Workload Balancing for Flight Dispatcher Scheduling

by

Rebecca Sarah Rayner

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Unlike other airline operations planning problems, optimization in flight dispatching is not common in literature. Flight dispatchers are centrally located and monitor multiple flights in different places simultaneously. Their work involves planning fuel requirements, routing, and weather monitoring, both before and during a flight. An area of opportunity exists in the assignment of work amongst dispatchers.

A desk contains a series of flights, and is served by a dispatcher or a series of dispatchers working consecutive shifts. In this work, we do not consider shifts and instead focus on assigning flights amongst a set number of desks. Our goal is to balance the workload of each desk, which is measured by the sum of each desk's maximum workload throughout the day.

Two formulations are presented that model the assignment of flights to desks, which we call the Flight Dispatching Problem. The Flight Dispatcher Schedule Formulation (FDSF) assigns flights amongst a set number of desks. The Set Covering Formulation (SCF) selects from known schedules (the assignment of flights to a single desk) to cover all flights with the specified number of desks (i.e., schedules). The base implementation solves the SCF using a column generation approach that creates new schedules with each iteration. Additional variants are also modelled where we limit which flights are assigned to the same desk.

Testing is performed on European Airline Data and American Airlines data. The instances range in size from 46 to 297 flights in one day. We find that the FDSF solves to optimality quickly for small instances but not for the larger ones. The base implementation converges within two hours for the small and mid-size instances. Gaps are reduced using an improvement heuristic in some cases. For the larger instances, neither implementation solves within two hours and the gaps after that time are very large. Constraining the

flight assignments provides trade-offs between computation time (which is typically faster) and solution quality (which is typically worse). We also tested the case where load varies throughout the flight.

For the base implementation, most of the computation time for larger instances is spent in the pricing problem. In some cases, this is improved by generating multiple columns in each iteration instead of just one. The solution of the pricing problem is an area where future work could be focused to improve the computational performance. Other areas for future work include modelling dispatch zones instead of decomposing the problem by zones, changing the balance metric in the objective, incorporating uncertainty, and including the shift component of the dispatching problem.

Acknowledgements

First, I would like to thank my supervisor Dr. Fatma Gzara for her support and guidance throughout this process. Even though we could not meet in person, her consistent feedback and organization provided stability in uncertain times. I also appreciate the opportunities that she provided for me.

I would like to thank my readers, Dr. Samir Elhedhli and Dr. Houra Mahmoudzadeh for their insight and feedback.

Additionally, thank you to my fellow members of the Waterloo Analytics and Optimization (WanOpt) lab who shared their experiences with me. In particular, thanks to Gohram Baloch for helping me get started in the lab and working through problems with me. Also to Paulo de Carvalho, who always had a positive outlook and helped me solve any technical problems, especially when I could not access the lab myself.

These last few years have provided a lot of unique challenges, so I want to thank all my fellow Management Sciences students and friends who helped create a sense of community and normalcy. They helped me stay motivated and accountable through difficult times, and for that I am very grateful.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 2020 and 2021, airline earnings before interest and taxes (EBIT) reached negative margins around the world, however, before COVID-19 these margins were already slim. In 2019, worldwide EBIT margins were 5.2% and have not been above 8.6% in the last 10 years. North America has fared slightly better with recent margins hovering around 9% (Mazareanu, 2021a). With the high costs of airlines, even a small increase in operating margins can lead to billions of dollars in savings. This is one area of opportunity that motivates a lot of operations research work in the airline industry. Research in this field includes flight scheduling, fleet assignment, maintenance routing, crew scheduling, and revenue management. Another potential source for improvement is flight dispatching. Creating flight dispatching schedules can be time-consuming and may not provide the most effective utilization of dispatchers, and thus presents an opportunity for airlines to reduce costs. This could arise both from creating schedules that result in lower dispatcher costs or by reducing the worker-time needed to produce schedules.

1.1 Background

A flight dispatcher communicates with pilots and has responsibilities preceding and during flights to ensure flight safety as well as efficiency. Prior to the flight, their duties include route planning, creating weather reports and fuel calculations, based on factors such as the type of aircraft, its maintenance history, and flight altitude (Kennedy, 1987), (American Airlines Newsroom, 2020). Once flights take-off, dispatchers monitor the flights and any changes to the flight plan (Air Transport Association of Canada, 2021). Avoiding turbulence is one of their goals (Otley, 2018) and they are responsible for determining safe landing places when a plane needs to be diverted (Kennedy, 1987). This collection of work is evaluated by the load, which represents the amount of worker effort to monitor and serve that flight at a given time. These sets of duties are characterized by two periods: the planning period and the post-planning period. The planning period is the time leading up to flight take-off, and the post-planning period is from take-off until landing. There is a higher load during the planning period (Santos et al., 2011).

Dispatchers are employed by airlines and are centrally located, monitoring flights around the world. They monitor flights at many different airports rather than tracking flights only with an origin or destination at their location, however, qualifications are different domestically versus internationally. There is extensive training required to be a dispatcher, and in the US they need to be certified through the Federal Aviation Administration. The Airline Dispatchers Federation is a volunteer organization that represents the dispatch profession, which had 2,154 members at the start of 2020 (Airline Dispatchers Federation, 2021). Dispatchers monitor multiple flights simultaneously, provided that the total workload required for the set of active flights does not exceed the maximum allowed workload. Most of American Airlines' flights are dispatched from the American Airlines'

Dallas Fort Worth (DFW) Integrated Operations Centre (IOC) (Otley, 2018), (Dickson, 2019).

Dispatchers work shifts, and over the course of their shift flights may begin and end. This collection of flights that a dispatcher covers is called a **desk**. A single dispatcher works a shift, typically up to 12 hours, however, desks may be active longer than this and would be covered by multiple dispatchers consecutively without gaps or overlap. Figure 1.1 displays an example of a desk that has eight flights over a 16-hour period, however, the dispatch duties are split into two shifts. The first shift is covered by a dispatcher from 8:00 - 16:00, at which point a second dispatcher relieves the first and works until the desk closes at 24:00. Flights D, E, and F are active at the time the shift switches so are monitored by both dispatchers whereas the other flights are entirely monitored by one dispatcher. Relieving dispatchers need to be updated on the status of flights including weather reports and mechanical failures, and could have 30 flights to take over (Kennedy, 1987).

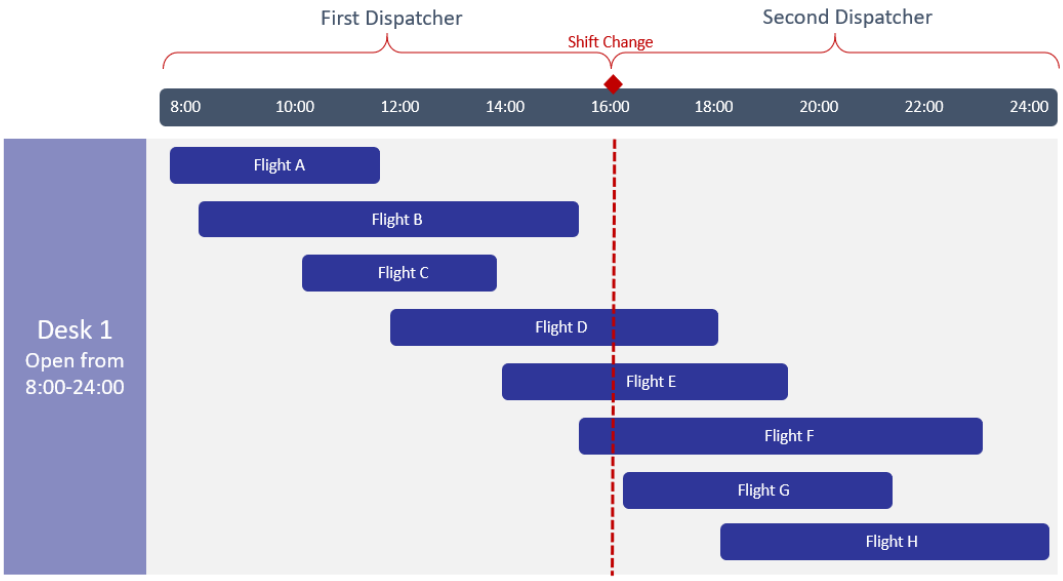


Figure 1.1: Desk split into two shifts.

Airlines have agreements with their dispatchers defining when shifts start and end, and their possible durations. For example, the following shift lengths and restrictions have been used (Santos et al., 2011):

1. Eight hour shift: Cannot start before 500h
2. Ten hour forty minute shift: Cannot start before 500h or end after 230h
3. Twelve hour shift: Run from 600h to 1800h and from 1800h to 600h

There may be a limit to the number of occurrences of each type of shift used in the full schedule. Additionally, in some cases, if a desk is closed and there are still active flights, the remaining load of these flights is switched to another desk (Santos et al., 2011). A flight may be split onto a second or third desk, provided that the entire planning period is during the first desk.

Dispatchers may have varied qualifications, for example, domestic and international qualifications (Otley, 2018). All American Airlines dispatchers are required to have domestic qualifications, and to dispatch flights internationally requires additional experience and training. Airlines use a separate process to determine which dispatchers will serve domestic or international flights (Transport Workers Union of America, 2016). Internationally, flights typically follow a weekly schedule whereas domestic flights typically follow a daily schedule (Barnhart et al., 2006).

To address a gap in the flight dispatching literature, we develop two formulations to model the assignment of flights to desks. The first is a full formulation of the problem and the second is a set covering formulation that is solved using column generation. Heuristic approaches are also presented. These models create schedules that assign all flights to desks and capture maximum desk workloads. The full formulation provides optimal solutions

for small instances with the column generation providing better solutions on the larger instances tested.

This thesis is organized as follows. Chapter 2 describes the problem of flight dispatching. Chapter 3 reviews literature in the airline industry as well as in problems that may have similar formulations or goals as the flight dispatching problem. Chapter 4 presents several models and the solution methodology used to solve the dispatching problem. Heuristics used to initialize and improve solutions are discussed in Chapter 5. Testing results along with the data used are presented in Chapter 6. Potential extensions are discussed in Chapter 7, and lastly, conclusions are presented in Chapter 8.

Chapter 2

Flight Dispatching Problem

In this chapter, we define the flight dispatching problem. Firstly, in Section 2.1 we define a schedule with different possible constraints. A sample instance is introduced and used to demonstrate a feasible solution to the flight dispatching problem using a set of schedules. Load and workload are defined in Section 2.2, and properties of the load are discussed. Sample calculations are demonstrated. Possible measures of balance are explored in Section 2.3 to define the objective for the flight dispatching problem.

2.1 Defining Schedules

We define a schedule as the flights assigned to one desk. For this work, schedules are created at the daily level, containing flights that depart in one workday. Schedules have few constraints though they must contain at least one flight, otherwise they would correspond to an empty desk. There are no constraints for when a schedule starts or ends or their duration, and we do not limit the idle time during a schedule.

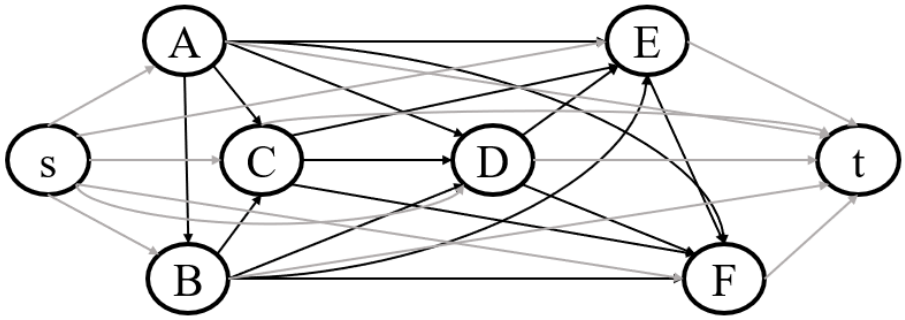
In creating schedules, limits can be imposed on which flights are allowed to be scheduled together. In the base case, any flight can be scheduled with any other flight. We consider the effect of limiting which flights can be scheduled together. One way of doing this is limiting any flights that depart within a specified time interval from being scheduled together. For example, any flights that have the same departure time cannot be on the same schedule. Another way to consider this is to limit the number of flights that depart within some time interval. We call these flight overlap constraints.

A sample instance of flights with their departure and arrival times, as well as the timing and load of the planning and post-planning periods is shown in Table 2.1. Each flight $i \in I$ has a departure time, S_i, arrival time, and the load required at each time t throughout the duration of the flight, f_{it}. For example, flight A's planning period is from time 400-699 with a load of 25 units and A's post-planning period is from time 700-900 with a load of 15 units. Time is displayed in minutes elapsed since midnight.

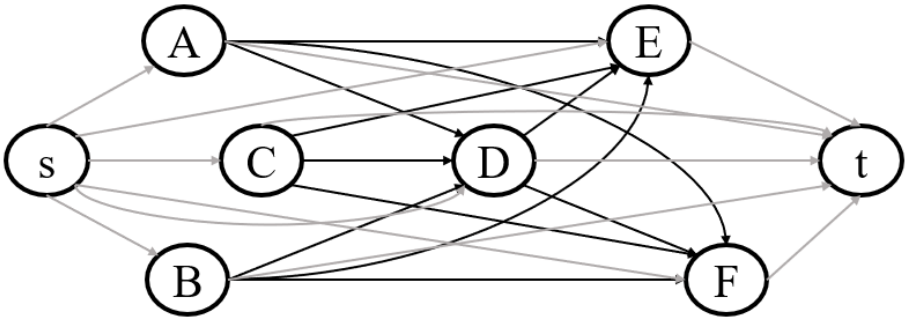
	Departure Time	**Planning Period End**	**Planning Load**	**Post-Planning Period Begin**	**Arrival Time**	**Post-Planning Load**
A	400	699	25	700	900	15
B	400	799	20	800	1200	10
C	500	799	20	800	1200	15
D	800	1199	20	1200	1600	10
E	1000	1299	15	1300	1500	10
F	1100	1399	25	1400	1800	15
G	1300	1699	15	1700	2100	5
H	1400	1899	15	1900	2100	10
I	1500	1899	15	1900	2300	5
J	1800	2299	10	2300	2800	5

Table 2.1: Sample instance of flights to be assigned to desks.

The flight dispatching problem can be modelled using a network structure, where nodes represent flights and an arc between two nodes exists if the corresponding flights can be assigned to the same desk. Arcs are directed, flowing from earlier flights to later flights.



(a) Network with no flight overlap constraints.



(b) Network with flight overlap constraint of 1 hour.

Figure 2.1: Flight schedule network representation.

Ties for flights departing at the same time are arbitrarily broken. Dummy source (s) and sink (t) nodes are created to represent the start and end of a path. Each path from s to t represents a feasible schedule. Figures 2.1a and 2.1b show the networks created using flights A-F from Table 2.1. Depending on how we define the problem, different network representations are created. In Figure 2.1a, we use the basic definition of a schedule. On this network, a feasible schedule could be created in which all flights are on a single desk. In Figure 2.1b, we constrain that flights departing within one hour of each other cannot be on the same schedule. This means flights A, B, and C must all be scheduled separately and therefore, at least three schedules/paths are needed to cover all flights.

Any path from s to t is a feasible schedule. To solve the flight dispatching problem we need a set of schedules that cover all flights given a number of desks, say m. This means that a feasible solution to the flight dispatching problem is one that uses m paths where each flight node is on exactly one path. Figure 2.2 shows two schedules that together cover all flights once. One schedule contains flights B, C, E, H, and I, and is labelled as Desk 1. The second schedule contains flights A, D, F, G, and J, and is labelled as Desk 2. In this case, two schedules are created, although many other schedules could be created from the same set of flights and there are other feasible solutions to the flight dispatching problem with two desks.

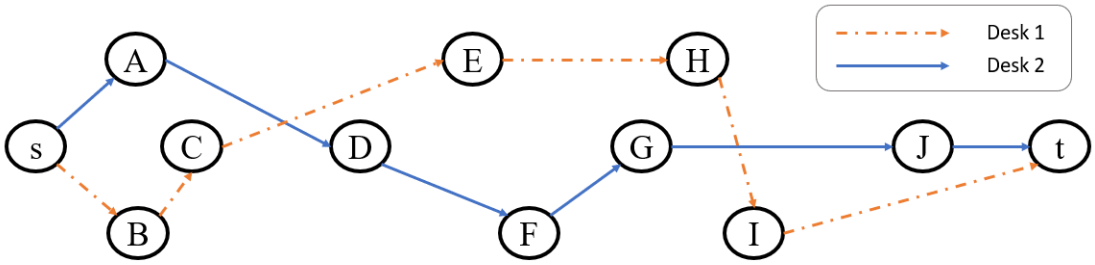


Figure 2.2: Flight to desk allocation with two desks.

