MIT LINCOLN LABORATORY

Evaluating the Robustness and Feasibility of Integer Programming and Dynamic Programming in Aircraft Sequencing Optimization

A Major Qualifying Project Report:

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Julia Baum

William Hawkins

Date: October 13, 2011

This work is sponsored by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and not necessarily endorsed by the United States Government.

Professor Jon P. Abraham, WPI Project Advisor

Professor George T. Heineman, WPI Project Advisor

Richard Jordan, MIT Lincoln Laboratory Liaison

Mariya Ishutkina, MIT Lincoln Laboratory Liaison

Abstract

This project uses Mixed Integer Linear Programming and Dynamic Programming to optimize the takeoff sequence of aircraft at Dallas/Fort Worth (DFW) by minimizing departure delay while also obeying separation requirements and position shifting constraints. We modeled taxi time uncertainties based on real data from DFW and analyzed the robustness of the optimization solution and the feasibility of using these methods in real-life. The runtimes of these methods proved to be feasible in real-time, however the solutions failed to be robust, creating a future need for a stochastic optimization.

Acknowledgements

We would like to acknowledge several people who helped us complete this project and made us feel welcomed at their workplace. First, we thank our liaisons, Richard Jordan and Mariya Ishutkina from MIT Lincoln Laboratory, for providing us with the opportunity to work on this project. They helped to shape the project and suggested papers for us that helped us better understand the problem. MIT Lincoln Laboratory provided our office space and computers we used to complete this project. Members from Group 43 included us in their office-wide activities and also offered to help us with our project in any way possible.

Next we would like to thank our project advisors, Professor Jon Abraham and Professor George T. Heineman, for providing feedback on our report drafts and for their guidance throughout the term. This thank you is extended to Professor Ted Clancy for coordinating the project site at MIT Lincoln Laboratory. Finally, we thank Emily Anesta and Seth Hunter for their oversight of our project at MIT Lincoln Laboratory.

Executive Summary

Due to growing demand for air travel, surface traffic congestion at many of our nation's busiest airports causes significant delays. These delays add up over time, enough to raise costs for airlines in terms of fuel, maintenance, and staffing, and to cause passengers to miss connecting flights. These associated costs cause airlines and airports to adopt practices that minimize the effects of delay, though such practices have their own costs. These problems will only worsen as demand increases with time, as the Federal Aviation Administration (FAA) predicts it to on a large scale over the next 20 years. Several possible solutions exist for dealing with the problem of congestion and delays, including but not limited to expanding physical capacity, charging airlines more for using runways during peak hours, and limiting the number of Air Traffic Controller (ATC) operations per hour at some of the busiest airports.

This project, however, considers optimizing the sequence in which aircraft take off by minimizing departure delay while balancing safety, equity and efficiency. Much research has been done over the years to optimize aircraft departure sequences, though it has mainly focused on re-sequencing aircraft at the runway and has only considered the deterministic case, in which uncertainty is not taken into account. Before any optimization techniques can be deployed to the real world, however, they must first be able to satisfy two requirements in addition to those specified above: they must run to completion in a reasonable amount of time, and they must produce adequate results when real-world stochastic variables replace their deterministic counterparts in models that take the predictability of such variables for granted.

Our goal was to first develop solutions that optimize the sequence and time that aircraft are released from a designated area on the surface of the airport, called the "spot", from which they taxi to the runway. This effectively relocates delay from the runway to the spot, saving fuel because aircraft can shut off one or more of their engines while waiting at the spot, which they cannot do at the runway. Secondly, we model uncertainty by perturbing the amount of time it takes for aircraft to taxi from the spot to the runway based on real data from Dallas/ Fort Worth Airport (DFW), and then compare the resulting departure delay to that of the deterministic case where the taxi times are known and unchanging.

We utilize two methods for the deterministic optimization, which have both been used extensively in re-sequencing research: Mixed Integer Linear Programming (MILP) and Dynamic Programming (DP). The MILP formulation focused on the runway activities as they are predicted to be based on spot releases, whereas the DP formulation looked at the spot activities in the present while indirectly obeying future separation constraints at the runway. The sequences are optimized by finding the minimum departure delay based on separation requirements at the runway specified by the FAA, while being sure to respect operational constraints imposed on the spot release sequence. The separation requirements at the runway ensure safety while the operational constraints at the spot ensure equity by not allowing aircraft to be shifted too far from their original spot release position, which is called Constrained Position Shifting (CPS).

The biggest challenge in the implementation of the Mixed Integer Linear Programming optimization was efficiency; to make the problem solvable, we grouped into bins those flights within smaller windows of time. The main difficulty in optimizing with Dynamic Programming was that the problem did not appear to have an optimal substructure. That is, we were unable to define a relationship between steps n and $n+1$ such that the optimal solution for step n would be guaranteed to lead to the optimal solution for step $n+1$. Such a recursive relationship between adjacent steps is crucial for a DP algorithm to produce an exact, optimal solution. Ideally, if both methods were capable of computing a globally optimal solution, the spot release schedules produced by them would be identical, except possibly in cases where there exists more than one solution that gives the same minimal departure delay.

The DP and MILP were both suboptimal due to differences between spot sequences and runway sequences caused by differences in taxi times. We used DP heuristics instead of an exact solution and MILP used a relaxed CPS constraint that alters the results slightly in a positive direction. Therefore they do not give the same exact optimization delays, but the results are fairly close between the two and the optimizations can still be compared. The timing results proved that the optimizations are feasible in real-time situations. We ran the optimizations on Dell desktops that had four dual-core processors with 4GB of RAM. The MILP optimization took about 45 seconds in total to run on a full day of data split into 15-minute bins, independent of CPS. The DP optimization took under a second to build the graph and solve it using a CPS of one and about 30

seconds for a CPS of two. These timing results prove that the optimizations would be feasible in real-time situations. For a full day of flights, the DP heuristics cannot finish in a reasonable amount of time when CPS is set to three positions or higher. Similarly, MILP cannot finish in a reasonable amount of time if the day is not broken down into bins since its approach to the problem is computationally intractable.

We set our deterministic baseline to be First Come First Serve (FCFS) at the runway with enforced minimum separation requirements, because this would be the easiest improvement in practice. DFW already uses FCFS at the runway, but they do not enforce that flights take off as soon as they safely can. We expected any of our optimizations to be able to do better than FCFS and so we were surprised when they did not. We quickly realized that this was because FCFS is a CPS of zero at the runway, but we were imposing CPS at the spot. The optimization delays under uncertainties were larger than the non-optimized delays, proving that the optimization is not robust under stochastic conditions. The reason for this was that the aircraft were waiting at the spot until their optimal spot leave time, however the uncertainties in taxi times changed the times and sometimes sequence in which they arrived at the runway. This created delay at the runway which was added to the delay at the spot, which was often more than the non-optimized stochastic delay at the runway. This result shows that a deterministic optimization is not beneficial in a stochastic world.

The change in runway sequence adds another dimension to the measure of the robustness of the optimization solutions. We measured sequence change three different ways. First, we looked at the sum of spaces that each flight shifted from the optimal takeoff sequence due to taxi time perturbations. Next we looked at relative sequence change and weighted relative sequence change. The relative sequence change was measured by the number of aircraft pairs where aircraft *i* was supposed to come before aircraft *j* in the optimal sequence, but instead *j* came before *i* in the final sequence. The weighted relative sequence change added in weights for how many positions these pairs were separated by. For some days the sequence did not change at all and for others a mere two flights switch positions with each other. More common was that separation times were violated, causing extra delay at the runway.

