Airline Networks. *Transportation Science*, 42(2):197–207, 2008.

[43] Niklaus Eggenberg, Matteo Salani, and Michel Bierlaire. Constraint-specific recovery network for solving airline recovery problems. *Computers & Operations Research*, 37(6):1014 – 1026, 2010.

[44] Matthias Ehrgott and David M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11(3):139–150, 2002.

[45] D. Espinoza, R. Garcia, M. Goycoolea, G. L. Nemhauser, and M. W. P. Savelsbergh. Per-seat, on-demand air transportation part i: Problem description and an integer multicommodity flow model. *Transportation Science*, 42(3):263–278, 2009.

[46] D. Espinoza, R. Garcia, M. Goycoolea, G. L. Nemhauser, and M. W. P. Savelsbergh. Per-seat, on-demand air transportation part ii: Parallel local search. *Transportation Science*, 42(3):279–291, 2009.

[47] Thomas Fiig, Karl Isler, Craig Hopperstad, and Peter Belobaba. Optimization of mixed fare structures: Theory and applications. *Journal of Revenue and Pricing Management*, 9:152–170(19), 3 January 2010.

[48] Jerzy Filar, Prabhu Manyem, and Kevin White. How airlines and airports recover from schedule perturbations: A survey. *Annals of Operations Research*, 108:315–333, 2001.

[49] Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[50] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 50(12):1778–1780, 2004.

[51] Sami Gabteni and Mattias Grönkvist. Combining column generation and constraint programming to solve the tail assignment problem. *Annals of Operations Research*, 171(1):61–76, 2009.

[52] Chunhua Gao, Ellis Johnson, and Barry Smith. Integrated airline fleet and crew robust planning. *Transportation Science*, 43(1):2–16, 2009.

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

[54] Ram Gopalan and K.T. Talluri. Mathematical models in airline schedule planning: A survey. *Annals of Operations Research*, 76:155–185(31), 1998.

[55] Mattias Grönkvist. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, chapter A Constraint Programming Model for Tail Assignment, pages 142–156. Springer Berlin / Heidelberg, 2004.

[56] Mattias Grönkvist. *The Tail Assignment Problem*. PhD thesis, Chalmers University of Technology, 2005.

[57] Ali Haghani and Min-Ching Chen. Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice*, 32(6):437 – 454, 1998.

[58] Christopher A. Hane, Cynthia Barnhart, Ellis L. Johnson, Roy E. Marsten, George L. Nemhauser, and Gabriele Sigismondi. The fleet assignment problem: solving a large-scale integer program. *Math. Program.*, 70(2):211–232, 1995.

[59] Pavithra Harsha. *Mitigating airport congestion : market mechanisms and airline response models*. PhD thesis, MIT, 2009.

[60] Timothy L. Jacobs, Barry C. Smith, and Ellis L. Johnson. Incorporating Network Flow Effects into the Airline Fleet Assignment Process. *Transportation Science*, 42(4):514–529, 2008.

[61] Hai Jiang and Cynthia Barnhart. Dynamic airline scheduling. *Transportation Science*, 43(3):336–354, 2009.

[62] Nader Kabbani and Bruce Patty. Aircraft routing at american airlines. In *Proceedings of The Thirty-Second Annual Symposium of AGIFORS*, 1992.

[63] Natalia Kliewer, Taeb Mellouli, and Leena Suhl. A time-space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616 – 1627, 2006.

[64] Niklas Kohl, Allan Larsen, Jesper Larsen, Alex Ross, and Sergey Tiourine. Airline disruption management–perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149 – 162, 2007.

[65] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[66] Shan Lan, John-Paul Clarke, and Cynthia Barnhart. Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions. *Transportation Science*, 40(1):15–28, 2006.

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

[68] Benoit Lardeux, Oriana Goyons, and Charles-Antoine Robelin. Availability calculation based on robust optimization. *To appear in Journal of Pricing and Revenue Management*, 2010.

[69] Sylvie Lavoie, Michel Minoux, and Edouard Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35(1):45 – 58, 1988.

[70] Loo Hay Lee, Huei Chuen Huang, Chulung Lee, Ek Peng Chew, W. Jaruphongsa, Yean Yik Yong, Zhe Liang, Chun How Leong, Yen Ping Tan, K. Namburi, E. Johnson, and J. Banks. Discrete event simulation model for airline operations: Simair. In *Proceedings of the Winter Simulation Conference*, volume 2, pages 1656 – 1662, 2003.

[71] Manoj Lohatepanont and Cynthia Barnhart. Airline Schedule Planning: Integrated Models and Algorithms for Schedule Design and Fleet Assignment. *Transportation Science*, 38(1):19–32, 2004.

[72] R. S. Mangoubi and Dennis F. X. Mathaisel. Optimizing Gate Assignments at Airport Terminals. *Transportation Science*, 19(2):173–188, 1985.

[73] Jeffrey I. McGill and Garrett J. van Ryzin. Revenue Management: Research Overview and Prospects. *Transportation Science*, 33(2):233–256, 1999.