Another way to visualize these schedules is shown in Figure 2.3, with each desk representing one schedule. This shows the duration that flights are active, so it allows us to see the overlap in flights. For example, at the start of flight H, the only other flight active for Desk 1 is flight E. This visualizes when the desk is closed, which is at 23:00 for Desk 1.

A single schedule corresponds to one desk, however, a desk may be split up and covered by multiple dispatchers in consecutive shifts. While shifts have time-based regulations, desks are not restricted. In this work, we only consider the creation of desks and do not consider how those desks are split into shifts. Theoretically, desks could be open indefinitely, however, we are creating schedules for a single day. Since domestic flights are

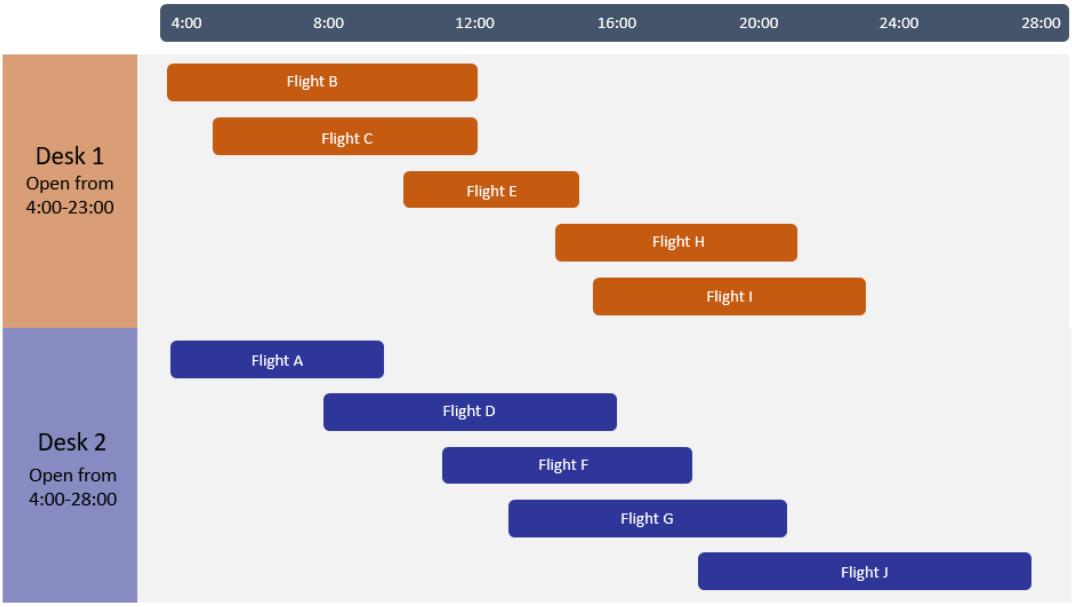


Figure 2.3: Schedule Gantt chart with two desks.

typically scheduled at a daily level, the schedules can repeat each day. For longer schedules, the same modelling could be used, however, any schedule used corresponds to one desk that is continuously open (that would likely need to be covered by multiple dispatcher shifts) with its maximum workload representing a single point in time. Longer schedules could be better modelled by incorporating the shift component where workload is measured for each shift rather than for a whole desk.

2.2 Defining Load

In Table 2.1, planning and post-planning loads are reported. We define load as the amount of work required by a single flight at a given time, which in the table is shown as the departure time until the end of the planning period as the planning load, and the remaining

time until arrival as the post-planning load. The planning period has a higher load than the post-planning period, which is generalized by the rule that the load is non-increasing throughout the duration of the flight. Conversely, the workload is the sum of the loads of the collection of flights assigned to a desk, or path, at a given time. Load and workload are both dependent on time, with the maximum workload defined as the maximum workload over time experienced by a single desk. Load is a unitless figure, though it could be defined as the number of minutes of work required in a time range (e.g., an hour) or refer to an amount of energy/focus needed by a dispatcher. This would allow for meaningful limits on a desk's workload.

Desk workload is calculated at the start of any flight assigned to that desk. This is calculated by adding the planning load of that flight plus the loads of any flights that are still active on the desk at that time. Since flight load cannot increase, and we are only concerned with the maximum workload, we only need to consider the workload at the departure time of flights - the only time it is possible for a desk's workload to increase. We denote the workload at the start of a flight by F_i. To illustrate the workload, consider Desk 1 from Figure 2.2. For this instance, the maximum workload for a desk at any given time is 40 units. For Desk 1, the workloads at the start of each flight are:

$$F_B = f_{B,400} = 20$$

$$F_C = f_{C,500} + f_{B,500} = 20 + 20 = 40$$

$$F_E = f_{E,1000} + f_{C,1000} + f_{B,1000} = 15 + 15 + 10 = 40$$

$$F_H = f_{H,1400} + f_{E,1400} + f_{C,1400} + f_{B,1400} = 15 + 10 + 0 + 0 = 25$$

$$F_I = f_{I,1500} + f_{H,1500} + f_{E,1500} = 15 + 15 + 0 = 30$$

We do not need to check the workload at the times of flights not assigned to this desk since they do not affect the workload. For example, since flight D is not added to this desk, we do not consider the workload at time $S_D = 800$. We know that because no flights are added between time 500 (S_C) and 800, the workload cannot exceed 40 during that time. In order to calculate workload, there must be a way to determine each flight's load at any given time. Flights need to be ordered by their departure time for our modelling and load function to be accurate. This means for any flight i, the only relevant flights for calculating workload are $j : j < i$. Ties for flights departing at the same time are broken arbitrarily.

2.3 Measuring Balance

A schedule defines one feasible assignment of flights to desks, and a set of schedules covering all flights creates a feasible solution to the flight dispatching problem. We need to evaluate these schedules and solutions to determine the best ones. The goal of this work is to create balanced schedules for flight dispatchers, however, there are many ways in which balance could be measured.

First, consider the balance between the dispatchers' workloads, as defined above. Workload balance could be measured as the total workload a desk will require throughout its duration, i.e., the total amount of work a dispatcher will need to do. However, some dispatchers may work longer shifts, so the total amount of work may not be the best metric of balance. Instead, the balance could be measured at every point in time, or in specific time epochs, for example, the hourly workload could be compared at each hour. To further consider balance, the entire dispatcher's shift could be considered. For example, a dispatcher may be expected to have a lighter load towards the beginning or end of their shift (while they are getting prepared or finishing/transitioning) than in the middle of their shift. This

could be measured if balance considers workload as a function of the dispatcher's shift time, and attempts to create fairly equal functions even if this means that at any given time there is a larger disparity in workload. Another metric is the maximum workload of each dispatcher, which would likely be the highest stress situation, and attempt to balance the maximum workload faced. Balance can be measured in any of these ways by finding the difference between the highest and lowest desk workloads, or by comparing each desk to the average and minimizing deviation. In the maximum workload scenario, each desk's workload is summed and minimized. Each of these metrics of balance may contradict another, so we must decide what is most appropriate for this work.

While workload is the main focus of this work, there are other components that could contribute to balancing dispatcher schedules. These elements include the number of total flights or the active flights at any given time, and the idle time (either cumulative or time-based). Regardless of the metric for balance, the way it is measured (e.g., cumulative or time-dependent) is important.

While the goal of this problem is to balance dispatcher schedules, the scope of this work focuses on desks rather than dispatchers. As such, we focus on minimizing the maximum workload of each desk. From the instance in Figure 2.2, we determine the maximum workload for a desk, denoted by F, by comparing the desk's workload at each time a flight is added, which in this case is 40.

$$F = max\{20, 40, 40, 25, 30\} = 40.$$

2.4 Conclusion

In this chapter, we define a schedule as the flights assigned to one desk. We present a network representation of the flight dispatching problem where each path from the source to sink node represents a schedule. Schedules do not have many constraints, so we test the effect of limiting which flights can be on the same schedule. The scope of our work focuses on the scheduling of desks and does not consider splitting desks into shifts. Load is defined as the amount of work required for a single flight at any given time, whereas workload is the total work required for all active flights on a desk at any given time. An instance with 10 flights is presented and used to demonstrate the flight network, potential schedules, and a feasible solution to the flight dispatching problem. Lastly, different metrics of balance used to evaluate flight dispatcher schedules are described. Potential measures of balance include total workload for a desk, desk workload as a function of time, and maximum desk workload. Variants of these measures are by number of total or active flights, or idle time. Balance can be measured through maximum desk differences, minimizing deviation, or minimizing each desk's worst case workload. We use the objective of minimizing the sum of maximum workloads over all desks.

Chapter 3

Literature Review

There is extensive operations research literature on the airline industry but little focus on dispatching. We start with an overview of optimization in airline operations planning and review problems related to scheduling. We then present a short overview of the bin packing problem and compare to the flight dispatching problem. Finally, we review applications involving workload balancing with a focus on nurse-patient-assignment, the balls-into-bins problem, and machine scheduling problems. Through this review, we discuss the similarities and differences between the flight dispatching problem and related problem classes.

3.1 Optimization in Airline Operations Planning

Airline Operations Planning includes four major airline problems that are typically solved sequentially: Schedule Generation, Fleet Assignment, Maintenance Routing, and Crew Scheduling (Barnhart et al., 2006). Schedule Generation determines which flights are

flown and at what times. These schedules allow for the formation of networks to solve the remaining problems. Fleet Assignment allocates the type of aircraft which will fly each of the scheduled flights. A network structure is used for Fleet Assignment where flight legs are represented by nodes for departure/arrival times and locations, with flight arcs considering maintenance time and ground arcs depicting idle aircraft (Barnhart et al., 2003). Maintenance routing assigns a specific aircraft to a sequence of flights while adhering to maintenance requirements. For maintenance routing, separate networks are formed by decomposing the flight network by type of aircraft required and they are typically solved using a network circulation model (Barnhart et al., 2003). Finally, Crew Scheduling assigns crews to a sequence of flights with restrictions from governing agencies, labour organizations, and airlines. Crew Scheduling is broken into the Crew Pairing Problem, where crews are assigned to flights, and the Crew Assignment Problem, where individual members are assigned to crews. Duty periods are a grouping of flights (typically within one day), and a sequence of duties along with layovers comprises a pairing. In the crew scheduling problem, a network structure is used for generating pairings, either in the form of a flight network - where there are nodes representing the departure and arrival of each flight and arcs represent possible connections between flights - or a duty period network - where nodes represent the start or end of a duty period and arcs represent possible overnight connections between duties (Barnhart et al., 2006).

The Crew Scheduling problem breakdown is similar to how the dispatching problem is broken down, with flights assigned to desks and then desks split into worker shifts. Airlines aim to create a balanced workload between crews through this scheduling (Barnhart et al., 2006). The assignment of flights to desks can be structured as a network where nodes represent flights, arcs represent flights that can depart sequentially on the same desk, and a path represents the flights on a single desk, similar to the flight network structure in

Crew Scheduling and Fleet Assignment.

Other models that incorporate uncertainty (Yen and Birge, 2006), (Rosenberger et al., 2002), delays (Shebalov and Klabjan, 2006), and robustness (Ehrgott and Ryan, 2002) have been explored that aim to minimize the actual costs incurred, rather than the planned costs.

Similar to Airline Operations Planning, the Flight Dispatching Problem is an airline problem. A key difference is the importance of location. In the planning problems, location is an important element that defines what is possible. Crews and aircraft take the flights, so they must service flights in a sequence where each flight departs from the same location the previous flight arrives. In dispatching, location may be used to decompose the problem, however, because dispatchers work centrally, there are no limitations on sequencing flights for a desk based on location. Dispatchers monitor multiple flights at once, whereas a crew or aircraft are only assigned to one flight at a time. A result of this is that a network structure may not be the best way to model the dispatching problem. This is because overlapping flights can be assigned to the same desk and there are few limitations as to which flights can be assigned consecutively to the same desk. This would likely create a dense network, and due to the structure of the maximum workload objective, it is not easy to dominate paths. In planning problems, the location and timing constraints limit the number of arcs in a network. Balance is measured in a different way, as for crew pairing is it based on number of hours per worker, whereas for dispatching it is based on the maximum workload for each desk.

The problem of flight dispatching has been described and modelled where flight dispatching is considered a component of a flight planning product from HPES-Agilaire called Workload Distribution (Santos et al., 2011). They consider flight legs over a 30 day horizon and use 20 to 30 desks at any given time. Four goals are considered, which are: evenness of desk workload at any time, consistency of schedules repeating daily, minimizing flight

splitting over multiple desks, and completeness of the assignment (not having unassigned flights). Schedules are created at the weekly level using a mixed integer formulation that is decomposed and solved heuristically.

They use a heuristic approach by decomposing their Mixed Integer Programming Model and solving a series of problems. To solve for up to a month, they create a "meta-week" to capture repeating flights, solving for the repeating flights and for the remaining flights. This varies from our approach of solving each decomposed problem for a day. Balance is measured in workload deviation per desk per hour, rather than as the sum of maximum workloads. Additionally, switching the desks that flights are assigned to is allowed, which we do not consider in our approach. We constrain that all flights must be scheduled, whereas they allow flights not to be scheduled at a penalty.

3.2 Bin Packing Problem

Workload balancing in flight dispatching has some similarities to the Bin Packing Problem (BPP). The BPP is a problem in which there is a set of items with varying sizes/weights that must all be packed into bins. The objective of the BPP is to minimize the number of bins required to pack all items (Delorme et al., 2016). In the standard formulation, all bins have the same capacity though there are problems that consider bins with varying capacity (Correia et al., 2008). The basic problem is in one dimension, with extensions to multiple dimensions, for example, three-dimensional pallet packing (Elhedhli et al., 2019). In the flight dispatching problem, desks are considered as bins and flights as items, with the load representing the weight/size. However, instead of minimizing desks subject to a fixed capacity, the number of desks is fixed with the objective of minimizing the maximum workload over desks. Alternatively, we could minimize the number of desks needed so

that no desk's workload exceeds some capacity, which would be more similar to the BPP. There would, however, still be differences based on the structure of taking the maximum workload over time, rather than a cumulative load measure.

The Generalized Bin Packing Problem (GBPP) allows for compulsory and non-compulsory items to be packed, where each provides some profit in addition to its weight and the goal is to minimize the net cost of selecting bins and packing items (Baldi et al., 2012). This could be useful if there were different classifications of flights - some that need to be dispatched and others that are not compulsory, for example, flights that might not be offered if they do not provide enough value. Scheduling of flights is not part of the dispatcher problem as it is part of the larger system of airline operations planning. To create this type of model, we would need to know the cost to open a desk and the value (e.g., revenue) from each flight.

Additional variants of the BPP include the Variable Size Bin Packing Problem (VSBPP) and the Variable Cost and Size Bin Packing Problem (VCSBPP) which allow for different capacities and costs to each bin (Crainic et al., 2019), which has been applied to machine scheduling (Correia et al., 2008). Currently, all dispatchers are considered equal, though it is possible to consider differences. For example, a dispatcher might be more efficient than others and therefore experience less load. These variants may be useful when considering shifts, where dispatchers are scheduled.

There is research on load balancing with the BPP, where it is applied to the Fractional BPP where an item may be split between multiple bins (Castro-Silva and Gourdin, 2019). This is not realistic to flight dispatching, as only one dispatcher monitors each flight at a time and if there is any flight splitting, it is split at the end of a shift rather than the load shared by multiple dispatchers at the same time.

The BPP provides a general structure that is used in flight dispatching, where items (flights) are assigned to bins (desks), however, the structure of the objectives and constraints are different. In the BPP, bins have a fixed capacity that is based on the cumulative weight of the items packed in it. Desks do not have a capacity and their workload is measured as a function of time, where we are concerned with the maximum workload. The objective in dispatching is to minimize a function of workload, whereas the BPP aims to minimize the number of bins needed. The GBPP, VSBPP, and VCSBPP incorporate variations on the BPP that could be incorporated into the Flight Dispatching Problem. These variations are outside our scope. The fractional BPP does not provide a structure that applies to load balancing in dispatching.