The next addition to our project would be to add in arrival crossing to the optimization and again test the effects of uncertainty. Once arrivals are added into the optimization, effects of

uncertainty could be measured in the same way as the departures only optimization. Departure delay would be computed as the sum of the differences between when flights were ready at the runway and when they were assigned to either take off or cross. After a deterministic optimization of both departures and arrivals has been studied under stochastic conditions, the next step to further this project would be to create a stochastic optimization. This type of optimization would take into account uncertainties as it optimizes, creating a more robust solution.

Table of Contents

Table of Figures

Table of Tables

1.0 Introduction

The National Airspace System (NAS) is responsible for all aircraft activity on the surface and in the air above the United States. They are in charge of 14,500 air traffic controllers (ATC), 19,000 airports, and on average 50,000 flights per day (FAA, 2008). The ATC communicates with aircraft pilots to manage the effects of weather and traffic conditions on the safety of the aircraft. The Federal Aviation Administration (FAA) predicts that aircraft congestion will become worse in the future as many major airports will come close to doubling their number of passengers by 2030 ("Terminal Area Forecast Summary, Fiscal Years 2010-2030", 2010). This congestion affects many people and businesses, including passengers, local and ground controllers, airlines, and even the environment.

The capacity of an airport is primarily determined by its runways, but the ability of an airport to achieve this capacity without becoming overly congested is determined by the efficiency of ATC on the surface of the airport. Currently it is up to each individual controller to make decisions to reduce traffic and maintain safe conditions for the aircraft. Although the controllers undergo the same training, different controllers can handle situations in different ways. Research has recently begun to look into decision support tools to ease controller workload while also reducing congestion, delay, and improving the efficiency of airports. A few examples of what decisions must be made include assigning gates, taxiways, and runways; scheduling takeoff, landing, and runway crossing sequences and times; and the overall airport configuration.

Strategies to increase existing capacity include *metering*, *re-sequencing* aircraft at a spot on the surface of the airport, and *re-sequencing* at the runway. The benefit of *metering*, which entails holding aircraft until they can taxi unimpeded to the runway, is the reduction of congestion and fuel burn. *Re-sequencing* aircraft at the runway minimizes the delay that is experienced by ordering aircraft based on their weight classes. *Re-sequencing* at the spot accomplishes the same goal by planning ahead, attempting to avoid long queues at the runway in the first place. These techniques have been looked at to optimize aircraft traffic on the ground (Malik, 2010 and Balakrishnan, 2007), but there has not been a large enough focus on optimizing surface conditions while also looking at the effects of uncertainties, which is the focus of our project.

The goal of this project was to measure the effects of uncertain taxi times on optimal departure and arrival crossing sequences that are constructed by optimizing spot release times. We did this by implementing scheduling algorithms using Mixed Integer Linear Programming and Dynamic Programming that can operate in real-time, which were tested with real data from Dallas/Fort Worth International Airport (DFW). We compared the departure delay incurred by optimal sequences with deterministic taxi times against those with stochastic taxi times to quantify the effects of uncertainty. This helped us conclude whether deterministic scheduling algorithms are appropriate for sequencing aircraft at the spot or if more robust solutions that take uncertainty into account are required.

2.0 Background

The goal of this project is to test the effect of uncertainty on optimal recommended schedules for aircraft taking off and crossing a single runway. This chapter examines previous work in this area. First we address the responsibilities of the National Airspace System in controlling all aircraft and airports to avoid accidents and congestion. We then introduce several ways of reducing congestion on the surface of airports and detail the advantages and disadvantages of each. From there we describe two methods for finding an optimal departure takeoff and arrival crossing sequence. We use Mixed Integer Linear Programming and Dynamic Programming to set up and solve the problem. Finally the chapter concludes with the effects of uncertain taxi times on the optimal sequence and how to model and measure those effects.

2.1 National Airspace System (NAS)

The National Airspace System (NAS) deals with everything related to airspace above the United States. Airspace is a block of air at any given place or time. There are several classes of airspace, which dictate whether NAS has control over them or not and how well known the area is. NAS is in charge of 14,500 air traffic controllers (ATC), 19,000 airports, and on average 50,000 flights per day (FAA, 2008). Air Traffic Control is an important aspect of NAS. For example, on September 11, 2001, all aircraft were forced to land or turn around and return to their place of origin. Air Traffic Control was in charge of dealing with the large amount of planes that needed to land (Paramount Business Jets, 2011).

2.1.1 Air Traffic Control (ATC)

In the United States, Air Traffic Control (ATC) helps to avoid accidents between aircraft. In addition to regulating traffic in the air, ATC must also control traffic on the ground to ensure that the aircraft take off safely and in a timely manner. The responsibility of controlling traffic on the ground is shared among the Ramp, Ground, and Local Controllers. The generic layout of an airport is shown in **Error! Reference source not found.**. First, the Ramp Controller (RC) decides when aircraft should push back from the gate and begin taxiing towards a predetermined spot on the airport surface, labeled “Spots” in the figure. This happens in the upper section of the picture, labeled “Ramp”. The Ramp Controller is often an airline employee and so the focus of this project will be on the decision making of the Ground and Local Controllers from the spots to

takeoff. The Ground Controller (GC) tells the aircraft when to leave the spots and directs traffic so that all aircraft arrive safely and timely at the appropriate runway. The GC is in charge of the "Taxiway" area in the figure. Finally, the Local Controller (LC) manages the runway to avoid conflict by telling the aircraft when to take off. The LC is also in charge of arriving aircraft which need to cross the runway to reach the terminal. An arriving aircraft can be seen at the bottom of the figure and just below the word "Runway" as it waits to cross (Malik, 2010).

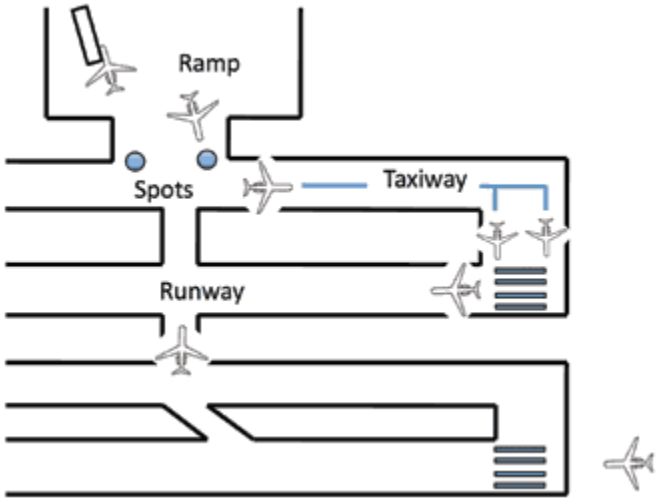


Figure 1: Generic Airport Configuration (NASA Aviation Systems Division, 2011)

2.1.2 The Bottleneck of an Airport

The capacity of an airport is determined by its runways, which are considered to be the bottleneck of the airport during peak hours (Malik, 2010). Throughput is the number of aircraft that take off during a certain time period. This rate is often used as a measure for the capacity of an airport. As the number of aircraft on the ground increases, the throughput of departing aircraft also increases. Once the number of aircraft becomes too large, throughput begins to saturate which causes delays. Throughput eventually becomes saturated because separation requirements imposed by the Federal Aviation Administration (FAA) dictate how long an aircraft must wait after another has just taken off. The saturation of throughput for Dallas/Fort Worth Airport is shown in Figure 2, parameterized by the number of arrival crossings. Currently the controllers tell aircraft to leave the gate and spots as soon as they safely can which can cause congestion at the taxiway and runway. This could be avoided with better surface decision tools to aid the controllers, which would allow more aircraft to take off. Other measures of how well an airport

is operating include its efficiency, which is measured by departure delay, and equity, which is measured by maximum delay.