[74] Anne Mercier, Jean-Franois Cordeau, and Franois Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451 – 1476, 2005.

[75] Anne Mercier and Franois Soumis. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34(8):2251 – 2265, 2007.

[76] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *The Bell Journal of Economics*, 13(2):402–417, 1982.

[77] Brian Rexing, Cynthia Barnhart, Tim Kniker, Ahmad Jarrah, and Nirup Krishnamurthy. Airline fleet assignment with time windows. *Transportation Science*, 34(1):1–20, 2000.

[78] Jay M. Rosenberger, Ellis L. Johnson, and George L. Nemhauser. Rerouting Aircraft for Airline Recovery. *Transportation Science*, 37(4):408–421, 2003.

[79] Jay M. Rosenberger, Ellis L. Johnson, and George L. Nemhauser. A Robust Fleet-Assignment Model with Hub Isolation and Short Cycles. *Transportation Science*, 38(3):357–368, 2004.

[80] Jay M. Rosenberger, A.J. Schaefer, D. Goldsman, E.L. Johnson, A.J. Kleywegt, and G.L. Nemhauser. Simair: a stochastic model of airline operations. *Proceedings of the Winter Simulation Conference*, 2:1118–1122, 2000.

[81] D. M. Ryan. The solution of massive generalized set partitioning problems in aircrew rostering. *The Journal of the Operational Research Society*, 43(5):459–467, 1992.

[82] Abdulkadir Sarac, Rajan Batta, and Christopher M. Rump. A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175(3):1850 – 1869, 2006.

[83] Andrew J. Schaefer, Ellis L. Johnson, Anton J. Kleywegt, and George L. Nemhauser. Airline crew scheduling under uncertainty. *Transportation Science*, 39(3):340–348, 2005.

[84] Sergey Shebalov. Practical overview of demand-driven dispatch. *Journal of Pricing and Revenue Management*, 8:166–173, 2009.

[85] Sergey Shebalov and Diego Klabjan. Robust Airline Crew Pairing: Move-up Crews. *Transportation Science*, 40(3):300–312, 2006.

[86] Hanif D. Sherali, Ki-Hwan Bae, and Mohamed Haouari. Integrated Airline Schedule Design and Fleet Assignment: Polyhedral Analysis and Benders' Decomposition Approach. *INFORMS Journal on Computing*, 2009.

[87] Barry C. Smith and Ellis L. Johnson. Robust Airline Fleet Assignment: Imposing Station Purity Using Station Decomposition. *Transportation Science*, 40(4):497–516, 2006.

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

[89] Raik Stolletz. Non-stationary delay analysis of runway systems. *OR Spectrum*, 30(1):191–213, 2008.

[90] Raik Stolletz. Operational workforce planning for check-in counters at airports. *Transportation Research Part E: Logistics and Transportation Review*, 46(3):414 – 425, 2010.

[91] Kalyan Talluri and Garrett van Ryzin. *The Theory and Practice of Revenue Management*. Kluwer Academic Publishers, 2004.

[92] Kalyan T. Talluri. The Four-Day Aircraft Maintenance Routing Problem. *Transportation Science*, 32(1):43–53, 1998.

[93] Hatice Tekiner, S. Ilker Birbil, and Kerem Bülbül. Robust crew pairing for managing extra flights. *Computers & Operations Research*, 36(6):2031 – 2048, 2009.

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

[95] Bureau of Transportation Statistics U.S. Department of Transportation. Airline on-time statistics and delay causes, 2008.

[96] Menkes H. L. van den Briel, J. Rene Villalobos, Gary L. Hogg, Tim Lindemann, and Anthony V. Mule. America West Airlines Develops Efficient Boarding Strategies. *Interfaces*, 35(3):191–201, 2005.

[97] Pamela H. Vance, Alper Atamturk, Cynthia Barnhart, Eric Gelman, Ellis L. Johnson, Alamuru Krishna, Deepa Mahidhara, George L. Nemhauser, and Ranjit Rebello. A heuristic approach for the airline crew pairing problem. 1997.

[98] Pamela H. Vance, Cynthia Barnhart, Ellis L. Johnson, and George L. Nemhauser. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, 3:111–130, 1994.

[99] Valdemar Warburg, Troels Gotsad Hansen, Allan Larsen, Hans Norman, and Erik Andersson. Dynamic airline scheduling: An analysis of the potentials of refleeting and retiming. *Journal of Air Transport Management*, 14(4):163 – 167, 2008.

[100] Oliver Weide, David Ryan, and Matthias Ehrgott. An iterative approach to robust and integrated aircraft routing and crew scheduling. *Computers & Operations Research*, 37(5):833 – 844, 2010. Disruption Management.

[101] Shangyao Yan, Ta-Hui Yang, and Hsuan-Hung Chen. Airline short-term maintenance manpower supply planning. *Transportation Research Part A: Policy and Practice*, 38(9-10):615 – 642, 2004.

[102] Joyce W. Yen and John R. Birge. A Stochastic Programming Approach to the Airline Crew Scheduling Problem. *Transportation Science*, 40(1):3–14, 2006.