3.3 Workload Balancing in Other Industries

There are many problems with balanced-based objectives, often in server allocation problems. One type of allocation is assigning customers to employees, such as in mail order firms (Khouja and Conrad, 1995) and nurse-patient-assignment. Another type of allocation is work assigned amongst machines or stations, such as in machine scheduling problems, and balance in computers/servers. There are scheduling problems that include balance, such as the Balanced Academic Curriculum Problem (Hnich et al., 2002). Some of these problems are further described and compared to the flight dispatching problem.

3.3.1 Nurse-Patient-Assignment

Similar to flight planning, there are several stages to scheduling nurses in a hospital. These include budgeting, scheduling, rescheduling, and assignment (Acar and Butt, 2016),

(Rosenberger et al., 2014). Assignment, or nurse-patient-assignment, is the allocation of nurses as the primary caregiver to patients. Patients are categorized and require different amounts of care, so one way to allocate nurses to patients is to attempt to balance their overall workload. There are other elements to their job that affect their workload as well, such as the distance between patient rooms for a single nurse which affects the amount of time needed to switch tasks. There are constraints that limit combinations of patients to a single nurse (e.g., patients with certain conditions cannot be assigned to the same nurse). Scoring metrics have been proposed to balance the overall workload of nurses through nurse-patient assignment (Acar and Butt, 2016). Throughout a shift, patients may be discharged from or admitted to the system so revised assignments are needed, however, this typically only involves assigning incoming patients rather than reassigning a patient to a different nurse. An alternative balance metric is minimizing excess workload per nurse per time epoch. This means that nurse workload is a function of time (Rosenberger et al., 2014).

This problem is similar to the flight dispatching problem as there are workers and jobs that must be assigned, with the goal of balancing workload. The number of nurses is fixed, as this comes from a previous schedule, and the number of desks is predetermined. Nurse schedules are not made significantly in advance of when they occur as they are based on current needs and will change throughout a shift. Conversely, dispatch schedules are ideally made significantly in advance of their use. Nurse shifts are typically 12 hours in length and follow a set schedule, which could align with some assumptions of the flight dispatcher shifts which we do not consider in this work. The stochastic elements of nurse-patient-assignment could relate to flight delays or cancellations that are possible extensions to this work. In flight dispatching, limiting which flights can be assigned together is similar to limiting patients assigned to the same nurse. The key differences are the use of a scoring

metric to consider different elements of a nurse's work, as opposed to just the workload, and the stochastic nature of the problem.

3.3.2 Balls-Into-Bins

Computer science is another field containing many problems centred around load balancing. One adjacent problem is the Balls-Into-Bins problem. This is a problem in which there are m balls and n boxes or bins. Balls enter the system sequentially and must be added to a bin, using some random process (Berenbrink et al., 2013). By selecting different random processes for choosing bins for each ball, there are different limits on the maximum load that any bin might attain (Raab and Steger, 1998). Typically, balls have equal weighting, with some extensions including varied weighting. An example of this problem is when users need access to computers or servers that are allocated as users arrive, and users want to use the computers/servers with the least load and must be allocated accordingly. There is a trade-off between checking each machine and finding the minimal load, so the random process balances those needs (Berenbrink et al., 2013).

This is similar to the flight dispatching problem in that flights (i.e., balls) must be allocated to dispatchers (i.e., bins) with a general goal of balancing load. They both consider items (i.e., flights or balls) that are not equal in load. A key difference in these problems is that in the flight dispatching problem, all flights along with their load and departure/arrival times are known in advance, whereas the arrival time and load of balls are not known in advance. In dispatching, a model's output provides the assignment of all flights to desks. The solution for the balls-into-bins problem is a random process that is used to assign balls as they arrive.

3.3.3 Machine Scheduling Problems

The parallel machine scheduling problem assigns n jobs to m equivalent machines and determines the sequence of those jobs. Typically, the objective is to minimize the makespan or schedule length. Alternative criterion include balancing workload or workflow (often minimizing the difference between the highest and lowest machine loads or the normalized sum of squared workload deviations), or maximizing expected production rates (Rajakumar et al., 2004), (Schwerdfeger and Walter, 2018). Each job needs a specified amount of processing time and may have a release time, preceding tasks, and a target completion time. Due dates allow for objectives that minimize lateness of job completion (Ouazene et al., 2011). A lot of the existing research presents heuristics for solving these problems (Rajakumar et al., 2004), (Schwerdfeger and Walter, 2018), (Chen et al., 1998).

This problem is similar to flight dispatching in that jobs (i.e., flights) are assigned to servers (i.e., machines and desks) with a goal of balancing the load. There are many differences as well. Jobs in the machine scheduling problem are not scheduled at a specific time (though they may have a range) whereas flights are specifically scheduled to start at certain times with planned end times (we do not consider delays). Jobs measure their load by the amount of processing time needed, whereas flight load is determined by load requirements and is time-dependent (though it may represent the amount of time needed per hour to monitor). While both involve assignment, machines only process one job at a time whereas desks are intended to have multiple flights at any given time. The balance metrics are slightly different; in the machine scheduling problem, each machine has an overall load and the difference between the highest and lowest overall load is what determines balance. An alternative metric for balance that was proposed is minimizing the normalized sum of squared workload deviations. While these metrics could apply to flight

dispatching, the approach we have taken is to find the highest workload at any given time for each desk, and minimize the sum of those values.

Another adjacent problem is the Assembly Line Balancing Problem (ALBP) in which a series of tasks must be allocated to work stations. Tasks do not take the same amount of time and may have precedence, i.e., there may be some tasks that must be completed before others, so work stations are ordered. The system is limited to the slowest (highest load) station, and objectives to this problem include minimizing the number of workstations (subject to a maximum cycle time) and minimizing balance delay (Kriengkorakot and Pianthong, 2007). In general, it is about balancing work amongst the stations. This is similar to flight dispatching because there is work that must be divided into different stations/desks with a goal of either minimizing workers or balancing work subject to a number of workers. A key difference is the precedence of tasks requiring ordered work stations which is not needed for flight dispatching.

3.4 Conclusion

In the airline industry, the existing literature on Airline Operations Planning has a gap in dispatching. Planning problems as well as the Flight Dispatching Problem are at the airline level and can utilize a network structure. However, planning problems are dependent on location and resources are only assigned to one flight at a time. In dispatching, location does not constrain the problem (though it may be used to break it down), and desks monitor multiple flights at once. This means the network structure is less advantageous, and the main similarities are the industry. Instead of using a network structure, Bin Packing Problems are more similar in structure where flights are packed into desks. The differences are in the measure of load, which is cumulative in bin packing rather than a

function of time, and that the BPP minimizes bins subject to capacity whereas the Flight Dispatching Problem minimizes workload with a fixed number of desks.

Workload balancing is prevalent in other fields, such as nursing, computing, and machining, however, they use different definitions of balance, and many of these problems have different structures that cannot be applied to dispatching. Nurses monitor multiple patients over a shift, and consider balance through a scoring metric or through excess workload per time period. In the Balls-Into-Bins problem, the goal is to evenly distribute the balls between bins, with decisions that are made sequentially using a random process as balls arrive. The parallel machine scheduling problem assigns jobs amongst a fixed number of machines with the goal of minimizing schedule length. Alternative objectives consider balancing machine load through minimizing deviations or the difference between the highest and lowest machine load. In machine scheduling, each machine only processes one job at a time and they are not time dependent. The Assembly Line Balancing Problem is similar in that tasks are assigned to workstations with objectives to minimize the number of workstations or balance delay. While each of these problems have balance considerations, none of them measure it using the sum of maximum workloads, and there are other structural differences between these problems. Nurse-Patient-Assignment is the most closely related problem to flight dispatching.

In the next chapter, formulations are presented along with the solution methodology used to solve the flight dispatching problem.

Chapter 4

Modelling and Solution Methodology

Two different formulations for creating flight-dispatcher assignments are provided and then solution methodology is discussed. In Section 4.1, the full formulation for the Flight Dispatching Problem is presented. A set covering formulation is then presented in Section 4.2, and the solution by column generation for that formulation is presented in Section 4.3. Additional constraints to the formation of schedules are modelled in Section 4.4.

4.1 Flight Dispatcher Scheduling Formulation

The following formulation assigns flights amongst a fixed number of desks, so that each flight is assigned to exactly one desk, with the objective of minimizing the maximum workloads. Each desk is denoted by $k \in K$ and has a minimum requirement of one flight assigned to it. Otherwise, it would be equivalent to the desk not being used. We want to force using the prescribed number of desks, however, we test the effect this has on the solutions by varying the number of desks. Flights are indexed by $i \in I$, and the

information needed about flights are the departure time, S_i, and the load required at each time t throughout the duration of the flight, f_{it}, until the arrival time of the flight. I is ordered by increasing S_i and ties for departure time are broken arbitrarily. This allows us to only consider flights indexed before flight i in the calculation of load.

The decision variables needed are presented below, followed by the Flight Dispatcher Schedule Formulation (FDSF).

$$x_{ik} = \begin{cases} 1 & \text{if flight } i \in I \text{ is assigned to desk } k \in K \\ 0 & \text{otherwise} \end{cases}$$

F_{ik} The workload for desk $k \in K$ at the start of flight $i \in I$ if it is added to the desk

F_k The maximum workload of desk $k \in K$ at any time

$$\text{[FDSF]: } \min \sum_{k \in K} F_k$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1 \qquad \forall i \in I \qquad (4.1)$$

$$F_{ik} = \begin{cases} f_{iS_i} + \sum_{j<i} f_{jS_i} x_{jk} & \text{if } x_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in I, k \in K \qquad (4.2)$$

$$F_k = \max_{i \in I} F_{ik} \qquad \forall k \in K \qquad (4.3)$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in I, k \in K \qquad (4.4)$$

$$F_{ik} \geq 0 \qquad \forall i \in I, k \in K \qquad (4.5)$$

$$F_k \geq 0 \qquad \forall k \in K \qquad (4.6)$$

Constraint 4.1 ensures that each flight is assigned to exactly one desk. The workload

of a desk is constrained by 4.2, which takes the active workload of each desk at the start of flight i if it assigned to that desk, or 0 otherwise. Constraint 4.3 takes the maximum workload for each desk. Constraints 4.4 - 4.6 are the binary and non-negativity constraints.

To linearize this model, Constraints 4.2 and 4.3 are changed to the following:

$$F_{ik} \geq f_{iS_i} + \sum_{j<i} f_{jS_i} x_{jk} - M(1 - x_{ik}) \qquad \forall i \in I, k \in K \qquad (4.7)$$

$$F_{ik} \leq M x_{ik} \qquad \forall i \in I, k \in K \qquad (4.8)$$

$$F_k \geq F_{ik} \qquad \forall i \in I, k \in K \qquad (4.9)$$

where M is a large constant. M sets an upper bound on a desk's workload. To create a tighter bound for M instead of using an arbitrarily high number, a dummy schedule is created with all flights scheduled to one desk. The workload of this schedule is then found, and it is the maximum possible workload for a desk. This becomes the value of M, as a larger value is never needed to find appropriate bounds on F_{ik}. Constraints 4.7 and 4.8 are the upper and lower limits to calculate the correct workload value of a desk. When a flight is not assigned to a desk, there is no workload assigned.

Constraints 4.7 and 4.8 do not, however, tightly constrain the workload actually incurring by desk k when i is allocated to it. Instead it creates a range on F_{ik} with a lower bound of

$$max \left\{ f_{iS_i} + \sum_{j<i} f_{jS_i} x_{jk} - M(1 - x_{ik}), 0 \right\}$$

and an upper bound of

$$min\{F_k, M x_{ik}\}.$$

This forces F_{ik} to 0 when the flight is not added to the desk, and therefore bounds F_{ik} by

$$f_{iS_i} + \sum_{j<i} f_{jS_i} x_{jk} \leq F_{ik} \leq F_k$$

when the flight is added to the desk. The limit is F_k instead of M because any value up to F_k does not affect the objective.

This model is solved by CPLEX on instances from two datasets with a time limit of two hours. For small instances, this model typically solves optimality within minutes. For larger instances, the gap between the lower and upper bounds remains large after two hours. These results are further discussed in Chapter 6. To solve larger instances, an alternative formulation is presented.

4.2 Set Covering Formulation

We propose a set covering formulation amenable to column generation to solve large scale instances. Each possible allocation of flights to a single desk is called a schedule, $h \in H$, and has a known maximum workload, C_h, and α_{ih} denotes the flights, $i \in I$, allocated to each schedule. $\alpha_{ih} = \begin{cases} 1 & \text{if flight } i \in I \text{ is on schedule } h \in H \\ 0 & \text{otherwise} \end{cases}$

C_h is calculated in a similar way as F_k, replacing the decision variable x_{ik} with the parameter α_{ih}.

$$C_h = \max_{i:\alpha_{ih}=1} \left\{ f_{iS_i} + \sum_{j<i} f_{jS_i} \alpha_{jh} \right\} \quad \forall h \in H$$

The total number of desks, or schedules selected, is noted by m, where $m = |K|$. The new decision variable is:

$$y_h = \begin{cases} 1 & \text{if schedule } h \in H \text{ is selected} \\ 0 & \text{otherwise.} \end{cases}$$

The Integer Problem (IP) is a set covering formulation that uses schedules to cover all flights.

$$\begin{aligned} \text{[IP]: } \min \quad & \sum_{h \in H} C_h y_h \\ \text{s.t.} \quad & \sum_{h \in H} \alpha_{ih} y_h \geq 1 && \forall i \in I && (4.10) \\ & \sum_{h \in H} y_h = m && && (4.11) \\ & y_h \in \{0, 1\} && \forall h \in H && (4.12) \end{aligned}$$

The objective minimizes the cost of the selected desks. Constraint 4.10 ensures that each flight is assigned to at least one schedule, and the number of schedules chosen is constrained by 4.11. Constraint 4.12 is the binary constraint.

By the problem definition, each flight must be assigned to exactly one desk. This would be a set partitioning formulation, where Constraint 4.10 is set to equality. In practice, we use the set covering formulation because if any flights are assigned to multiple desks, they cannot affect the maximum workload of any of those desks. Otherwise, an optimal solution would be found in which the flight is removed from the desk where it affects the maximum workload. If a solution is found with a flight assigned to multiple schedules, this is addressed in post-processing.

4.3 Solution by Column Generation

To solve the IP, a column generation approach is used. We need to generate the schedules in set H. Since the number of feasible schedules is very large we generate schedules iteratively. The IP is relaxed to form the Relaxed Master Problem (RMP). This solves over only the set of known schedules, $\overline{H}$, and relaxes y_h to be continuous in the interval $[0, 1]$ rather than binary.

$$\text{[RMP]: } \min \sum_{h \in \overline{H}} C_h y_h$$

$$\text{s.t.} \quad \sum_{h \in \overline{H}} \alpha_{ih} y_h \geq 1 \qquad \forall i \in I \tag{4.13}$$

$$\sum_{h \in \overline{H}} y_h = m \tag{4.14}$$

$$y_h \leq 1 \qquad \forall h \in \overline{H} \tag{4.15}$$

$$y_h \geq 0 \qquad \forall h \in \overline{H} \tag{4.16}$$

Its dual problem is:

$$\text{[DP]: } \max m\lambda_0 + \sum_{i \in I} \lambda_i$$

$$\text{s.t.} \quad C_h - \sum_{i \in I} \alpha_{ih} \lambda_i - \lambda_0 \geq 0 \qquad \forall h \in \overline{H} \tag{4.17}$$

$$\lambda_i \geq 0 \qquad \forall i \in I \tag{4.18}$$

The dual variable λ_i corresponds to each flight $i \in I$, and corresponds to constraints

4.13. The dual variable λ_0 corresponds to the total number of desks used, and corresponds to constraint 4.14.

The column generation algorithm starts by solving the RMP on a subset of schedules $\overline{H} \subseteq H$ and then checks to see if the solution is optimal to the original problem, IP. If it is not optimal, a new schedule is generated by solving a pricing problem (PP):

$$\text{[PP]: } \min F - \sum_{i \in I} \overline{\lambda}_i x_i$$

$$\text{s.t.} \quad F_i \geq f_{iS_i} + \sum_{j<i} f_{jS_i} x_j - M(1 - x_i) \qquad \forall i \in I \qquad (4.19)$$

$$F_i \leq M x_i \qquad \forall i \in I \qquad (4.20)$$

$$F \geq F_i \qquad \forall i \in I \qquad (4.21)$$

$$\sum_{i \in I} x_i \geq 1 \qquad (4.22)$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \qquad (4.23)$$

$$F_i \geq 0 \qquad \forall i \in I \qquad (4.24)$$

$$F \geq 0 \qquad (4.25)$$

The PP creates a schedule for one desk. This is similar to the FDSF, without the need for the desk index k. Since there is only one desk, we are assigning flights to that desk, weighing the workload against the value of the flights added. There must be at least one flight in any schedule.