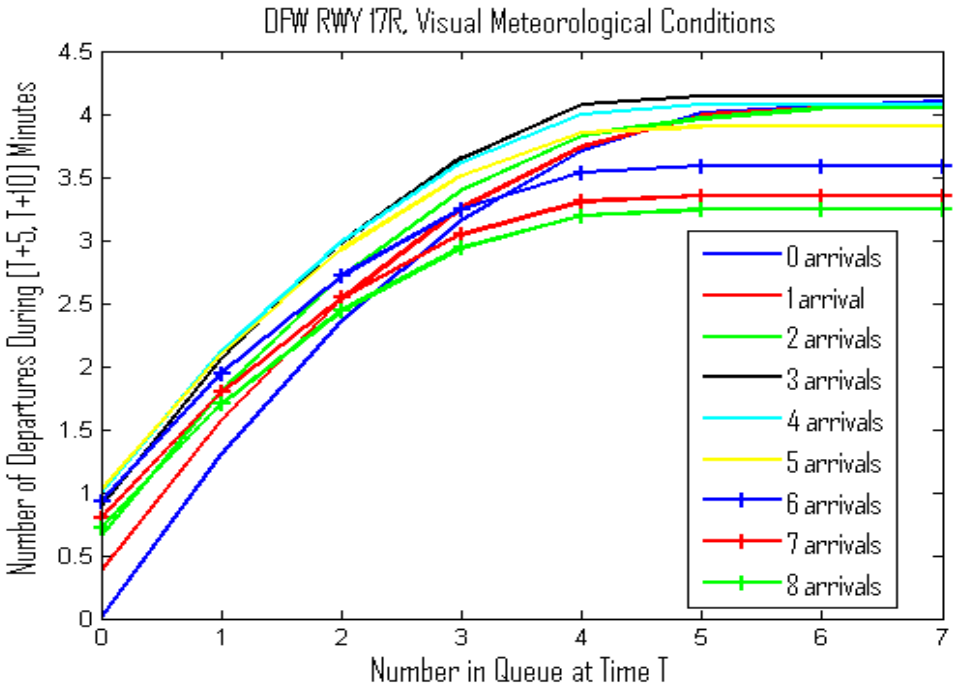


Figure 2: DFW Airport throughput saturation (Jordan, 2011, Personal Communication)

2.1.3 Issues of Congestion

The congestion of an airport affects many people and businesses, including passengers, local and ground controllers, airlines, and even the environment. Passengers are interested in getting to where they are going without any delay. The workload of controllers increases when there is traffic. They must make immediate decisions to attempt to reduce the congestion or at least avoid any dangerous situations. Stop and go traffic results in high fuel burn due to the high thrust needed to achieve taxi speeds (Malik, 2010). This raises fuel costs and affects the environment negatively. Additionally, missing appointed schedules increases costs for airlines in terms of pay for crews, maintenance, and the potential for cancelled flights.

The National Center of Excellence for Aviation Operations Research (NEXTOR) conducted a study to measure the overall cost of flight delay in 2007 (Ball et al, 2010). They measured direct costs similar to those listed above, but also considered the cost of lost demand, the costs to passengers when their flight is canceled or they miss a connection on a multi-leg trip, the indirect

cost to the U.S. economy, and many other subtle and indirect costs. As a result of their analysis, NEXTOR determined that in 2007, delay cost airlines $8.3 billion, passengers $16.7 billion, $2.2 billion in lost demand, and lowered the U.S. gross domestic product (GDP) by an estimated $4 billion, for a total overall cost of $31.2 billion. The authors note that other studies computed similarly high figures. They also note that though not all delay can be removed from the system due to untimely events such as adverse weather conditions or mechanical problems, they predict that at least 50-60% of the cost from delay in 2007 could have been eliminated had the system had a higher capacity and had more efficient procedures and policies been in place (Ball et al, 2010).

The Federal Aviation Administration (FAA) predicts that congestion will become significantly worse in the future. Air travel demand is expected to increase at a constant rate through at least the year 2030. In a report entitled "Terminal Area Forecast Summary, Fiscal Years 2010-2030" by the FAA, they predict that many major airports will come close to doubling their number of passengers by 2030. The top airports in terms of growth will be Washington Dulles, John F. Kennedy and Charlotte, which are forecast to grow by 4.2, 3.6 and 3.6 percent each year, respectively. Overall, the report forecast that the number of passengers at the airports that were studied would increase from 515,474,000 in 2009 to 927,498,000 in 2030 ("Terminal Area Forecast Summary, Fiscal Years 2010-2030", 2010).

2.2 Methods of Reducing Congestion

Because air travel demand in 2030 is expected to be almost double what it is currently, it is important to address the issue of airport capacity, which is widely considered to be the limiting factor in the expansion of the NAS. Lee discussed different categories of solutions to alleviate arrival congestion that also apply to departures: physically increasing capacity, shifting congestion from peak hours, and using existing airport capacity more efficiently (Lee, 2008).

2.2.1 Increasing Physical Capacity

One solution that would certainly increase the overall capacity of the NAS would be to extend existing airports by adding more runways, or by building new airports altogether. With more airports and runways, each individual runway in the system would have less demand, reducing congestion and delay and their associated problems.

However, building new airports or just adding more runways is not the ideal solution for two major reasons. The first reason is that doing either is very expensive and requires significant planning. The last major airport built in the U.S. was Denver International Airport (DIA) (Ishutkina, 2009), which cost $4.8 billion and took six years to construct, from when it began in 1989 to when it opened for business in April 1995. DIA's impact on the capacity of the NAS was limited in that it was built as a replacement for nearby Stapleton International Airport. Additionally, the final cost of the construction of the airport was nearly $1 billion more than its initial predicted cost, and the opening of the airport was delayed 16 months due to complications with its automated baggage system, which was added after the original planning of the airport (US General Accounting Office, 1995). The DIA case hardly inspires future airport construction in the United States without better planning in place.

The second reason is that the surrounding area may not lend itself to physical expansion. While DIA, among other airports such as Dallas/Fort Worth (DFW), were designed to be extendable to cater to future demand, not all airports have the luxury of having enough land to expand their physical capacity. Airports such as London's Heathrow and New York's JFK airports are in densely populated areas that provide either no physical room for expansion or whose plans for expansion are met with public opposition (Ishutkina, 2009). In Boston, it took 30 years for a new runway to be approved at Logan International Airport, and only on the condition that flights would not make overland takeoffs or arrivals, limiting the noise pollution experienced by the surrounding community (Marchi, 2005). Additionally, in anticipation of the new runway, the location of the Hyatt Harborside Hotel was carefully chosen by state legislators to be built directly in line with where the runway would be to limit its length and the resulting effect on the community (Howe, 2006).

2.2.2 Shifting Congestion from Peak Hours

Given the costs and other factors involved with building new airports and runways, another option to control congestion at existing airports is to shift flights away from peak hours through either rationing or changing the schedule. Rationing the runways would be done by either increasing fees to use them or by limiting the number of flights that can take off and land during peak hours. Boston's Logan Airport tried to do something like this in 1988 with their Program for Airfield Capacity Efficiency (PACE), but it was unsuccessful since it was determined to be

unreasonable and discriminatory to owners of small aircraft by the Department of Transportation. This was later upheld by the First Circuit Court of Appeals (Hardaway, 1991). Under this scheme, flat fees for landing were roughly quadrupled, while weight-based rates were significantly reduced. This caused significant reductions in fees for large aircraft, but the opposite for small aircraft, and was applied over the entire day rather than just for peak hours. This reduced congestion throughout the entire day. The unfairness of the plan led to the decision by the Department of Transportation to order Logan to abandon PACE (LA Times, 1988). A similar plan does not seem to have been attempted since then, though the FAA recently gave airport operators more flexibility in pricing landing slots (USDOT, 2008). Even if runways were rationed by the number of aircraft allowed to use them during a given time period, it would not increase the overall capacity of airports since the congestion of peak periods would just be reallocated to less-congested periods. Such periods may not exist for airports already running at nearly full capacity, such as Heathrow and JFK (Ishutkina, 2009).

Another idea that has been considered is to encourage customers to fly during non-peak periods, such as late at night, by offering reduced ticket prices. This however would be difficult to implement across the board since individual airlines determine their own ticket prices and they would not change their schedules unless there was a market for late night travel or some other incentive (Venkatakrishnan, 1991).

2.2.3 Decision Support Tools for ATC

Given the drawbacks to the first two approaches, more emphasis has been placed on increasing existing airport capacity through decision support tools for air traffic controllers. There are many decisions that must be made on the surface of an airport to ensure that it is operating safely and efficiently. Griffin *et al.* looked at the optimizations of most of these surface decisions. They developed five optimizations which they named the “Taxiway Planner”, “Runway Planner”, “Configuration Planner”, “Runway Assigner”, and “Gate Assigner” and optimized them in the order that they were presented (2010). The goal of the Taxiway Planner was to minimize taxi time, fuel burn, and emissions by controlling the gate pushback and achieving unimpeded taxi times. The Runway Planner maximized throughput by sequencing takeoffs and landings based on weight classes. These two decision support tools are named metering and re-sequencing and are discussed in greater detail below, as they are the tools that will be used in this project. The

Configuration Planner decides upon the best airport configuration to maximize airport capacity. An airport's configuration is a set of runways that are being used and depends on factors such as weather and traffic. The Runway Assigner assigns aircraft to a runway in order to minimize emission costs and balance runway usage. Finally the Gate Assigner assigns gates for aircraft so that ramp congestion is minimized. They tested these optimizations on data from Detroit Metropolitan Wayne County (DTW) and determined that the Taxiway Planner had the greatest effect on reducing taxi times and taxi time delay (Griffin, 2010).

Our project will look at parts of the "Taxiway Planner" and the "Runway Planner" optimizations. In "Managing departure aircraft release for efficient airport surface operations", researchers at the NASA Ames Research Center also considered the benefits of metering and re-sequencing as decision support tools for ATC (Malik, 2010). They used Dallas/Fort Worth International Airport as their model and focused on metering departure aircraft at the spot. Metering is the act of holding aircraft at a spot on the surface of the airport (see **Error! Reference source not found.**) until they are able to taxi unimpeded to the runway. The benefit of metering is the reduction of congestion and fuel burn. Metering reduces stop and go traffic between the spot and the runway. Aircraft may also be able to turn off one of their engines while at a spot for a further reduction of fuel burn. Simaiakis also looked at the effects of metering, but at Boston Logan. The aircraft were held at the gate instead of at a spot. This resulted in an average fuel savings of 16-20 gallons per aircraft that experienced a hold at the gate (Simaiakis, 2011).

Another method to reduce congestion is minimizing delay by re-sequencing the original takeoff order. When an aircraft takes off it leaves behind it a wake of turbulence. The next aircraft must then wait a certain amount of time before taking off in order to avoid this turbulence. The amount of time between the two takeoffs, called the separation time, depends on the aircraft types of both of the aircraft, the predominant factor being their size. Separation times are derived from the Federal Aviation Administration (FAA) separation distances based on average takeoff speeds for each aircraft type. Re-sequencing aircraft minimizes the delay that is experienced at the runway, by ordering aircraft based on separation times. This is known as the Departure Sequencing Problem (DSP). There are two places that DSP has been explored: the spot and the runway. The optimal sequence is the sequence with the least amount of delay that also obeys constraints such as separation times, position shifting, ready times, and priority departures. A

constraint on position shifting restricts the maximum number of positions that any given aircraft can shift from the original sequence to the optimized sequence, and is known as Constrained Position Shifting (CPS).

The algorithms for re-sequencing at the spot and at the runway are the same. The difference between them is whether the algorithm is looking into the future or solving the problem in the present. In the spot algorithm, once the optimal takeoff times have been solved for, it is determined at what time each aircraft should be released from the spot to begin taxiing towards the runway. In the runway algorithm, the aircraft are already at the runway ready to take off and the algorithm determines when they should take off and in what order. In other words, the spot optimization tells aircraft when to leave the spot by looking into the future at when they should take off. The runway optimization just tells the aircraft when to take off and operates in the present. For the runway optimization it is necessary to have multiple queues so that the aircraft can change the sequence that they are currently in. For example if there are three queues, the aircraft can line up in the outer two queues, leaving the inner queue clear for anyone to use it to approach the runway, independent of their queuing order. If such queues are not available then the optimization can instead look at departure-arrival re-sequencing, but not departure-departure re-sequencing.

2.2.4 Dallas/Fort Worth International Airport (DFW)

At some airports, such as Dallas/Fort Worth International Airport (DFW) shown in Figure 3, arrivals must cross the departure runway to reach the terminal. Such crossings therefore must be considered when re-sequencing aircraft. Additionally, the gain in throughput at any given airport depends on many factors, including the level of demand and the distribution of aircraft that take off. At DFW, the level of demand is not high enough to cause much congestion, so metering is of little benefit. Most of the aircraft that depart from DFW fall into the “large” weight class, meaning that not much time is gained by optimizing the sequence at the last spot. The aircraft distribution at DFW is 2% small, 88% large, 5% heavy, and 5% B-757 (Gupta, 2009). In fact, most of the gain in throughput at DFW would come from simply enforcing that aircraft take off as soon as they can while still respecting minimum separation times between takeoffs. Currently, aircraft wait longer to take off than necessary, for unknown reasons. While DFW is the focus of

the simulations in this project, our results will be applicable to airports with similar runway layouts but with higher demand.