If the objective value of the PP minus λ_0 is negative, then it it not optimal to the IP. The PP solution defines a new schedule that is added to $\overline{H}$, and the process is restarted.

Schedules are added to $\overline{H}$ until no schedule with negative reduced cost is identified, or the relative gap between the upper bound (UB) and lower bound (LB) is within an acceptable range, e.g., 10^{-6}. The lower bound of the RMP at iteration n is calculated by

$$\max\left\{LB_{n-1}, m\pi + \sum_{i\in I}\overline{\lambda}_i\right\},$$

where π is the objective value of the PP. The UB is the most recent objective value of the RMP. Since $\overline{H}$ is updated at every iteration, the UB either decreases or stays the same.

The algorithm for column generation is presented in Algorithm 1. This is referred to as the base implementation. The column initialization is described in Chapter 5. Once the column generation algorithms stops, the IP is solved on $\overline{H}$.

Algorithm 1 Pseudo code for Column Generation

Initialize columns, $\overline{H}$, and form RMP
$UB \leftarrow \infty$ ▷ Upper bound
$LB \leftarrow 0$ ▷ Lower bound
Main loop
while UB $\neq$ LB **do**
  Solve RMP ▷ obtain $\overline{\lambda}_i$, $\overline{\lambda}_0$
  $UB \leftarrow$ RMP objective ▷ Update UB
  Solve PP using $\overline{\lambda}_i$ ▷ obtain solution $\overline{x}_i$
  $LB \leftarrow max\{LB_{n-1}, m\pi + \sum_i \overline{\lambda}_i\}$ ▷ Update LB
  where π is the objective value of the PP $= F - \sum_i \overline{\lambda}_i x_i$
  if $\overline{C}_h - \sum_i \alpha_{ih}\overline{\lambda}_i - \overline{\lambda}_0 \geq 0$ **then**
  optimal, stop.
  else Add new column, h, defined by $\alpha_{ih} = \overline{x}_i$ to $\overline{H}$
  Add load of new column to C_h
  end if
end while

4.4 Adding Flight Overlap Constraints to Schedules

In section 2.1, the only constraint for schedules is that they cannot be empty which we model with constraint 4.22. As more work is involved during the planning period which occurs at the beginning of a flight, limiting flights with similar departure times from being monitored by the same desk may lead to more balanced schedules. A second PP formulation is presented in which flights departing within some time interval cannot be assigned to the same desk. We define τ as the time interval for which flights cannot be dispatched together. For example, if $\tau = 0$, then flights departing at the same time cannot be scheduled to the same desk, and if $\tau = 5$, then flights departing within five minutes of each other cannot be scheduled to the same desk. This is used to determine the values of a new parameter:

$$\beta_{ij} = \begin{cases} 1 & \text{if flight } j \in I \text{ can follow flight } i \in I \text{ based on } \tau \\ 0 & \text{otherwise.} \end{cases}$$

Then, we use the flight overlap constraint

$$x_i + x_j \leq 1 + \beta_{ij} \qquad \forall i \in I, j \in I, j > i \tag{4.26}$$

This leads to a minimum number of desks needed to create a feasible solution, where the minimum number of desks is equal to the maximum number of flights that depart at the same time. Alternatively, we could limit the number of flights departing within an interval of time. We denote n as the time interval and V as the maximum number of

flights allowed. The constraint then becomes

$$\sum_{j:S_i \leq S_j \leq S_i+n} x_j \leq V \qquad \forall i \in I \tag{4.27}$$

While this constraint only considers flights departing from the start of i to n minutes later, it does constrain any n minute period through the constraint on previous flights. In this work, we test the effect of the flight overlap constraint that limits which flights can be scheduled together.

Adding constraints to the PP does not change the solution methodology, except for the initialization of columns. The next chapter describes the column initialization for the base implementation, as well as the adjustments needed to model the flight overlap constraints.

4.5 Conclusion

In this chapter, we provide the full flight dispatcher schedule formulation, as well as a set covering formulation using schedules. To solve, a column generation approach is taken and the algorithm is presented as well as the additional models needed to follow the algorithm. This is the base implementation. Flight overlap constraints which limit the flights that can be assigned to the same desk are modelled. Heuristic approaches to initialize the algorithm and attempt to improve solutions are discussed in the next chapter.

Chapter 5

Heuristics

Heuristics are used in this work to initialize the columns for the column generation algorithm, and to try to improve upon the integer solutions found. The initialization and variants for the flight overlap constraints are first presented followed by the improvement algorithm.

5.1 Initialization of Column Generation Algorithm

To initialize $\overline{H}$, a greedy algorithm is used to create the first m columns and a swapping method is used to create additional columns based on the first m created. The first m columns create a feasible solution to the IP and RMP. The pseudo code of this algorithm is shown in Algorithm 2. The indexing used for this algorithm begins at 0. DeskA and DeskB are the columns being swapped while DeskC and DeskD are the output of the swap. The pairs for swapping are defined at the end of the algorithm.

Using the data from Table 2.1, the greedy heuristic is shown for three desks in Table

Algorithm 2 Pseudo code for Column Initialization

Greedy Heuristic
for i from 0 to $m-1$ **do**
    Assign flight i to desk i
end for
for i from m to $|I|-1$ **do**
    Calculate the active load of each desk
    If necessary, update each desk's feasibility of adding the current flight
    Add flight i to the feasible desk with the lowest active load
end for
Swapping Algorithm
while Swapping Pairs Exist **do**
    for i from 0 to swapIndex-1 **do**
        DeskC[i] = DeskA[i]
        DeskD[i] = DeskB[i]
    end for
    for i from swapIndex to $|I|-1$ **do**
        DeskC[i] = DeskB[i]
        DeskD[i] = DeskA[i]
    end for
end while
Swapping Pairs
for i from 0 to $|I|-1$ **do**
    Swap columns i and $i+1$
end for
Swap columns 0 and $|I|-1$
for i from 0 to $|I|/2$ **do**
    Swap columns i and $|I|-i$
end for

5.1. The first three flights (A, B, and C) are allocated to the three desks respectively. Each additional flight is added sequentially to the desk with the lowest active load at the time of the new flight. Flight D is the first additional flight and it departs at time 800, so the active flights for desk 1, 2, and 3 are 15, 10, and 15 respectively. This means desk 2 has the lowest active load and flight D is added to this desk. Moving forward, flights B and D both contribute to the load of desk 2. This guarantees a feasible solution to the base RMP and IP in which these m columns are selected.

		Desk 1		**Desk 2**		**Desk 3**		
i	S_i	**Flights**	**Load**	**Flights**	**Load**	**Flights**	**Load**	**Min Desk**
D	800	A	15	B	10	C	15	2
E	1000	A	0	B, D	30	C	15	1
F	1100	A, E	15	B, D	30	C	15	1
G	1300	A, E, F	35	B, D	10	C	0	3
H	1400	A, E, F	25	B, D	10	C, G	15	2
I	1500	A, E, F	25	B, D, H	25	C, G	15	3
J	1800	A, E, F	15	B, D, H	15	C, G	20	1

Table 5.1: Example of initialization algorithm with $m = 3$.

Additional columns or schedules are created using a swapping algorithm. This algorithm takes two columns, say column A and B, and an index. Two new columns are created that contain the flights from column A until the index and column B after, and vice versa. An example is shown in Figure 5.2 where desks 1 and 3 are swapped from flight F and beyond (the middle index).

Any columns can be swapped but to obtain a sufficient number of columns and ensure each of the initial desks is used in a swap, the swaps are performed with:

1. Subsequently indexed columns, I.e., columns 0 and 1, columns 1 and 2, ..., columns $m-1$ and 0.

2. Opposite indexed columns, I.e., columns 0 and $m-1$, columns 1 and $m-2$ etc. With

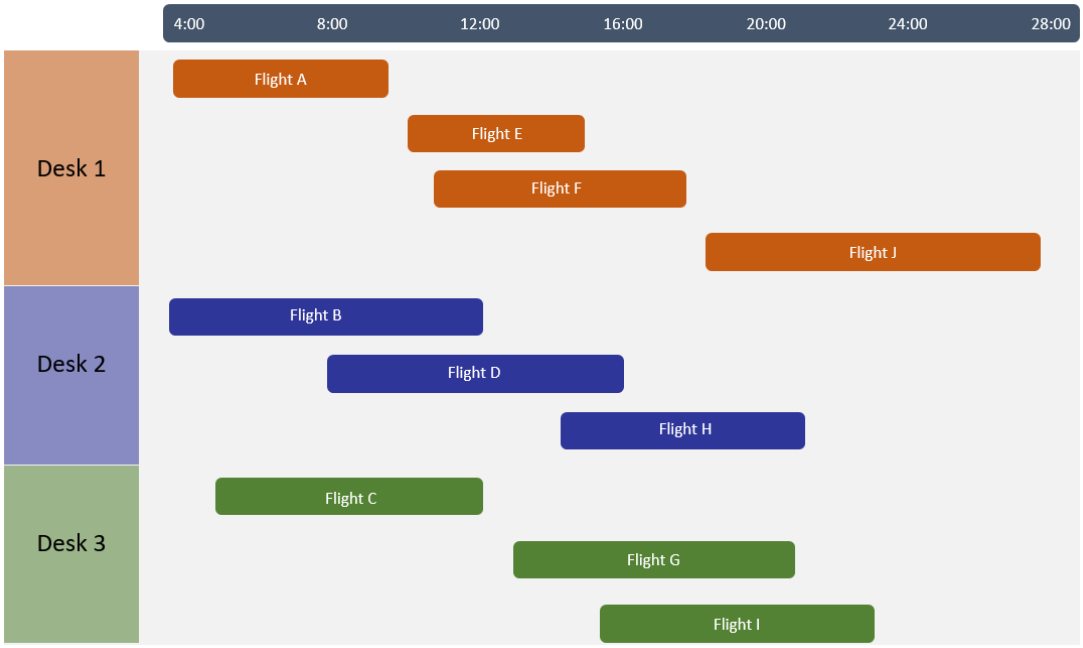


Figure 5.1: Columns formed using greedy algorithm.

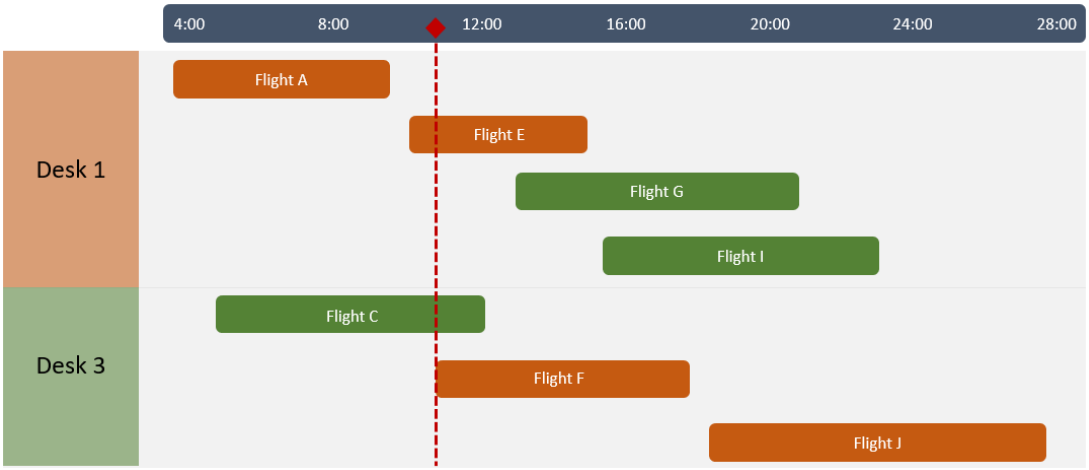


Figure 5.2: Columns formed by swapping desks 1 and 3.

an odd m, the column indexed by $\lfloor \frac{m}{2} \rfloor$ is not swapped.

If there are additional constraints in defining schedules, this heuristic needs to be modified accordingly. When schedules cannot contain flights departing at the same time, the number of desks must satisfy the minimum requirement which is equal to the maximum number of flights departing at the same time. If there are fewer desks, then a feasible solution is not possible. Otherwise, the greedy heuristic follows the same logic but with an additional feasibility check. In addition to the previous flights assigned and the active load, each desk keeps track of the departure time of the most recent flight added to that desk. If this departure time is within τ of the new flight's departure time, this is not a feasible desk for that flight. The flight is added to the desk with the lowest active load amongst those that are feasible.

The swapping algorithm needs to be modified to ensure that swaps do not results in multiple flights departing within τ on the same desk. To do this, the swap index may be modified. The departure time of the index is compared to the previous flight's departure time. If they are within τ, the index is increased by one and checked again. Once the index has a sufficient difference in departure time from the previous flight, it is used for all swaps. If this is not possible, checks for flights around the swap index would need to be performed post-swap.

Similar modifications would need to be made in the case where the number of flights within a time interval is limited. For the greedy heuristic, instead of tracking last departure time, the number of departures within the specified interval would be tracked for each desk. This is what would define feasibility. The swapping may require post-processing to eliminate any intervals exceeding the allowed number of flights, If there are consecutive flights whose difference in departure times is greater than the interval, the latter flight is used as the swap index and post-processing is not required.

5.2 Improvement Algorithm

The improvement algorithm takes a feasible solution and attempts to decrease overall workload by swapping out flights at the bottleneck of each desk. The bottleneck for each desk is the flight that produces the highest F_{ik} value. Swaps first try simply removing a flight from one desk (say Desk A) and adding it to another (say Desk B). If this does not improve the workload, then a flight is swapped from Desk B to Desk A. The flight removed from B is taken from the bottleneck, but when this is the same flight that was just added, the previous active flight is swapped instead. When an improvement is found, the incumbent is updated and the algorithm is reset since new desks have been created. Otherwise, the next pair of desks is checked. The pseudo code for this heuristic is found in Algorithm 3.

5.3 Conclusion

In this chapter, we describe the initialization of columns which uses a greedy heuristic followed by a swapping algorithm. These are used to start the column generation procedure. An improvement algorithm that swaps flights from each desk's bottleneck is described. The results for each implementation are presented in the next chapter.

Algorithm 3 Pseudo code for Improvement Algorithm

```
    Improvement Heuristic
 1: Start with an incumbent solution
 2: for each desk k do
 3:     Check that there are enough flights on the desk to perform a swap
 4:     Find the bottleneck index for desk k, b_k
 5:     for each desk h ≠ k do
 6:         Switch the bottleneck flight from desk k to desk h
 7:         Calculate the new sum of workloads
 8:         if Workload decreases then
 9:             Update the incumbent with the new schedules
10:             Reset the algorithm
11:         else
12:             Get the bottleneck index of desk h, b_h.
13:             if b_h = b_k then
14:                 Set b_h to the previous flight assigned to desk h.
15:             end if
16:             Switch the bottleneck flight from desk h to desk k
17:             Calculate the new sum of workloads
18:             if Workload decreases then
19:                 Update the incumbent with the new schedules
20:                 Reset the algorithm
21:             else
22:                 Reject swaps. Do not update incumbent.
23:             end if
24:         end if
25:     end for
26: end for
```

Chapter 6

Testing and Results

In this chapter, results from testing are presented to compare the performance of the implementations discussed in Chapter 4. In Section 6.1, we discuss the datasets that are used for testing instances of the model. One dataset is from European Airline Data and another is from American Airlines, which is broken into several instances based on flight locations. The testing focuses on instances with 46 to 163 flights, and explores the limitations on instance size. Testing results are shown in Section 6.2 and includes parameter tuning on CPLEX parameters, a comparison of the models and implementations, analysis on the effect of the number of desks, the performance with the flight overlap constraints, and results using varied load. We compare solutions by looking at computational performance and solution quality.

6.1 Data

There are two sets of data used for testing, one from American Airlines (AA) and the other from European Airline Data (EAD). The AA dataset is broken down to obtain instances with varying numbers of flights per day.

6.1.1 American Airlines Dataset

The AA dataset is available on Kaggle (Mondal, 2019) and comes from the US Department of Transportation. It contains flight data from 2015 for airlines in the US, with data for airlines, airports, and flights. It has been used in many analyses, primarily focused on analyzing and predicting flight delays. As each airline manages its own dispatchers, a single airline is used for this analysis.