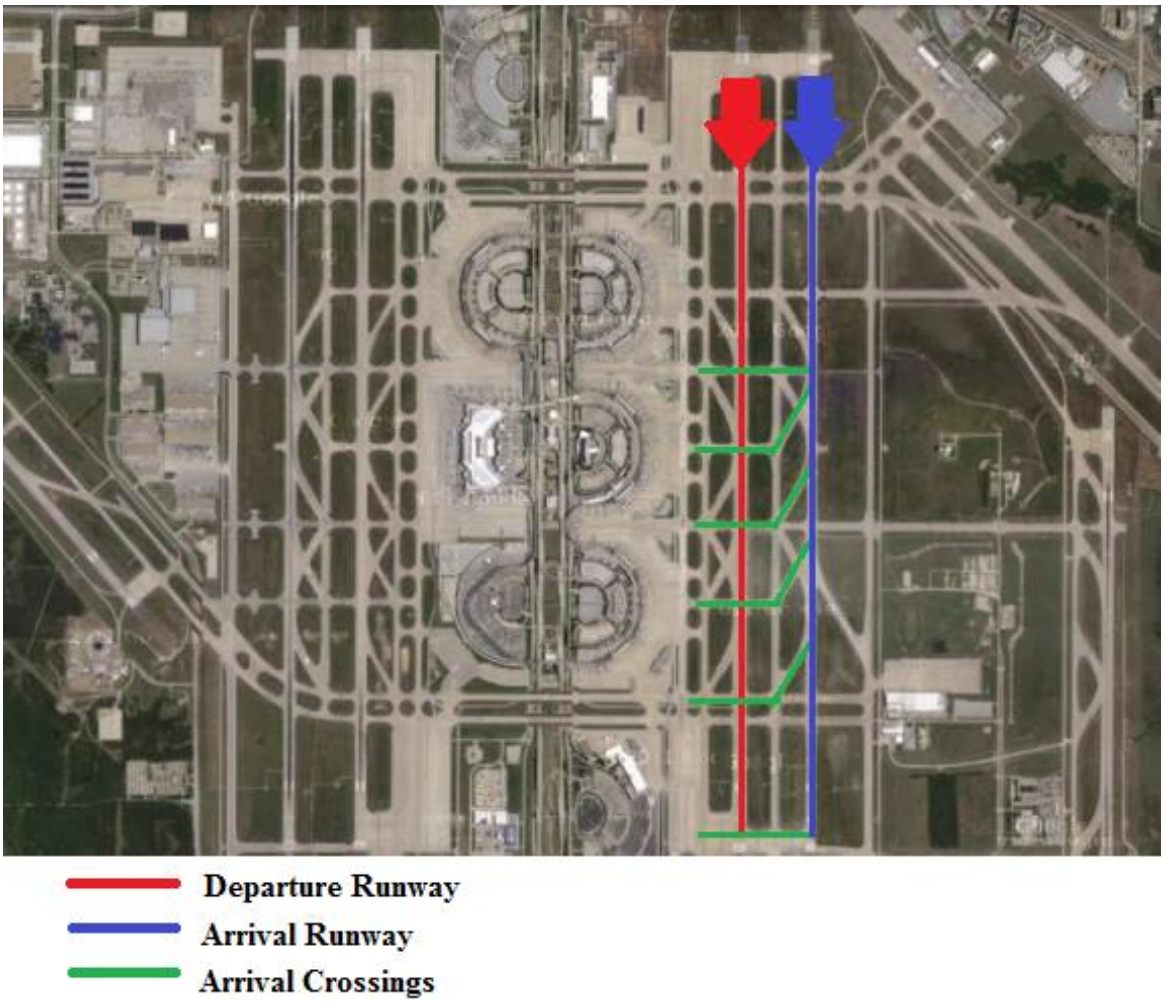


Figure 3: Arial view of DFW, provided by Google Satellite

2.3 Methods of Solving the Departure Sequencing Problem (DSP)

Several methods have been used to optimize the surface activities of aircraft. The surface activities that we focus on are metering at the spot, taxiing from the spot to the runway, and the sequence of departures taking off and arrivals crossing. There is a single runway in our problem that is based off of DFW and multiple queuing areas at the runway so that any aircraft can take off at any time regardless of its current position in the runway sequence. This project will investigate two methods to optimize the aircraft traffic on the ground: Mixed Integer Linear Programming (MILP) and Dynamic Programming (DP). Both of these methods have been used

to optimize aircraft traffic on the ground (Gupta, 2010 and Balakrishnan, 2007), but neither has done so while also taking into account realistic uncertainties.

2.3.1 Mixed Integer Linear Programming (MILP)

Mixed Integer Linear Programming is an NP-hard mathematical method used for planning activities. Linear Programs solve for a set of variables to minimize or maximize an *objective function* while also obeying certain constraints. A MILP is a Linear Program that has at least one integer constraint. This is the case for the traffic optimization in order to determine the sequence of the aircraft. Examples of objective functions that have been used include minimizing system delay, minimizing the maximum delay, or maximizing makespan (Gupta, 2009). The term maximum delay refers to the greatest amount of delay that a single flight incurs on a single takeoff or crossing. This is measured by subtracting when the flight was scheduled to takeoff or cross from when it actually did. Maximizing throughput is difficult to implement so often minimizing makespan is used instead (Gupta, 2009). Makespan is the time at which the last aircraft takes off. A small makespan means that all of the aircraft took off and crossed quickly.

Linear programs (LP) are often solved by an algorithm called the simplex method invented by George Dantzig (Dantzig, 1955). The constraints of the LP create a *polytope*, a multidimensional shape with many sides, which is known as the feasibility region. Any point in the *polytope* is a feasible solution to the LP. The vertices of the *polytope* are known as basic feasible solutions. The simplex method iteratively travels along the edge of the feasible region, visiting a path of vertices which are the basic feasible solutions, until an optimal solution is found. Each iterative step results in an improved basic feasible solution from the previous and so the problem can be solved in polynomial time for the average-case complexity. We will use the branch and cut (B&C) algorithm to solve the MILP for this project. B&C is a combination of the branch and bound (B&B) and the cutting planes algorithm (Chen, 2010). The first step of B&C is to convert the MILP into an LP by ignoring the integer constraints. This constructed LP is solved using the simplex method. If the optimal solution of the LP also obeys the integer constraints then the algorithm is done. Otherwise, the cutting plane algorithm is used iteratively to make the solution closer to an integer solution. Again if an integer optimal solution is found then the algorithm is done. If it has not been found and there are no more cutting planes, then the B&B algorithm is used (Chen, 2010).

The B&B algorithm can also be used alone to solve a MILP. B&B creates multiple sets whose union equals the feasible region of the MILP. In the case of the B&C algorithm, the MILP is divided into two versions of itself. The first version includes a new constraint variable that is greater than or equal to the next integer that is larger than the current solution. The second version does the same but the variable is less than or equal to the next smaller integer. The simplex method is used again to find the optimal solutions for both problems. This is repeated until an optimal solution has been found that satisfies all of the constraints of the original problem (Chen, 2010).

There are many current applications to Mixed Integer Linear Programming. It is frequently used in Operations Research (OR) (Chen, 2010). Almost a quarter of reports that were published from 1979-2006 in *Interfaces* used MILP. *Interfaces* is a journal of real-life applications for operations research and management science. Some applications of these reports include transportation and distribution, work staff scheduling, government services, and financial services to name a few. Transportation and distribution was the most common application of MILP in the reports that were investigated (Chen, 2010).

2.3.2 Dynamic Programming (DP)

Dynamic programming algorithms solve problems by decomposing them into simpler sub-problems that the eventual final solution depends upon. The first step to solving a dynamic programming problem is to represent it as a directed acyclic graph between defined starting and ending nodes. Once this graph has been developed, the shortest path from the start to end node is found using an iterative process. The basic sub-problem is the shortest path to the starting node, which is zero. This represents the base case. Once the base case is solved then subsequent sub-problems are solved using the solution from the base case. Once these subsequent sub-problems have been solved the algorithm continues to solve larger sub-problems and find shortest paths that use previous solutions until it reaches the end node. When it reaches the end node, there are no more sub-problems left to be solved and the shortest path through the whole graph has been found (Dasgupta, 2006).

Some common real-world applications of Dynamic Programming include finding the shortest path through a graph, which is used in routing protocols such as Open Shortest Path First (OSPF)

and Intermediate System to Intermediate System (IS-IS), sequence alignment, which has applications in bioinformatics, and interval scheduling, whose solutions can be applied to resource allocation. For the application of airport surface operations optimization, the optimal sequence is obtained by finding the shortest path through a directed acyclic graph where the nodes represent subsequences of the final sequence and the edges represent possible leader-follower relationships between subsequences. These edges are weighted according to the additional delay incurred by scheduling the aircraft in the target node after that of the source node. The shortest path through this graph with respect to the weighted edges is computed using dynamic programming, giving the optimal spot release sequence.

Although recursive in nature, similar to divide-and-conquer algorithms, dynamic programming does not break a problem into fractions of itself, but rather slightly smaller versions of itself that build upon each other. In fact, for many problems, the dynamic programming approach is much more efficient than its recursive counterpart, since it only needs to solve each sub-problem once, whereas the recursive algorithm solves each multiple times (Dasgupta, 2006). However, many dynamic programming problems are NP-hard. One example is the Travelling Salesman Problem (TSP), which involves a salesman who is attempting to travel to n cities in the shortest route possible, while only visiting each city once. An asymmetric problem instance of the TSP allows for different distances between the cities depending upon which direction the salesman is traveling, and is also intractable. Without constrained position shifting, the takeoff sequence optimization problem would also be intractable since it can be modeled as an asymmetric version of the Travelling Salesman Problem (TSP). This can be done by having the aircraft be the "cities", and having the separation times between aircraft based on their respective weight classes be the asymmetric "distances" between the cities. However, having constrained position shifting makes the problem tractable and solvable in nearly linear time (Balakrishnan, 2007). This efficiency makes such an algorithm a good candidate for real-time use, if it can also produce robust results when uncertainties are present.

2.4 Uncertainty of Taxi Times

Most papers on the subject of optimizing airport surface operations assume a deterministic optimization where everything is known. However, the real world is never deterministic. In order to make these optimization algorithms more appropriate in real circumstances, it is necessary to

consider stochastic, or changing, conditions. Several causes of uncertainties listed by Malik *et al*. include "weather, the airline schedule, ramp operations, aircraft turn-around time and other factors" (Malik, 2010). There are several places on the airport surface where uncertainties can be investigated. Some examples are when the aircraft were ready to leave the gate, spot or runway, when the aircraft did leave these areas, and their taxi times between them. This project will focus on the uncertainties of taxi times, determining their effect on the optimal takeoff schedule.

2.4.1 Effects of Taxiing Uncertainty on Spot Release Re-sequencing

Due to the uncertainty in taxi times from the spot to the runway, the original planned optimal takeoff schedule becomes sub-optimal in one of three ways: a leading aircraft taxies for a longer amount of time than expected, a following aircraft taxies for a shorter amount of time than expected, or two aircraft switch places in the predicted takeoff sequence due to perturbations in their predicted unimpeded taxi times. If a leading aircraft taxies longer than expected, it backs up the rest of the sequence due to separation requirements, assuming that the rest of the aircraft were spaced no further apart than those separation times. Similarly a following aircraft that taxies shorter than expected will incur delay at the runway in order to obey separation times. Matters become worse when a pair of leading and trailing aircraft taxi slower and faster than expected, respectively, which can lead to a change in the takeoff sequence. Since separation times are based on the weight classes of both the leading and trailing aircraft, such changes in the takeoff sequence can make the resulting sequence considerably sub-optimal, depending on the weight classes of the aircraft that were switched and of the aircraft immediately before and following them.

2.4.2 Modeling Taxi Time Uncertainty

Stochastic taxi times can be modeled in several different ways. The first and least accurate would be to randomly choose them from a uniform distribution. This is inaccurate because there is an equal chance of choosing a taxi time on the outer boundary of the distribution as choosing one closer to the mean, which is not seen frequently in real-life. A more accurate prediction would come from a Gaussian distribution, as more taxi times that are closer to the mean would be chosen. An even more accurate method for modeling real data from DFW is to use a triangular distribution that is dependent on several characteristics of the aircraft. This triangular distribution

was taken from the 10th percentile of taxi times from 13 different days of data from DFW. The 10th percentile was taken in order to model unimpeded taxi times. The characteristics that are used to model the taxi times can include how far the aircraft is taxiing, where in the airport it is coming from, and the aircraft type. These characteristics determine the interval of the triangular distribution that is to be used. Examples of these distributions are shown in Figure 4. The randomness of choosing from a distribution allows for a fairly accurate model of the possible uncertainties of taxi times (Jordan, 2011, Personal Communication).

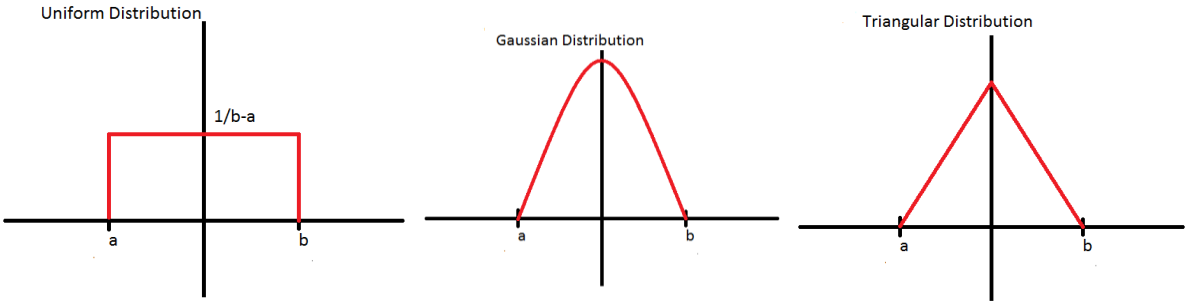


Figure 4: From left to right: Uniform, Gaussian and Triangular Distributions

2.4.3 Measuring Robustness

In his Master's thesis "Tradeoff evaluation of scheduling algorithms for terminal-area air traffic control," Lee describes two methods to measure the robustness of a predetermined arrival schedule when considering the uncertainty of predictions of when aircraft will be ready to land. The first was termed *reliability* by Balakrishnan, and measured the probability that separation requirements would be violated in a given landing schedule. According to Lee, only measuring the overall probability for the entire schedule could sacrifice the separation time between a pair of aircraft for the overall reliability of the sequence, and therefore proposes an alternative (Lee, 2008).

Lee proposes a second method that minimizes the probability of pairs of aircraft violating separation requirements rather than looking at the sequence as a whole. In his approach, he considers the situation where a trailing aircraft is ready to land before it is allowed to under separation requirements. What he terms the *weakness* of a sequence is the maximum of the probabilities of a pair of aircraft being ready to land too close to each other, for all pairs of adjacent aircraft in the landing schedule (Lee, 2008).

Though these two approaches were applied to arrival sequencing, they can be applied to both departure and arrival sequencing and minimizing system delay by considering the probabilities of delay being incurred by uncertain taxi times, possibly weighted by the amount of delay that could occur. With an appropriate measure of robustness, existing algorithms can be modified and new algorithms can be developed that balance the tradeoffs between minimizing delay and maximizing robustness, much like what Lee did with arrivals.