AA is chosen for this analysis because it is an airline with a high number of flights and some data about their operations is available online. For the US, AA is the airline with the largest market share of US domestic flights in 2020 (Mazareanu, 2021b), with the American Airlines Group offering around 6700 flights per day across 50 countries (American Airlines), (Otley, 2018). From the dataset, AA flights are taken from a single day in which there are 1473 flights, all of which are within the US and its territories.

Additional processing is required for testing the flight dispatching problem. The times are first converted to be in the 24-hour format. In one day, flight departure times are found between 5:00 and 23:55 without any significant gaps, but there are flights departing between 0:00 and 2:00. Instead of considering these flights to be before 5:00 on the same calendar day, they are considered to occur after midnight and are given times from 24:00 to 26:00. This means the start of the "day" is between 2:00 and 5:00. The flight times are

then converted to minutes elapsed since 0:00, so 5:00 is at 300 minutes.

Dispatch zones can easily be formed by finding regions, as each airport's state is known. US states can be categorized into regions based on service areas outlined by the Federal Aviation Administration (FAA) (Federal Aviation Administration, 2021). These regions are Eastern, Central and Western. The US territories are categorized separately. Since there are few of these flights, they are removed for this testing. There are 36 flights departing or arriving in these territories, so the remaining number of flights is 1437. The regions of the origin and destination airports are used to determine the dispatch zone, which allows the data to be decomposed and solved separately for each zone. Flights are grouped by the pair of their origin and destination airports, but direction is not important. For example, a flight departing in the East and arriving in the Central, or departing in the Central and arriving in the East would both be classified as Central-East (CE). International flights would be served by dispatchers who are classified as international dispatchers so that would be a separate instance and is not considered for this analysis (but the process would be the same).

The instances by zones are displayed in Table 6.1, with the time window of flights and load statistics for each zone included.

	Number of Flights	**Earliest Departure**	**Latest Arrival**	**Minimum Load**	**Maximum Load**	**Average Load**
CC	297	345	1480	1	10	4.63
CE	434	315	1505	4	16	10.24
CW	369	330	1865	5	37	12.63
EE	163	310	1487	2	13	8.08
EW	124	370	2018	17	27	22.61
WW	50	410	1840	2	26	13.44

Table 6.1: American Airlines zonal data.

6.1.2 European Airline Dataset

The European Airline Dataset (EAD) contains 173 flights over four days, and it is broken down into four days with almost identical schedules (Lu and Gzara, 2015). Days 2 and 3 are equivalent, and the primary differences are missing flights at the beginning of day 1 and the end of day 4.

Departure and arrival times are known and given in minutes elapsed since midnight of day 1 but can be adjusted for each day. Using day 2, there are 46 flights departing between 270 and 1150 which is just below a 15-hour range.

Specific airline information is not included in the data but the flights span seven airports. This data has been used for crew pairing in airline scheduling, which does not need specific airport information and instead needs potential connections.

6.1.3 Other Parameters

In the flight dispatching problem, the number of desks is an input to the model, and we can test different desk quantities to compare results. For this testing, we run each instance with three values for the number of desks. These are determined by targeting the average number of flights per desk to 15, 20, and 25 flights, and adjusting the desks to rounded, unique numbers for each instance, with the final numbers shown in Table 6.2.

Two versions of load are tested. The first is where load is constant throughout the duration of the flight (using the planning load). The second uses the post-planning load after the first hour of the flight. Load is only considered to start at the departure time of the flight. In reality, the work would start before the flight departs - often three hours prior to departure. To capture this work, the modelling would not change but the flight

	25	**20**	**15**
46	2	3	4
50	2	3	4
124	5	6	8
164	7	8	11
297	12	15	20

Table 6.2: Target number of desks per instance.

departure times would be earlier and the time in the planning period would be increased.

Load is not included in either dataset, so we estimate load values. The loads for the AA flights are estimated by taking a portion of the distance travelled by each flight (which is related but not directly proportional to flight duration). The planning load ranges from 2 to 37 per flight. Post-planning load is calculated based on the planning load.

$$\text{Post-Planning Load} = \max\left\{\left\lfloor \frac{\text{Planning Load}}{2} \right\rfloor, 1\right\}$$

This results in longer flights having a higher load. It does make sense that longer flights may require more load because there is more to consider, e.g., more climates/regions that would be passed through. However, this does not take into consideration that flights with greater distance are longer in duration so while the overall load may be greater, it does not necessarily mean that the load per hour should be significantly different. The planning loads for EAD are estimated using a range of 5 to 20, and have an average load of 11.98. Post-planning loads are calculated the same way as with the AA data.

We model load as a unitless measure, however, load can represent the amount of attention in minutes per hour that the flight requires from a dispatcher (Santos et al., 2011). In their report, there is only a single sample datapoint, in which the flight load is 1 minute. A dispatcher would have a standard capacity of 60 minutes of load per hour but may be

able to take on a limited amount of overage. To incorporate this type of load, we would need to consider each hour of work rather than snapshots in time.

6.2 Testing

Testing is focused on four instances: three from AA and one from EAD. The AA instances used are East/East (EE), East/West (EW), and West/West (WW). Some testing is also performed on the Central/Central (CC) instance to explore implementation limitations. We solve the FDSF by CPLEX, and the IP by column generation with multiple columns populated, which we refer to as the base implementation. Differences in performance are explored using the flight overlap constraints and by modifying the number of desks. Most instances use constant load, and we present differences when using varied load. Models are coded in C++ Visual Studio 2019 and solved using CPLEX version 20.1.0 on a 64-bit Windows 10 with Intel(R) Core(TM) i7-4790 3.60GHz processors and 8.00 GB RAM. In most cases, a two-hour time limit is used.

Statistics are defined in Table 6.3, in which gaps are calculated by the following equation, where x represents the objective value of the appropriate model

$$\frac{x - LB}{LB}.$$

The gap values given are the percent gap. The Total-T, IH-T, and FDSF-T are given in seconds, whereas other times are given as the percentage of time spent in that function.

S	The solution pool populate parameter that limits the number of solutions stored in the solution pool
PopGap	The solution pool relative gap parameter that limits the gap in objective in the solution pool
NbCol	Total number of columns generated
Iter	Number of iterations of column generation
RMP-T	Percentage of time solving the RMP
PP-T	Percentage of time solving the PP
IP-T	Percentage of time solving the IP
Other-T	Percentage of time solving in other processes
Total-T	Total time in seconds to run column generation implementation
IH-T	Time in seconds to run the Improvement Heuristic (IH)
FDSF-T	Time in seconds to run the Flight Dispatcher Schedule Formulation (FDSF)
LRB	Lagrangian lower bound
IP-T	Objective value of the IP solution on $\bar{H}$
Gap	Percent gap between IP and LRB
D-Min	Minimum workload over all desks
D-Max	Maximum workload over all desks
D-Avg	Average workload over all desks
IH	Objective value of the IH solution
I-Gap	Percent gap between IH and LRB
FDSF	Objective value of the FDSF solution
F-Gap	Percent gap between FDSF objective and best bound (LB)

Table 6.3: Problem parameters and solution statistic definitions.

6.2.1 Parameter Tuning

There are several parameters in CPLEX that affect the column generation (CG) procedure. These include S, the solution pool populate parameter that limits the number of solutions stored in the solution pool, and *PopGap*, the solution pool relative gap parameter that limits the gap in objective in the solution pool. This means these parameters affect how many additional columns can be added in each iteration (however, this is an approximate limit) and how close in objective to the optimal column they must be. Table 6.4 shows results from solving the Set Covering Formulation on the EW instance with six desks. Modifying the parameters S and *PopGap* has an effect on solution time but not solution quality. In each case, the IP objective value is 983 and the MP objective value is 981 for a gap of 0.02%. Additionally, in each solution the minimum desk workload is 154, the maximum desk workload is 178, and the average desk workload is 163.83. The overall

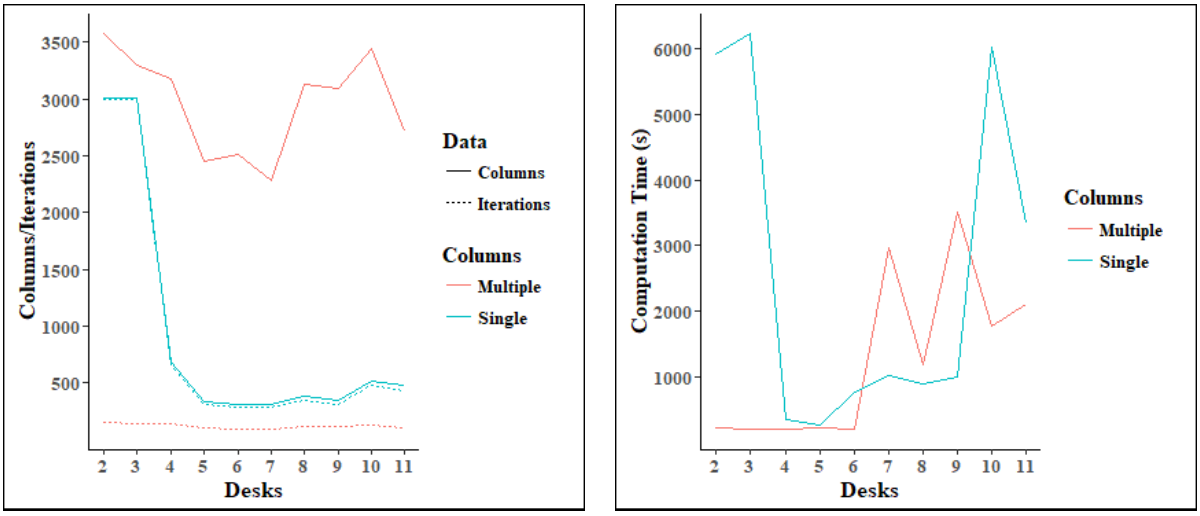
solution time is displayed, as well as the percentage of that time spent in each part of the problem. The number of columns generated and iterations are affected. For the EW instance, the fastest solution time came with $S = 15$ and $PopGap = 0.05$.

S	***PopGap***	**NbCol**	**Iter**	**RMP-T**	**PP-T**	**Pop-T**	**Other-T**	**Total-T**
5	0.01	2106	240	0.84	94.82	3.32	1.02	5922.4
	0.05	2182	216	0.58	97.43	1.15	0.84	8280.0
	0.10	2270	197	0.50	97.64	1.00	0.86	7920.0
10	0.01	3292	210	1.52	89.73	6.70	2.06	4640.0
	0.05	3042	171	1.76	91.44	3.65	3.15	3040.0
	0.10	3002	156	1.13	94.47	2.16	2.24	4170.0
15	0.01	4027	194	3.28	58.80	33.10	4.81	2636.2
	0.05	3091	129	1.62	90.23	4.31	3.84	2582.5
	0.10	3400	123	0.81	95.62	1.57	2.00	5320.0

Table 6.4: Parameter tuning on EW instance with six desks.

We compare to the implementation without using populate, so only one column is generated in each iteration. Several problems using the EW instance do not solve within the two-hour time limit, so we compare the results of EE instance with a varying number of desks.

Figure 6.1 compares the results when a single column is added in each iteration of column generation versus when additional optimal or near-optimal columns are populated in each iteration. The solution values are equivalent (e.g. same objective value, but possibly different columns chosen) for both cases, so we compare the computational performance. Figure 6.1a shows the total number of columns and iterations in each case. When only a single column is added, the only difference between columns and iterations comes from the initial columns generated. When multiple columns can be added, the total number of columns generated is greater than when only one is added each time, however, the number of iterations is lower and are fairly consistent over the number of desks. Similar trends are



(a) Number of columns and iterations.

(b) Computation time.

Figure 6.1: Effect of populating multiple columns versus a single Column.

present with other instances. Additionally, in Figure 6.1b, we compare the computation time. For some problems, adding a single column is faster whereas for others, adding multiple columns is faster. In this instance, adding multiple columns is better overall and there are greater potential gains from generating multiple columns than just one. The data for the figures can be found in Tables A.9 and A.10. The remaining tests use the populate function with parameters $S = 15$ and $PopGap = 0.05$ where applicable.

6.2.2 Testing of Base Column Generation Implementation

Each of the instances are tested on the set covering formulation. We solve the LP relaxation of the IP using column generation to obtain a lower bound. Then the IP is solved on $\overline{H}$ to obtain a feasible solution, which is an upper bound to the IP. The base implementation uses column generation with multiple columns populated and no flight overlap constraints. The IP bounds are compared to the improvement heuristic (IH) on the feasible solution

obtained by solving the base implementation, and the flight dispatcher scheduling formulation (FDSF) solved directly by CPLEX. For the column generation implementations, the column generation procedure has a two-hour time limit. At that point the IP is solved. The Lagrangian lower bound (LRB) is displayed, and when the problem is solved within two-hours, this is equivalent to the upper bound of the Relaxed Master Problem (RMP UB). The FDSF is solved with a two-hour time limit, so tests that reached optimality in less than that time have a gap of 0. Otherwise, the best feasible solution is shown along with the optimality gap after two-hours. The IH does not have any time limits as it solves within seconds.

The results for the base implementation are shown in Tables 6.5 and 6.6, with additional columns to allow a comparison to the other implementations. In Table 6.5 the computational results are presented for each of the instances tested. The solution quality results are in Table 6.6. Note that 124 flights with 8 desks is the only instance that exceeds the two-hour time limit. This means it is the only instance where column generation does not converge. The RMP UB after two hours is 981.766, so the gap between upper and lower bounds is 2.68%.

This shows that with smaller instances, optimal solutions are quickly found with the FDSF. The integer solutions found using the base implementation and IH take longer and are not optimal. However, with larger instances, the FDSF does not reach optimality in the time limit and has larger optimality gaps. For these instances, many converge with column generation within the time limit. The optimality gaps are significantly better using the base implementation for larger instances. The IH takes less than two seconds in each case, and sometimes improves on the base solution. This justifies the implementation of column generation on the set covering formulation in order to achieve better results on the larger instances.

$\|I\|$	m	NbCol	Iter	RMP-T	PP-T	Pop-T	Other-T	Total-T	IH-T	FDSF-T
46	2	2377	103	39.68	5.28	9.09	45.95	70.51	0.079	0.60
	3	884	36	15.77	9.49	26.68	48.06	26.70	0.163	1.80
	4	715	35	8.43	8.75	59.02	23.80	46.05	0.242	3.00
50	2	3270	152	47.43	4.88	6.60	41.09	120.07	0.085	0.60
	3	967	40	17.36	9.04	20.47	53.14	28.51	0.22	68.40
	4	566	22	12.35	10.46	23.38	53.81	17.74	0.157	7568.40
124	5	2786	115	1.03	93.92	2.37	2.68	3330.00	0.361	7203.00
	6	3091	129	1.53	90.74	4.14	3.59	2790.00	0.462	7268.40
	8	3618	144	0.64	96.30	1.77	1.29	8660.00	0.845	7287.60
163	7	2279	87	0.76	93.53	2.48	3.22	2962.70	1.157	7200.00
	8	3134	116	3.35	75.04	9.39	12.21	1200.00	1.236	7237.20
	11	2713	105	1.56	73.17	20.21	5.06	2120.00	1.013	7200.00

Table 6.5: Base implementation computational performance results.

$\|I\|$	m	LRB	IP	Gap	D-Min	D-Max	D-Avg	IH	I-Gap	FDSF	F-Gap
46	2	124	128	3.23	57	71	64.0	128	3.23	124	0
	3	124	132	6.45	39	51	44.0	128	3.23	124	0
	4	124	132	6.45	27	42	33.0	131	5.65	124	0
50	2	291	293	0.69	142	151	146.5	293	0.69	291	0
	3	291	296	1.72	64	132	98.7	291	0.00	291	0
	4	291	294	1.03	67	76	73.5	294	1.03	291	55
124	5	981	984	0.31	186	203	196.8	982	0.10	981	287
	6	981	983	0.20	154	178	163.8	983	0.20	981	528
	8	(956.169)*	986	3.12	114	135	123.3	985	3.02	981	474
163	7	340	344	1.18	45	55	49.1	342	0.59	340	295
	8	340	347	2.06	42	45	43.4	342	0.59	340	441
	11	340	344	1.18	26	37	31.3	344	1.18	340	314

*Did not converge within two hours.

Table 6.6: Base implementation solution quality results.

Although the instance using 50 flights and three desks solves to optimality, with four desks it does not solve in the time limit. This is the first instance in which the base implementation provides a smaller optimality gap. We also find that the integer solutions don't follow a clear trend and the number of desks increase. For example, with 50 flights the highest IP comes with three desks whereas the highest IH solutions comes with four desks. Similarly, the optimality gaps from the FDSF also don't directly correspond to the number of desks.

By comparing the bounds from column generation with the CPLEX results on the FDSF, we find that the LRB is tight. In Table 6.6, the LRB is equal to the FDSF solution (which represents an upper bound) in all instances in which column generation converged. In the remaining case, the difference between the FDSF solution and the upper bound from column generation is 0.776. The optimality gaps from the base implementation are mainly due to the difference between the IP solution and the RMP solution. This is an area for potential improvement.

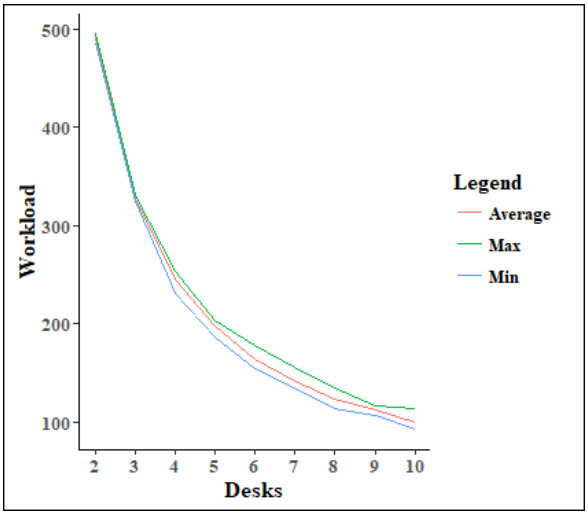
In the instances that were not solved by CPLEX within two hours, the incumbent solutions were all found within the first minute of solving. The remaining time is spent reducing the gap by increasing the lower bound. In most cases, the initial upper bound from column generation is close to the final bound but does not reach the final value until the LB converges as well. Overall, there is much more change in the LB than the UB. When analyzing the solutions to the RMP, where fractional schedules can be used, we find that many schedules are selected with low y_h values. Most selected schedules have $y_h \leq 0.10$ with many at $y_h \leq 0.02$. This means that we cannot estimate integer solutions based on the RMP solutions.

6.2.3 Effects of Changing the Number of Desks

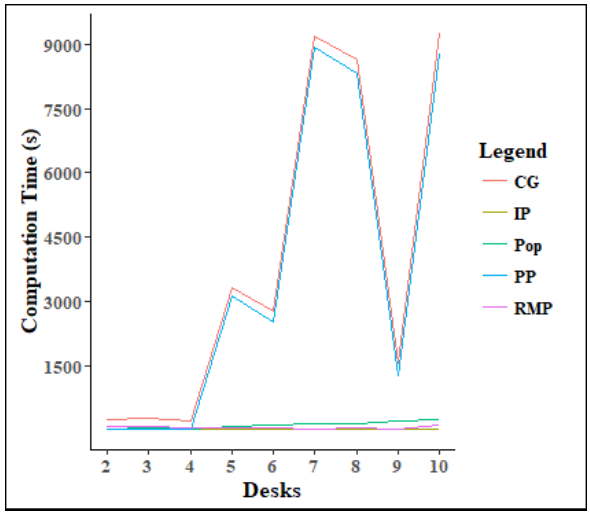
We perform additional tests to explore the effect of the number of desks and other parameters on the solution and computation. We test for $m = 2$ to 10. While this may provide unrealistic results in terms of number of desks and therefore flights per desk and overall workload, it does allow us to see trends in the problem. The data for these figures can be found in Appendix A.

In Figure 6.2a, we see the minimum and maximum desk workloads as well as the average workload over the desks for the EW instance. As the number of desks increases, the average workload decreases and the minimum and maximum workloads are non-increasing as well. This makes sense as the work is spread to more workers. As we increase the number of desks from two to five, there is a dramatic decrease in maximum desk workload. From five to nine, there is a decrease but at a slower pace. The change from nine to ten is very small. Similar patterns are present for average and minimum workload per desk. This can demonstrate the marginal benefit on workload of adding one more to desk. Figure 6.2b shows the computation time of the column generation procedure, which includes the time spent in the RMP, PP, and populate, and shows the time to solve the IP. Generally, as the number of desks increases, the computation time increases but this is not always the case (for example, nine desks solves within the time limit whereas the problems surrounding it do not).

We find that the PP takes up most of the CG time and the total time. Due the objective including maximum desk workload, it is not easy to find the optimal schedule. It is beneficial for a schedule to include any flights that do not increase the maximum workload as then they do not need to be covered by another schedule and will therefore not contribute to another schedule's maximum workload (unless needed to meet the minimum



(a) Desk workload by number of desks.



(b) Computation time by number of desks.

Figure 6.2: Effect of number of desks on desk workload and computation.

requirement of one flight per schedule). The maximum workload structure also makes it difficult to dominate solutions because not all time windows contribute to the value of a desk. This shows that the PP is where we could do further work to try to decrease the computation time. The data for these figures can be found in Tables A.1 and A.2.

We look at the total workload which is the model's objective. This can be measured using the Lagrangian lower bound, the integer solution found from column generation (which is an upper bound), or the IH. The bounds for the EAD instance are shown in Figure 6.3a. The LRB is strictly non-decreasing but in this case, is constant from 2 to 7 desks. As the number of desks increases, the sum of maximum workloads increases or remains the same. This is due to the structure of only considering the maximum workload of each dispatcher. With fewer desks, each dispatcher has more work but there is more overlap in their schedules and fewer flights are contributing to the maximum workload objective. The integer solutions have a generally increasing trend, but this is not strict. As

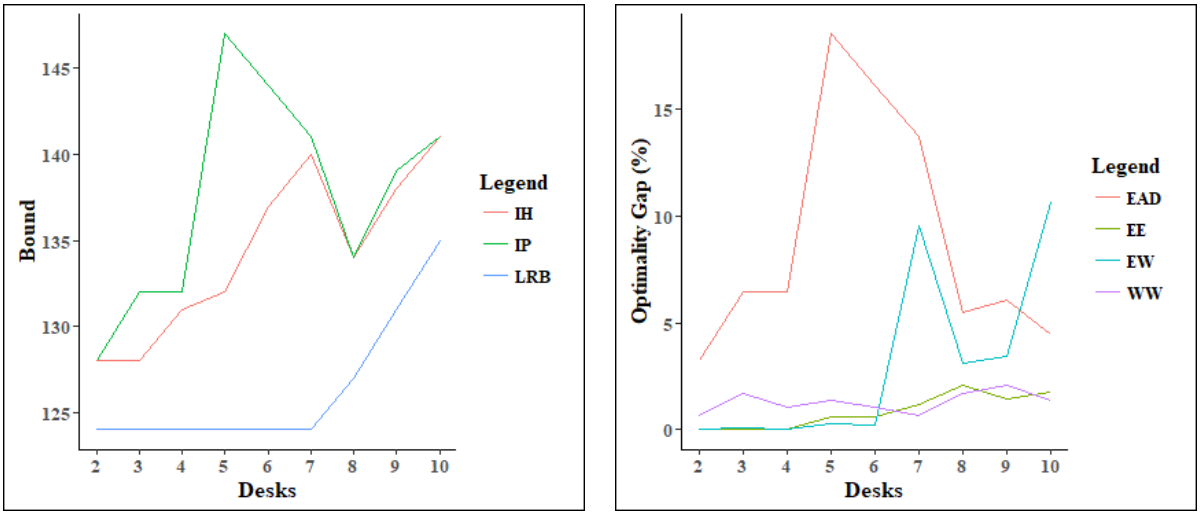
this is just an initial upper bound this does not represent the true optimal solutions. The optimal solutions should follow the strictly non-decreasing pattern as desks are increased. With the other instances, the LRB does not change (except when problems do not converge within two-hours) between two and ten desks. The integer solution values do change.

We explore in Figure 6.3 the differences in optimality gaps by number of desks for each of the instances. On the base implementation, the problems with the highest gaps are mostly from the EAD instance which solves to optimality quickly using the FDSF. The problems on this figure that did not converge are EW with seven, eight, and ten desks, which represent two of the other peaks in optimality gap. Without EAD and the instances that did not converge, the integer solutions all have a gap under 5%.

There is a general trend that increasing the number of desks slightly increases the optimality gap, however, this trend may start to reverse after a threshold number of desks, similar to EAD. The number of flights in an instance does not have a clear effect, as the WW and EE instances have a more similar pattern and they have 50 and 163 flights respectively. This may mean that other characteristics, such as load or flight distribution, affect the solution quality.

6.2.4 Flight Overlap Constraints

In Section 4.4, additional constraints are presented that limit which flights can be assigned to the same desk. Two flight overlap constraints are tested: flights departing at the same time ($\tau = 0$), and flights departing within five minutes of each other ($\tau = 5$). The goal of these constraints is to balance workload by not assigning flights departing around the same time to the same desk. These additional constraints may make some instances infeasible because there is an insufficient number of desks. We pre-process the minimum number



(a) Bounds from CG and IH.

(b) Optimality gaps by dataset.

Figure 6.3: Effect of number of desks on solution quality.

of desks required for feasibility and only report on those. Tables 6.7 and 6.8 display the results for the base implementation with these added constraints.

With $\tau = 0$, the only instances that cannot be solved are with 46 flights and 2 desks (denoted by 46-2) and 124-5. As we increase τ, fewer solutions can be found as more flights contribute to the maximum departure interval. For $\tau = 5$, in additional to the previous instances, we cannot solve for 46-3, 50-2, and 124-6.

In most cases, adding this constraint decreases the computation time. It is also true that as we increase τ from 0 to 5, the more constrained version ($\tau = 5$) generally takes less time to solve. The instances that took less time with flight overlap constraints than the base implementation are in the smaller instances, where the total time is less than one minute. The only instance that did not converge within two hours is 163-11 with $\tau = 0$, though this instance converges without the constraint and when using $\tau = 5$. We need to consider solution quality in additional to computational performance. The LRB is

$\|I\|$	m	NbCol	Iter	RMP-T	PP-T	Pop-T	Other-T	Total-T
				$\tau = 0$				
46	3	1202	50	20.58	6.90	25.53	47.00	34.41
	4	848	36	10.84	8.89	48.88	31.40	36.63
50	2	2328	104	35.71	6.59	9.30	48.40	70.44
	3	1062	46	18.99	8.86	18.40	53.76	31.07
	4	820	32	14.51	10.26	24.07	51.16	25.60
124	6	2373	96	4.46	67.14	13.98	14.42	533.53
	8	3111	128	0.61	95.37	2.63	1.39	6900.00
163	7	2145	81	8.28	32.10	21.22	38.40	229.88
	8	2127	80	5.33	49.46	20.58	24.63	353.13
	11	3432	131	0.67	95.25	2.36	1.72	7770.00
				$\tau = 5$				
46	4	740	33	12.31	8.19	43.88	35.62	29.67
50	3	1285	51	20.51	7.63	19.81	52.05	38.18
	4	648	26	12.16	10.07	30.80	46.98	22.69
124	8	2365	99	2.39	76.96	13.92	6.73	1150.00
163	7	2109	82	10.15	20.47	22.00	47.38	193.72
	8	2024	78	5.25	48.26	21.93	24.56	351.01
	11	2243	88	2.25	75.25	13.57	8.94	1080.00

Table 6.7: Flight overlap constraints computational performance results.

$\|I\|$	m	LRB	IP	Gap	D-Min	D-Max	D-Avg
				$\tau = 0$			
46	3	124	132	6.45	39	51	44.0
	4	124	132	6.45	27	42	33.0
50	2	291	291	0.00	133	158	145.5
	3	291	292	0.34	87	105	97.3
	4	291	292	0.34	58	82	73.0
124	6	981	1001	2.04	157	178	166.8
	8	981	986	0.51	114	135	123.3
163	7	340	346	1.76	44	54	49.4
	8	340	348	2.35	39	50	43.5
	11	(263.096)*	348	32.27	26	37	31.6
				$\tau = 5$			
46	4	124	138	11.29	27	48	34.5
50	3	291	292	0.34	84	107	97.3
	4	291	294	1.03	58	82	73.5
124	8	981	1000	1.94	99	136	125.0
163	7	340	346	1.76	44	54	49.4
	8	340	347	2.06	39	50	43.4
	11	340	354	4.12	27	37	32.2

*Did not converge within two hours.

Table 6.8: Flight overlap constraints solution quality results.

the same for all instances that converge, so the key difference comes from the IP solution and gap. We find that the solutions for the WW instance are better or equal to the base implementation, but in other cases the results are worse or equal to the base solutions. The gaps for 124-8 with the flight overlap constraints are improved from the base model but this is due to improvement in the LRB, since it did not converge with the base implementation. The integer solution from the base implementation is equal to the solution with $\tau = 0$, which is better than the solution found with $\tau = 5$.

6.2.5 Comparing all Models

Using the data from previous tables, we can compare the performance of the base implementation, IH, FDSF solved with CPLEX, and base implementation with flight overlap constraints using $\tau = 0$, and $\tau = 5$. In this section, several instances are highlighted.

In each case, a single instance is displayed with a set number of desks. However, because not all instances provide a solution to the model with flight overlap constraints, the smallest feasible number of desks is included in those cases. In Table 6.9, we compare the results for EW instance with six desks. In this case, the base implementation (and IH) provide the best gap. The models with flights overlap constraints solve in less time, but the final integer solutions are larger. These could provide a trade-off between time and integer solution values. Finally, the FDSF finds the best objective value (that we know is optimal from the LB of the base model) but does not prove optimality within the two-hour time limit and ends with a gap of 528%.

For the EAD instance, the smallest tested, all problems solve in under a minute. Results are presented in Table 6.10. The FDSF finds an optimal solution, whereas each of the integer solutions from column generation have a gap. The improvement heuristic improves

Version	SCF	IH	FDSF	$\tau = 0$	$\tau = 5$
$\|I\|$	124	124	124	124	124
m	6	6	6	6	8
LB	981	981	156.23	981	981
Objective	983	983	981	1001	1000
Gap	0.20	0.20	527.92	2.04	1.94
Time (s)	2582.46	0.46	7268.66	533.53	1150.00

Table 6.9: Comparison of models for EW instance with six desks.

Version	SCF	IH	FDSF	$\tau = 0$	$\tau = 5$
$\|I\|$	46	46	46	46	46
m	3	3	3	3	4
LB	124	124	124	124	124
Objective	132	128	124	132	138
Gap	6.45	3.23	0.00	6.45	11.29
Time (s)	26.70	0.16	1.56	34.41	29.67

Table 6.10: Comparison of models for EAD instance with three desks.

the solution but still does not reach optimality.

6.2.6 Limitations

We tested on the CC instance, as this is the next smallest in size. We start with the lowest target number of desks, which is 12. A comparison of the models is presented in Table 6.11. We find that this instance is not solved within two hours for any of these implementations (except the IH, which needs a base solution as input). The optimality gaps remain very large after two hours. For each of the column generation implementations, over 95% of the time is spent in the PP. Similar results are found when testing with 20 desks. Further testing is not performed on this instance or larger instances. Computational improvements could lead to larger instances, such as this one, converging within two hours. This is where

Version	SCF	IH	$\tau = 0$	$\tau = 5$	FDSF
$\|I\|$	297	297	297	297	297
m	12	12	12	12	12
LB	26.06	26.06	26.34	30.85	41.47
Objective	275	267	280	276	241
Gap	955.27	924.57	962.84	794.58	481.11
Time (s)	22700.00	5.67	7600.00	7510.00	7206.08

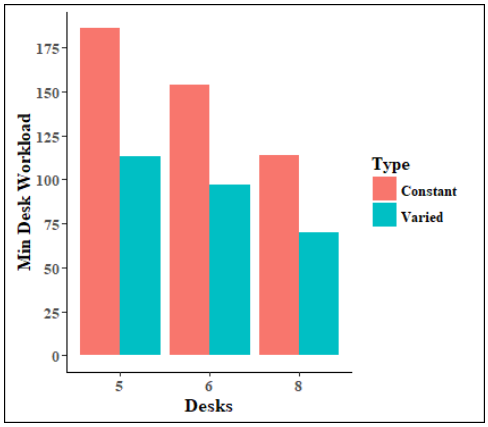
Table 6.11: Comparison of models for CC instance with 12 desks.

future efforts could be focused to improve the implementation for larger instances.