3.0 Methodology

In this project, we optimized the takeoffs on a single runway by minimizing departure delay. We had also planned to consider runway crossings of arriving aircraft, but were unable to due to difficulties with the simpler problem of just departures. Optimizations require an objective function, in our case minimizing departure delay, which was the goal of the optimization. In addition there were certain constraints that restrict the variables which were being solved. We were solving for optimal takeoff times and an optimal sequence, optimal meaning those which gave the minimal departure delay. The primary constraints that we imposed by the Mixed Integer Linear Programming (MILP) and Dynamic Programming (DP) methods included separation times and Constrained Position Shifting (CPS). Both MILP and DP have been used by other researchers to optimize aircraft surface operations (Malik, 2010 and Balakrishnan, 2010). However, they have only been investigated under deterministic, or unchanging, circumstances. Our overall project goal was to compare the operational feasibility of the MILP and DP methods and the robustness of the solutions when stochastic variables were added into the optimization problem. Our objectives to complete this project were as follows:

1. Use Mixed Integer Linear Programming and Dynamic Programming with deterministic variables to find the optimal sequence of departing and arrival aircraft.
2. Model uncertainties in taxi times based on real aircraft departure data from Dallas/Fort Worth International Airport (DFW).
3. Analyze robustness of the optimization solution and the feasibility of Mixed Integer Linear Programming and Dynamic Programming methods under stochastic conditions.

3.1 Problem Approach

In order to present our progress to our advisors and liaisons, we set up weekly meetings. During these meetings we discussed issues that we were working through, issues that had been resolved, and results that we found. This allowed us to get valuable feedback and also ideas to try out during the coming week that would help us discover deeper results or understandings. We wrote up and sent out a meeting agenda before each meeting to prepare the attendees for what topics would be discussed. We also took turns mediating and scribing the meeting. The scribe of each meeting then had the task of sending out meeting minutes to summarize what was talked about and actions that we planned to take.

During the fourth meeting, we came to a decision that altered the project goal. We discovered that the departure-only problem with stochastic taxi times was a great enough feat for a nine week project and that adding in arrivals would not be possible. We were especially unsure whether we would be able to develop a Dynamic Programming algorithm good enough to improve on our baseline. Since our goal was to compare Mixed Integer Linear Programming and Dynamic Programming approaches to the problem, it would not have been helpful to have a departure and arrival solution for just the MILP. Once we completed our departure algorithms, we came up with ideas of how arrivals might be added into the optimizations.

3.2 Deterministic Optimization using Mixed Integer Linear Programming and Dynamic Programming

We investigated how Mixed Integer Linear Programming and Dynamic Programming approach the sequence optimization problem. Julia Baum implemented a MILP algorithm using MATLAB while William Hawkins implemented DP algorithms in Java. For each approach, a version of the problem that only considers departures was implemented and tested. Smaller data sets were tested initially, which was determined by a limited time window or a small number of flights. Once testing proved that our implementations of these algorithms provided the correct solutions for the smaller sets, we moved on to testing 24-hour departure schedules. The outcomes of these methods are optimized takeoff times, an optimized takeoff sequence and corresponding spot release schedule, and the departure delay associated with the optimization. We compared the departure delays of the optimization, simulated First Come First Serve (FCFS), and real-world takeoff times against one another. Simulated FCFS showed the benefits of waiting no longer than the minimum separation times to take off when compared to the real-world data from DFW. Currently, aircraft wait longer than necessary at the runway before taking off, for unknown reasons. The optimization was expected to have even more benefits due to the re-sequencing of aircraft.

In coding our respective algorithms, we strove to follow best coding practices in terms of the readability of the text itself as well as the maintainability and flexibility of the design. We tested our code on a regular basis and used the appropriate tools to do so, such as testing suites and code coverage tools. We saved several incremental versions of our code in case we needed to refer back to previous versions. We also designed our algorithms to be extendable to minimize

our reliance on older versions of the code. We appropriately documented our code to facilitate future use by other developers at MIT Lincoln Laboratory.

While we both worked on our own algorithms, Julia Baum served the role of expert in terms of the mathematics and background information involved in the problem, as she spent a great amount of time researching and understanding the problem during an internship prior to the start of this project. William Hawkins was the expert on matters relating to coding, such as what tools to use to accomplish our project goals and what design concepts and coding practices to adopt to maximize the flexibility and maintainability of our code for future use by other developers.

3.2.1 Dallas/Fort Worth International Airport Data and its Limitations

The test data that was used for the Mixed Integer Linear Programming and Dynamic Programming optimizations were real-life departure data from Dallas/Fort Worth (DFW) International Airport, since the layout of its runways contains the essential elements we intended to investigate. The data from DFW for departures consists of the time at which each aircraft was ready at the last spot, when it took off, and the aircraft type. The times that the aircraft were ready at the spot and their time of takeoff are actual times that radar detected for each flight that day. The unimpeded taxi times are defined as the 10^{th} percentile of taxi times based on a data sample consisting of 13 fair weather days. The ready to take off time is then estimated by adding the spot ready time and the unimpeded taxi time together. This assumes that each aircraft left the spot as soon as it was ready and taxied unimpeded to the runway.

Table 1 contains a sample fragment of the available data. The ready times are seconds after time zero in Zulu time, commonly known as Greenwich Mean Time (GMT). The unimpeded times are also in seconds, representing the amount of time it takes for the aircraft to taxi from the spot to the runway without stopping in between or encountering any traffic. The aircraft types are represented by the numbers 1-4 where 1 is a small aircraft, 2 is a large aircraft, 3 is a heavy aircraft, and 4 is a B757. The times at which departing aircraft were ready at the runway were used to generate the First Come First Serve (FCFS) sequences for MILP, and their spot ready times were used to generate the FCFS sequences for the DP algorithms. Aircraft were then assigned a number based on their position in the FCFS sequence in order to keep track of them during the optimization. The departure delays of the real world and FCFS sequence, which were

used as a baseline for those of the optimized MILP and DP solutions, was calculated by summing the differences between the takeoff times and the ready to take off times for each aircraft of the test data. The real-world system delay is shown in the table. FCFS is not shown in the table as it is simulated and does not come directly from the DFW data.

Ready to leave spot (S_i)	Unimpeded taxi time (u_i)	At runway and ready to take off (α_i)	Aircraft type (weight class 1-4)	Actual take off time	Delay
43219	529	43748	4	43783	35
60986	420	61406	2	61406	0
85961	490	86451	2	86537	86
…	…	…	…	…	

Table 1: Example data from DFW

We have four days' worth of data from DFW in 2010: June 14th, 15th, 16th, and 18th. All four days have similar congestion and aircraft type distribution, as DFW is fairly consistent day to day. The number of aircraft for these days are 485, 382, 393, and 404, respectively. There is a 76-minute gap in the data from June 15th where no flight information was recorded; other gaps in the data also appear. These gaps occur due to poor radar signals. Spot group number nine, for example, is known for being a blind spot to the radar, and so there are less spot leave times for aircraft leaving from that spot (Ishutkina, 2011, Personal Communication). In addition to poor radar, some pilots do not turn on their transponder when they begin taxiing and so their spot position is lost. For the four days for which we have data, an average of 469 flights departed each day, but we only have data for an average of 416 flights a day. This means that we are missing data for about 10% of the aircraft that are departing. The radar coverage is better at the runway compared to at the spot, and so there is more data on runway times and departure times than on spot times. Appendix C contains graphs and charts that analyze the four days of data. This analysis further shows the similarity between the days of data.

Since Dallas/Fort Worth International Airport has such a homogenous aircraft mix, we decided to uniformly assign aircraft types to the spot ready times that we had from DFW for each day. This created an even distribution of aircraft types, which would get more benefits from re-sequencing. We assigned these aircraft new taxi times taken from the triangular distribution, based on their aircraft types. We did not assign them new spot groups, however. We then

repeated the deterministic optimization on this data set with both optimization methods and different values of CPS.