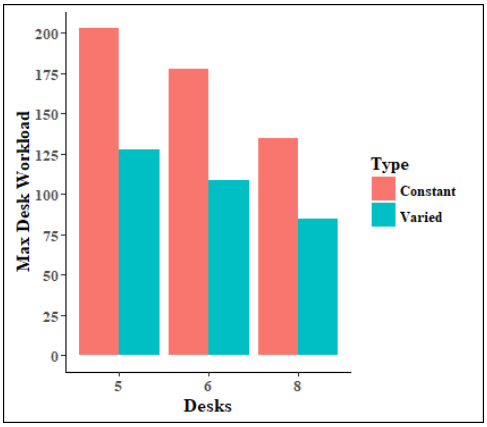
6.2.7 Varying Load Over Time

So far all results are with constant load throughout the duration of the flight. We also consider when the load decreases during the post-planning period. As discussed in Section 6.1.3, the loads during the post-planning period are decreased by 50% and occur from one hour after the flight's departure time until its arrival time.

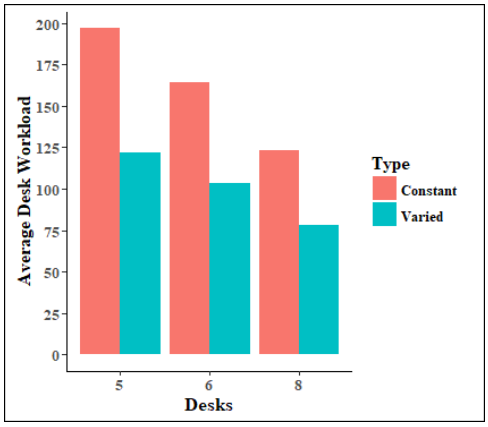
Results are presented for the EW instance with post-planning load, and we compare to the results with constant load. Figure 6.4 demonstrates the differences in solution quality and computational performance when the load is varied throughout the flight. The data for the varied load can be found in Tables A.11 and A.12.



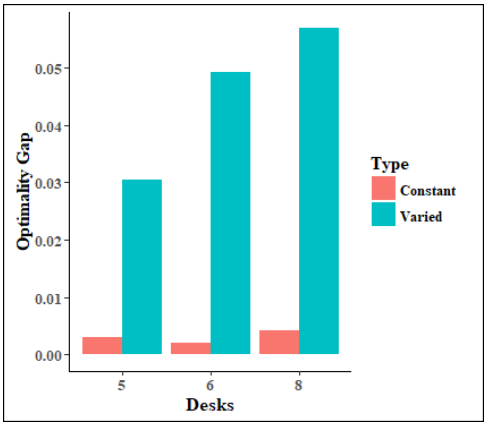
(a) Minimum desk workload.



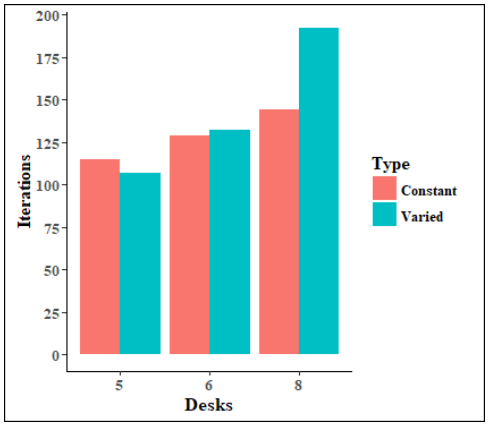
(b) Maximum desk workload.



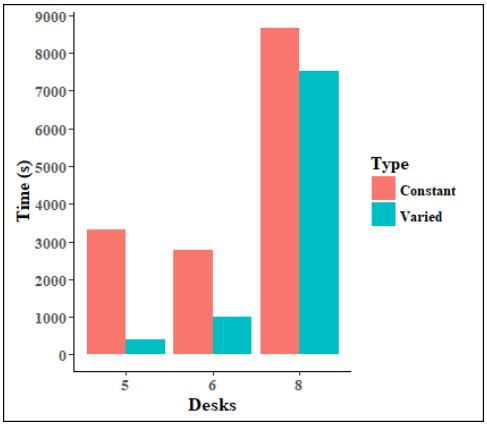
(c) Average desk workload.



(d) Optimality gap.



(e) Iterations.



(f) Computation time.

Figure 6.4: Comparison of varied and constant load results.

As expected, the minimum, maximum and average desk workloads all decrease when varied load is used (where the variations are all decreases from the base). In these cases, the optimality gap is larger with varied load but the computation time is decreased. There is no clear pattern with the number of iterations. This analysis is not to make conclusions about the differences in the results, but rather to demonstrate the functionality of varying load.

6.3 Conclusion

Two different datasets are used for testing, but these can be decomposed into several instances using dispatch zones. Tests are focused on instances containing 46, 50, 124, and 163 flights in a single day. We tested the base column generation implementation, with a single column and multiple columns generated in each iteration. In all cases, the solution quality is equivalent, but there are differences in computation time. While for some cases, a single column is faster, we find that for the EE instance there are more gains from populating multiple columns. We tune the populate parameters based on the performance of the EW instance. These settings are used for the other tests.

The base column generation implementation, and the improvement heuristic performed on its solution, are compared to the Flight Dispatcher Schedule Formulation (FDSF). We find that for small instances, the solution of FDSF by CPLEX reaches optimality quickly but this is not the case for larger instances. For the EE and EW instances, and some cases with the smaller instances, the FDSF does not converge within two hours and the bounds provided by the base implementation are tighter. The heuristic finds improvements to the IP solution in some cases but not all, and always runs quickly. The base implementation is compared to versions adding the flight overlap constraint with $\tau = 0$ and $\tau = 5$. While

no consistent pattern exists, generally this constraint decreases the solution time but the optimality gaps worsen.

The effects of modifying the number of desks and the load were explored. As expected, increasing the number of desks decreases the workload per desk but the overall objective follows a non-decreasing trend. There is a generally increasing trend in computation time as desks increase. Using load that varies during the flight decreases the desk workload and the computation time, but increases the optimality gaps on the instances tested.

None of these implementations provide good results when using the next smallest instance, which contains 297 flights. Most of the computation time on the larger instances is focused on the PP, so this is where future improvements could be made. Other areas for future work and extensions are presented in the next chapter.

Chapter 7

Extensions and Future Work

There are various areas for potential extensions or future work that focus on capturing different elements of the problem. In Section 7.1 we discuss alternative ways for dispatch zones to be considered in the modelling. Additional or alternative constraints and objectives are presented in Sections 7.2 and 7.3 respectively. Including consideration for uncertainty are discussed in Section 7.4. Lastly, Section 7.5 discusses adding the shift problem to capture both parts of flight dispatcher scheduling.

7.1 Incorporating Dispatch Zones

In our current testing, we break down the AA data into dispatch zones and then solve each decomposed instance with a specified number of dispatchers for each zone. This breaks down the dataset in a simple way, but there are other ways in which dispatch zones could be considered. For example, we may specify a total number of dispatchers, but not how many will be assigned to each zone. This means that all of the instances we decomposed

are linked with a single m. In this case, all flights would be solved in one problem but the schedules generated would each only contain one zone. This would introduce a new parameter and decision variable, and add constraints 7.1 - 7.3 to the PP.

let L represent the set of zones, $l \in L$

$$\text{let } \sigma_{il} = \begin{cases} 1 & \text{if flight } i \in I \text{ is in zone } l \in L \\ 0 & \text{otherwise} \end{cases}$$

$$\text{let } z_l = \begin{cases} 1 & \text{if zone } l \in L \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

$$x_i \leq \sigma_{il} z_l \qquad \forall i \in I, l \in L \tag{7.1}$$

$$\sum_{l \in L} z_{l \in L} = 1 \tag{7.2}$$

$$z_l \in \{0, 1\} \qquad \forall l \in L \tag{7.3}$$

Another way to consider zones would be to have an additional cost for a desk containing flights from multiple zones. This would essentially act as a penalty, as there could be added work when flight zones are more diverse. Constraint 7.2 would no longer be included and the PP objective would be updated to 7.4, where there is a cost function based on how many zones are included.

$$\min F - \sum_{i \in I} \bar{\lambda}_i x_i + f(\sum_{l \in L} z_l) \tag{7.4}$$

Similarly, an alternative form of zonal considerations would be to measure load by groups of flights. For example, if two flights with very similar flight paths and timing are assigned to the same desk they would likely require less total load, as some of the work overlaps, than if they are assigned to two different desks. This sort of measurement of load would form a non-linear load function as we would need to consider flight load relative to other flights on a schedule.

7.2 Alternative Constraints

There are alternative constraints that can be added to the definition of a schedule, and therefore the PP. The total number of flights assigned to a desk can be constrained by 7.5 and 7.6. By adding an additional parameter and decision variable, we constrain the number of active flights per desk. Constraint 7.8 constrains the active flights, with 7.7 defining the active flights, similar to how the load is calculated.

$$d_{it} = \begin{cases} 1 & \text{if flight } i \in I \text{ is active at time } t \\ 0 & \text{otherwise} \end{cases}$$

w_i The number of active flights at S_i if flight $i \in I$ is added to the schedule

$$\sum_{i \in I} x_i \geq minNumberOfFlights \qquad \forall i \in I \qquad (7.5)$$

$$\sum_{i \in I} x_i \leq maxNumberOfFlights \qquad \forall i \in I \qquad (7.6)$$

$$w_i \geq 1 + \sum_{j<i} x_j d_{jS_i} - M(1 - x_i) \qquad \forall i \in I \qquad (7.7)$$

$$w_i \leq maxActiveFlights \qquad \forall i \in I \qquad (7.8)$$

In addition to the number of flights, we can consider the idle time when the desk contains no active flights. This is something that we want to avoid, especially for long periods of time, as the dispatcher is getting paid but is not doing any work and there may be a higher imbalance of load during that time. To do this, additional parameters are needed to keep track of the time:

e_i The idle time leading into the start of flight $i \in I$, if flight $i \in I$ is added to the desk

g_i The arrival time of the latest flight that is active at the start of flight $i \in I$

a_i The arrival time of flight $i \in I$

Idle time is limited by adding constraints 7.10, 7.13 and 7.14 to the PP. We either constrain the maximum idle time between any two active flights for a dispatcher (i.e., a single continuous break) by 7.15, or the cumulative idle time throughout a desk by 7.16. The remaining equations show the derivations which are linearized for the PP.

$$
\begin{aligned}
g_i &= \max_{j<i}(a_j x_j) && \forall i \in I && (7.9)\\
&\geq a_j x_j && \forall i \in I, j \in I, j < i && (7.10)\\
e_i &= max\{S_i - \max_{j<i}(a_j x_j), 0\} x_i && \forall i \in I && (7.11)\\
&= max(S_i x_i - g_i, 0) && \forall i \in I && (7.12)\\
e_i &\geq S_i x_i - g_i && \forall i \in I && (7.13)\\
e_i &\geq 0 && \forall i \in I && (7.14)\\
e_i &\leq maxIdleTime && \forall i \in I && (7.15)\\
\sum_{i \in I} e_i &\leq maxTotalIdleTime && && (7.16)
\end{aligned}
$$

7.3 Alternative Objectives

As discussed with different problems considering balance, there are various ways to define and measure balance. These could be implemented by adjusting the Flight Dispatcher Schedule Formulation or IP objectives, but this may affect the dual problem and the solution methodology which is based on the current formulation.

To consider the difference in workload of the worst desk and the best desk, we modify the RMP by adding two new parameters; C_{min}, the desk chosen with the minimum workload, and C_{max}, the desk chosen with the maximum workload. The the objective is changed to minimizing $C_{max} - C_{min}$ where $C_{min} \leq C_h y_h$ and $C_{max} \geq C_h y_h$, $\forall h \in H$.

To compare all chosen schedule workloads to the average workload, the sum of squared deviations (SSD) is used. To find the SSD, we need to have the mean workload of the

selected desks which will be called C_{mean}. Then we need to find the difference between a schedule's workload with the mean, for each desk that is selected.

$$SSD = \sum_{h \in H} [(C_h - C_{mean}) y_h]^2$$

This could be used as is, or normalized. This makes the problem non-linear as we have C_{mean} multiplied by y_h but C_{mean} is already dependent on y_h as it is calculated from the selected schedules.

The alternative constraints proposed in section 7.2 can be converted into objectives. For example, instead of constraining the maximum number of flights, the objective could be to minimize the number of flights assigned to the busiest desk. Similarly, the other types of balance discussed in this section could be applied to the alternative objectives.

7.4 Uncertainty

There are a few areas of potential uncertainty that could affect the workload of these schedules. A common area of uncertainty in airline operations is flight delays. When a flight is delayed, the overall load and when it occurs can change. Since dispatchers monitor weather and safety conditions, they may be doing work before a flight has been delayed and during the delay period. They would then still need to monitor the flights after departure, and the timing of this load would not be as planned.

There is potential uncertainty in the amount of load each flight actually incurs. As mentioned above, one reason may be because of delays, but the conditions of the aircraft or weather may alter the load of each flight as they are occurring. We do not consider any overlap in the load that may occur when two flights with similar regions/paths occur at

the same time. Some of the dispatcher's work may be applied to both flights and therefore the actual workload is decreased.

The last potential area of uncertainty may be in the number of dispatchers who are actively working at a given time. This could be based on people arriving late or early, which may just affect shift changeover times, but contingencies could be useful for when a dispatcher cannot cover their shift. Even during a quick break, a dispatcher needs to ask another dispatcher to monitor their flights (Kennedy, 1987), so it is a common occurrence for dispatchers to be covering extra flights.

7.5 Shift Considerations

In defining the problem, there were two key components: assigning flights to desks, and scheduling dispatcher shifts to cover desks. This work focuses on the former, so shift rules and guidelines are not considered in this modelling. Including shift considerations would require a new model either to formulate the whole problem or to connect with this work.

This would entail defining when a desk starts and ends (i.e., when they start work for the first flight for the desk, and when the last flight arrives and a desk can be closed), and potentially how that would be broken up if multiple dispatcher shifts are required. There are specific time constraints with set durations that a dispatcher can work. This may require looking at a longer period of time since a desk can remain open for more than 24 hours, and the timing of shifts may not perfectly align with the timing of one "day" of flights.

7.6 Conclusion

In this chapter, extensions and future work are proposed that would model different elements from the dispatching problem. Different ways to incorporate regions into the flight dispatching problem are discussed. So far, regions are used to decompose problems, but we could use the pricing problem to limit flights to a single region, add penalties for assigning a dispatcher flights in multiple regions, or measure workload with flights dependent on other flights. Additional constraints are presented that would limit the number of flights or active flights allowed for each dispatcher, or constrain idle time. These are also different possible objectives, and alternative measures of balance are discussed, such as minimizing the maximum difference in workload or (normalized) sum of squared deviations. Areas of uncertainty that could be incorporated in future work include flight delays, uncertain loads, and number of dispatchers present. Lastly, the inclusion of shifts would address the complete problem of flight dispatcher scheduling. Conclusions are presented in the next chapter.

Chapter 8

Conclusions

Flight dispatching is an area of opportunity in the literature of airline operations planning. The problem of dispatching includes assigning flights amongst a set of desks, and splitting up those desks into dispatcher shifts according to agreed upon rules. We propose models to address the problem of assigning flights to desks with an objective of balancing workload. We measure balance by determining each desk's highest workload throughout the day and minimizing the sum of the selected desk's maximum workloads.

We develop the Flight Dispatcher Schedule Formulation (FDSF) which is solved directly by CPLEX. While this solves quickly for small instances, its effectiveness is limited with larger instances. A set covering formulation is presented that is solved using a column generation algorithm to address larger-scale instances. This solves over a subset of of schedules, where each schedule is the assignment of flights to a single desk. To initialize the set of schedules, a greedy heuristic and swapping algorithm is used that guarantees a feasible solution to the Integer Problem (IP) and Relaxed Master Problem (RMP). A variation to this base column generation implementation adds flight overlap constraints to

the Pricing Problem (PP), in which we limit flights that depart within some time interval, τ, cannot be assigned to the same desk. We present an improvement heuristic (IH) to try to improve upon the integer solutions found through the column generation implementation.

We test on two datasets, one from European Airline Data (EAD), and one from American Airlines (AA) that is broken down into several instances. We test instances with 46, 50, 124, and 163 flights. We vary the number of desks for each instance to target average flight per desk values of 15, 20 and 25. The smaller instances are solved directly by CPLEX on the FDSF in seconds. The larger instances, do not solve within a two-hour time limit and optimality gaps remain large at the end of this time. Most instances converge in column generation within two hours, though optimality gaps are present from the integer solutions. By looking at the RMP solutions, we find that many desks are selected with values of $y_h \leq 0.10$. This means we cannot estimate integer solutions from the RMP solutions. The IH provides smaller gaps in some cases. We find that the Lagrangian lower bound (LRB) is tight when it converges, as it is equal to the FDSF solution which is an upper bound.

Using an instance from AA containing 297, we find that none of the implementations converge within the time limit, and the optimality gaps remain very large after this time. With instances that take longer to solve, we find that most of the time is spent in the PP. Improving the computation time of the PP would allow us to solve larger instances. Future work and possible extensions include considering dispatch zones in the PP rather than as separate instances, adding constraints to define schedules that limit the number of flights or idle time (at a time or cumulatively), considering different balance objectives, incorporating uncertainty, and including the shift component of dispatcher scheduling.

References

Acar, I. and Butt, S. E. Modeling nurse-patient assignments considering patient acuity and travel distance metrics. *Journal of Biomedical Informatics*, 64:192–206, 12 2016.

Adler, M., Berenbrink, P., and Schröder, K. Analyzing an infinite parallel job allocation process. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1461 LNCS, pages 417–428. Springer Verlag, 1998.

Aickelin, U. and Dowsland, K. A. An indirect genetic algorithm for a nurse-scheduling problem. *Computers and Operations Research*, 31(5):761–778, 4 2004.

Aickelin, U. and White, P. Building better nurse scheduling algorithms. *Annals of Operations Research*, 128(1-4):159–177, 2004.

Air Canada Corporate Communications. Air Canada Corporate Profile, 2020. URL https://www.aircanada.com/ca/en/aco/home/about/corporate-profile.html.

Air Transport Association of Canada. Career Pathways: Flight Dispatcher. *Fly Canada*, 2021. URL https://flycanada.org/career-pathways/flight-dispatcher/.

Airline Dispatchers Federation. About ADF — Airline Dispatchers Federation, 2021. URL https://www.dispatcher.org/dispatcher/about-adf.

American Airlines. American Airlines Group. URL https://www.aa.com/i18n/customer-service/about-us/american-airlines-group.jsp#:~:text=Overview,1%2C000%20destinations%20in%20150%20countries.

American Airlines Newsroom. Dispatching live from the airline's nerve center, 2020. URL http://news.aa.com/american-stories/american-stories-details/2017/Straight-to-the-Gate-Andy-Egloff/default.aspx.

Azai, Y. and Epstein, L. On-line load balancing of temporary tasks on identical machines. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pages 119–125, 1997.

Azar, Y. On-line load balancing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1442, pages 178–195. Springer Verlag, 1998.

Baldi, M. M., Crainic, T. G., Perboli, G., and Tadei, R. The generalized bin packing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(6): 1205–1220, 11 2012.

Barnhart, C., Belobaba, P., and Odoni, A. R. Applications of operations research in the air transport industry. *Transportation Science*, 37(4):368–391, 11 2003.

Barnhart, C., Cohn, A. M., Johnson, E. L., Klabjan, D., Nemhauser, G. L., and Vance, P. H. Airline Crew Scheduling. In *Handbook of Transportation Science*, pages 517–560. Kluwer Academic Publishers, 2 2006.

Berenbrink, P., Sauerwald, T., Stauffer, A., and Khodamoradi, K. Balls-into-bins with nearly optimal load distribution. In *Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 326–335, New York, New York, USA, 2013. ACM Press.

Castro-Silva, D. and Gourdin, E. A study on load-balanced variants of the bin packing problem. *Discrete Applied Mathematics*, 264:4–14, 2019.

Chen, B., Potts, C. N., and Woeginger, G. J. A Review of Machine Scheduling: Complexity, Algorithms and Approximability. In *Handbook of Combinatorial Optimization*, volume 3, pages 1493–1641. Springer US, 1998.

Correia, I., Gouveia, L., and Saldanha-da Gama, F. Solving the variable size bin packing problem with discretized formulations. *Computers and Operations Research*, 35(6):2103–2113, 6 2008.

Crainic, T. G., Fomeni, F. D., and Rei, W. The Multi-Period Variable Cost and Size Bin Packing Problem with Assignment Cost: Efficient Constructive Heuristics. CIRRELT, 7 2019.

Curran, A. Which Canadian Airlines Have The Largest Fleets? *Simple Flying*, 2020. URL https://simpleflying.com/canadian-airlines-largest-fleets/#:~:text=In%202018%2C%20Air%20Canada%20had,with%20a%2034%25%20market%20share.

Delorme, M., Iori, M., and Martello, S. *Bin packing and cutting stock problems: Mathematical models and exact algorithms*, volume 255, pages 1–20. Elsevier, 11 2016.

Dickson, G. Up all night at American Airlines – What happens after hours at the Fort Worth HQ? *Fort Worth Star-Telegram*, 2019. URL https://www.star-telegram.com/news/business/aviation/article226769464.html.

Ehrgott, M. and Ryan, D. M. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11(3):139–150, 2002.

Elhedhli, S., Gzara, F., and Yildiz, B. Three-Dimensional Bin Packing and Mixed-Case Palletization. *INFORMS Journal on Optimization*, 1(4):323–352, 2019.

Federal Aviation Administration. FAA Runway Safety Group - Service Areas & Regional Offices, 2021. URL https://www.faa.gov/airports/runway_safety/regions/.

Hnich, B., Kzfitan, Z., and Walsh, T. Modelling a balanced academic curriculum problem. 08 2002.

Kennedy, J. M. Monitors Every Trip : Air Dispatcher Is Pilot's Partner in Flight Safety. *Los Angeles Times*, 1987. URL https://www.latimes.com/archives/la-xpm-1987-09-03-mn-5786-story.html.

Khouja, M. and Conrad, R. Balancing the assignment of customer groups among employees: Zero-one goal programming and heuristic approaches. *International Journal of Operations and Production Management*, 15(3):76–85, 1995.

Kriengkorakot, N. and Pianthong, N. The assembly line balancing problem : Review articles *. *The KKU Engineering Journal*, Volume 34:133–140, 03 2007.

Lu, D. and Gzara, F. The robust crew pairing problem: model and solution methodology. *Journal of Global Optimization*, 62(1):29–54, 4 2015.

Martello, S., Pisinger, D., and Vigo, D. The Three-Dimensional Bin Packing Problem. *Operations Research*, 48(2):256–267, 2000.

Mazareanu, E. EBIT margin of commercial airlines worldwide from 2010 to 2021, by region. *Statistica*, 2021a. URL https://www.statista.com/statistics/225856/ebit-margin-of-commercial-airlines-worldwide/.

Mazareanu, E. Leading airlines in the U.S. by domestic market share 2020. *Statistica*, 2021b. URL https://www.statista.com/statistics/250577/domestic-market-share-of-leading-us-airlines/.

Mondal, R. Flight-Delay, 2019. URL https://www.kaggle.com/rahulstephenites2/flightdelay?select=flights.csv.

Otley, T. Interview: Scott Ramsey, MD of flight despatch and operations control, American Airlines. *Business Traveller*, 2018. URL https://www.businesstraveller.com/features/interview-scott-ramsey-md-of-flight-despatch-and-operations-control-american-airlines/.

Ouazene, Y., Hnaien, F., Yalaoui, F., and Amodeo, L. The Joint Load Balancing and Parallel Machine Scheduling Problem. In *Operations Research Proceedings 2010*, pages 497–502. Springer, Berlin, Heidelberg, 2011.

Raab, M. and Steger, A. "Balls into bins" — a simple and tight analysis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1518, pages 159–170. Springer Verlag, 1998.

Rajakumar, S., Arunachalam, A. V. P., and Selladurai, A. V. Workflow balancing strategies in parallel machine scheduling. 23:366–374, 2004.

Rosenberger, J., Green, D., Keeling, B., Turpin, P., and Zhang, J. Optimizing nurse assignment. 05 2014.

Rosenberger, J. M., Schaefer, A. J., Goldsman, D., Johnson, E. L., Kleywegt, A. J., and Nemhauser, G. L. A stochastic model of airline operations. *Transportation Science*, 36 (4):357–377, 2002.

Santos, C. A., Zhang, A., Lopez, I., Herrick, G., and Raper, M. Optimal dispatcher workload distribution. *HP Laboratories Technical Report*, (195):1–26, 2011.

Schaus, P. *Solving Balancing and Bin-Packing problems with Constraint Programming*. PhD thesis, Ecole polytechnique de Louvain, 2009.

Schwerdfeger, S. and Walter, R. Improved algorithms to minimize workload balancing criteria on identical parallel machines. *Computers and Operations Research*, 93:123–134, 5 2018.

Shebalov, S. and Klabjan, D. Robust airline crew pairing: Move-up crews. *Transportation Science*, 40(3):300–312, 2006.

Transport Workers Union of America. Agreement between American Airlines, Inc. and the flight dispatcher, dispatchers in training, and operations specialists in the service of American Airlines, Inc. as represented by Transport workers Union of America, AFL-CIO, 2016. URL https://twu514.org/files/2016/04/AA-Dispatchers-TA-Complete.pdf.

Valério De Carvalho, J. M. LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141(2):253–273, 9 2002.

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

APPENDICES

Appendix A

Additional Results

A.1 Results of Varying the Number of Desks

m	NbCol	Iter	RMP-T	PP-T	Pop-T	Other-T	Total-T
2	2377	103	39.68	5.28	9.09	45.95	70.51
3	884	36	15.77	9.49	26.68	48.06	26.70
4	715	35	8.43	8.75	59.02	23.80	46.05
5	800	35	8.56	8.95	56.31	26.18	45.85
6	747	39	7.96	7.27	63.49	21.28	52.23
7	521	48	5.04	9.34	74.78	10.84	88.73
8	622	66	6.80	8.30	73.19	11.71	106.34
9	388	57	3.67	7.85	81.53	6.95	126.10
10	501	84	3.76	7.80	82.39	6.05	204.14

Table A.1: Varying m on EAD instance computational performance results.

m	LRB	IP	Gap	D-Min	D-Max	D-Avg	IH	I-Gap
2	124	128	3.23	57	71	64.00	128	3.23
3	124	132	6.45	39	51	44.00	128	3.23
4	124	132	6.45	27	42	33.00	131	5.65
5	124	147	18.55	27	35	29.40	132	6.45
6	124	144	16.13	16	35	24.00	137	10.48
7	124	141	13.71	7	33	20.14	140	12.90
8	127	134	5.51	5	22	16.75	134	5.51
9	131	139	6.11	7	20	15.44	138	5.34
10	135	141	4.44	5	20	14.10	141	4.44

Table A.2: Varying m on EAD instance solution quality results.

m	NbCol	Iter	RMP-T	PP-T	Pop-T	Other-T	Total-T
2	3270	152	47.43	4.88	6.60	41.09	120.07
3	967	40	17.36	9.04	20.47	53.14	28.51
4	566	22	12.35	10.46	23.38	53.81	17.74
5	795	30	13.88	10.41	25.75	49.95	23.75
6	605	23	12.15	10.89	28.82	48.14	19.10
7	539	22	10.77	11.84	33.71	43.68	19.10
8	614	26	9.71	10.69	45.62	33.98	27.64
9	639	25	10.20	9.12	41.49	39.19	25.33
10	522	19	11.26	11.09	30.47	47.18	17.04

Table A.3: Varying m on WW instance computational performance results.

m	LRB	IP	Gap	D-Min	D-Max	D-Avg
2	291	293	0.69	142	151	146.50
3	291	296	1.72	64	132	98.67
4	291	294	1.03	67	76	73.50
5	291	295	1.37	52	73	59.00
6	291	294	1.03	36	56	49.00
7	291	293	0.69	28	51	41.86
8	291	296	1.72	26	53	37.00
9	291	297	2.06	25	50	33.00
10	291	295	1.37	3	56	29.50

Table A.4: Varying m on WW instance solution quality results.

m	NbCol	Iter	RMP-T	PP-T	Pop-T	Other-T	Total-T
2	3895	164	28.63	6.04	8.46	56.86	224.08
3	4006	172	26.50	11.11	12.93	49.47	257.49
4	3058	133	21.69	13.23	14.74	50.34	192.04
5	2786	115	1.03	93.92	2.37	2.68	3330.00
6	3091	129	1.53	90.74	4.14	3.59	2790.00
7	2397	100	0.26	97.44	1.51	0.79	9190.00
8	3618	144	0.64	96.30	1.77	1.29	8660.00
9	2114	91	1.33	80.30	13.72	4.64	1550.00
10	4745	191	1.04	94.76	2.71	1.49	9250.00

Table A.5: Varying m on EW instance computational performance results.

m	LRB	IP	Gap	D-Min	D-Max	D-Avg
2	981	981	0.00	486	495	490.50
3	981	982	0.10	325	331	327.33
4	981	981	0.00	231	254	245.25
5	981	984	0.31	186	203	196.80
6	981	983	0.20	154	178	163.83
7	(899.82)*	986	9.58	134	155	140.86
8	(956.17)*	986	3.12	114	135	123.25
9	981	1015	3.47	106	117	112.78
10	(896.58)*	992	10.64	92	114	99.20

*Did not converge within two hours.

Table A.6: Varying m on EW instance solution quality results.

m	NbCol	Iter	RMP-T	PP-T	Pop-T	Other-T	Total-T
2	3574	152	23.53	6.10	4.92	65.45	233.22
3	3304	146	23.35	5.90	5.76	64.98	210.78
4	3183	145	23.56	5.80	7.27	63.36	200.57
5	2455	101	10.46	34.22	16.36	38.95	239.04
6	2510	97	12.23	19.05	20.67	48.06	202.93
7	2279	87	0.76	93.53	2.48	3.22	2960.00
8	3134	116	3.35	75.04	9.39	12.21	1200.00
9	3092	118	1.13	91.56	3.92	3.39	3500.00
10	3446	133	2.86	76.80	12.92	7.42	1780.00

Table A.7: Varying m on EE instance computational performance results.

m	LRB	IP	Gap	D-Min	D-Max	D-Avg
2	340	340	0.00	169	171	170.00
3	340	340	0.00	109	116	113.33
4	340	340	0.00	79	89	85.00
5	340	342	0.59	65	74	68.40
6	340	342	0.59	54	61	57.00
7	340	344	1.18	45	55	49.14
8	340	347	2.06	42	45	43.38
9	340	345	1.47	33	42	38.33
10	340	346	1.76	31	39	34.60

Table A.8: Varying m on EE instance solution quality results.

A.2 CG Results Using a Single Column

m	NbCol	Iter	RMP-T	PP-T	Other-T	Total-T
2	3003	2995	15.46	63.67	20.87	5910.00
3	3007	2996	14.77	63.66	21.57	6220.00
4	680	664	11.00	60.36	28.64	365.51
5	332	313	4.42	54.55	41.03	266.90
6	311	287	1.57	76.58	21.85	776.95
7	316	289	1.22	81.33	17.45	1020.00
8	378	346	1.96	67.34	30.70	901.54
9	342	307	1.40	79.69	18.91	997.29
10	515	475	0.54	92.87	6.59	6010.00
11	478	435	0.74	90.17	9.08	3350.00

Table A.9: Column generation on EE instance, Computational Performance Results.

m	LRB	IP	Gap	D-Min	D-Max	D-Avg
2	340	340	0.00	169	171	170.00
3	340	340	0.00	109	116	113.33
4	340	340	0.00	79	89	85.00
5	340	342	0.59	65	74	68.40
6	340	342	0.59	54	61	57.00
7	340	344	1.18	45	55	49.14
8	340	347	2.06	42	45	43.38
9	340	345	1.47	33	42	38.33
10	340	346	1.76	31	39	34.60
11	340	344	1.18	26	37	31.27

Table A.10: Column generation on EE instance Solution Quality Results.

A.3 Results Using Varying Load

$\|I\|$	m	NbCol	Iter	RMP-T	PP-T	Pop-T	Other-T	Total-T
124	5	2669	107	7.70	51.55	20.05	20.69	403.52
124	6	3313	132	4.50	60.13	25.14	10.24	997.51
124	8	4517	192	1.20	89.88	7.15	1.76	7510.00

Table A.11: Varying the load on EW instance computational performance Results.

$\|I\|$	m	LRB	IP	Gap	D-Min	D-Max	D-Avg
124	5	590	608	0.03	113	128	121.60
124	6	590	619	0.05	97	109	103.17
124	8	(571.86)*	624	0.06	70	85	78.00

*Did not converge within two hours.

Table A.12: Varying the load on EW instance solution quality results.