3.2.2 Mixed Integer Linear Programming Formulation

We chose to use YALMIP to formulate the MILP in MATLAB. YALMIP is a toolbox for MATLAB and is used for modeling and solving optimization problems ("YALMIP Wiki", 2011). It allows a user who is familiar with MATLAB to more easily implement an optimization formulation. YALMIP has several commands that are useful for the MILP formulation. The first is *spdvar* which defines a decision variable, in our case the takeoff times. Similarly, *binvar* sets up a decision variable that is binary, which is used for the integer sequence variable. Finally *solvesdp* is the command to solve for the linear program. The inputs of *solvesdp* include the objective function, constraints, chosen solver, and other setting options. YALMIP includes an internal algorithm to solve for MILP problems, but also allows for external solvers to be used ("YALMIP Wiki", 2011). We used Gurobi as our external solver. Gurobi is one of the best commercial solvers available ("Gurobi Optimization", 2011). There are other solvers that could be used to solve the problem at hand. For example, Gupta *et al.* used ILOG CPLEX as their solver (2009). The algorithm that we used in Gurobi is Branch and Cut, which was described in the Background Chapter.

Optimizing a full day's worth of flights is operationally infeasible using MILP. To solve for a day of flights in less than five minutes, we split the day into 15-minute windows or bins. The optimization of June 14th took over five minutes for 30-minute bins, otherwise larger bins would have been chosen as they result in lower departure delays. The 15-minute bins are determined by runway ready times starting at the minimum ready time for each day. Each 15-minute bin is locally optimized one after the other. Subsequent bins are dependent on the optimization of the last flight of the previous bin. The MILP algorithm stores the optimal takeoff time and the aircraft type of the last flight in the previous bin to maintain separation time requirements between bins. The MILP was set up as shown in the formulation below, where variables were inputs and parameters were solved for:

Variables for MILP

n = number of aircraft, $1 \leq i \leq n$ and $1 \leq j \leq n$

k = number of positions CPS allows an aircraft to shift

α_i = time at which aircraft i is at the runway and ready to take off

$\Delta_{i,j}$ = the separation times between the i^{th} and j^{th} aircraft in the final sequence, based on sequence, weight classes, 1 = small, 2 = large, 3 = heavy, and 4 = B757

M = very large number, e.g. $\sum_{i=1}^{n} \alpha_i$

t_{last} = the takeoff time for the last aircraft in the previous 15-minute bin

$type_{last}$ = the aircraft type of the last aircraft to take off in the previous 15-minute bin

p_i = the position of the i^{th} aircraft in the original spot sequence

m_i = the position of the i^{th} aircraft in the optimal spot sequence determined by $x_{i,j}$

So_i = the optimal spot leave time for the i^{th} aircraft

Parameters for MILP

t_i = time at which aircraft i should take off, determined by optimization

$$z_{i,j} = \begin{cases} 1 & \text{if aircraft } i \text{ takes off before aircraft } j \\ 0 & \text{if aircraft } i \text{ does not take off before aircraft } j \end{cases}$$

$$x_{i,j} = \begin{cases} 1 & \text{if aircraft } i \text{ leaves the spot before aircraft } j \\ 0 & \text{if aircraft } i \text{ does not leave the spot before aircraft } j \end{cases}$$

Objective for MILP

Minimize departure delay

$$\min \quad F(t)$$

$$\text{where } F(t) = \sum_{i=1}^{n} (t_i - \alpha_i)$$

Constraints for MILP

1. An aircraft cannot take off before it is ready to.

$$t_i \geq \alpha_i \quad \forall i$$

2. If aircraft i is before aircraft j then aircraft j is not before aircraft$i$.

$$z_{i,j} + z_{j,i} = 1 \quad \forall i,j \quad i \neq j$$

$$x_{i,j} + x_{j,i} = 1 \quad \forall i,j \quad i \neq j$$

3. Separation times between aircraft are obeyed.

$$t_j - t_i \geq \Delta_{i,j} * z_{i,j} - (1 - z_{i,j}) * M \quad \forall i,j$$

$$\Delta = \begin{bmatrix} 60 & 60 & 60 & 60 \\ 90 & 60 & 60 & 60 \\ 120 & 120 & 90 & 120 \\ 120 & 90 & 90 & 90 \end{bmatrix} \quad leading\ rows, trailing\ columns$$

4. No aircraft leaves at the same time in the optimal spot leave schedule.

$$So_j - So_i \geq 1 * x_{i,j} - (1 - x_{i,j}) * M \quad \forall i,j$$

5. Separation times between bins are obeyed.

$$t_i - t_{last} \geq \Delta_{i,type_{last}} \quad \forall i$$

6. Constrained Position Shifting (CPS): aircraft can only shift *k* places from their position in the original spot leave sequence.

$$|p_i - m_i| \leq k$$

Constraints three and four are the integer constraints that make this problem a Mixed Integer Linear Program problem. $z_{i,j}$ and $x_{i,j}$are both binary variables and they determine whether two aircraft are next to each other in the sequence. Both of these constraints should only be enforced for aircraft that are next to each other, or the problem will run very slowly. In order to make a constraint only be enforced some of the time we used an either-or constraint. This is why the $-(1 - z_{i,j}) * M$ part of the constraint is added. If $z_{i,j}$ or $x_{i,j}$ is equal to 1, meaning that *i* and *j* are adjacent in the sequence, then this part of the constraint equals zero and the separation times are enforced. On the other hand if $z_{i,j}$ or $x_{i,j}$ is equal to 0 then the second half of the constraint dominates the constraint but leaves the separation time to be greater than a large negative number, which in practice negates the constraint.

Delta represents the separation time requirements between two aircraft. This time depends on the weight classes of both of the aircraft and on which aircraft is before the other in the sequence. The departure separation time matrix has rows and columns of weight classes from 1-4. An example of how to interpret this Δ matrix is if a B757 aircraft (weight class 4) takes off before a small aircraft (weight class 1) then the small aircraft will have to wait for 120 seconds. Once the whole day has been solved for, the code returns the optimal takeoff times, the optimal sequence, and the departure delay of both FCFS and the optimal takeoff and crossing times for comparison. From the optimal takeoff times and the optimal takeoff sequence, optimal spot leave times and spot leave sequence for departures were found by subtracting the unimpeded taxi times. This allows the departing aircraft to be held at the spot until their optimal spot leave time. They then taxi unimpeded and take off as soon as they arrive at the runway.

3.2.3 Dynamic Programming Formulation

The Dynamic Programming formulation appears to be quite different from the MILP, although it obeys the same constraints. The MILP formulation focused on the runway activities as they

would be in the future, whereas the DP formulation looks at the spot activities in the present while indirectly obeying future separation constraints at the runway. Due to this indirect constraint, the DP algorithm proved to be more difficult than was expected. This was due to the fact that in order for a DP algorithm to guarantee optimal solutions to a problem, the problem must have a property called "optimal substructure." This means that when the DP algorithm breaks down the problem recursively into smaller steps, the optimal solution from a given step must be reachable from the decisions made to reach the optimal solution of the step before it. Due to the indirect CPS constraint on the runway sequence and that the spot release and runway sequences differ due to differences in unimpeded taxi times, we were unable to reach such a recursive relationship between the optimal solutions of successive steps for this problem. Therefore, instead of pursuing an exact solution, we implemented a series of heuristic DP algorithms instead. The DP heuristics can solve for the whole day at once, as long as the maximum allowable number of position shifts experienced by any aircraft, k, is kept at low values such as 1 or 2, which are considered typical for both arrival and departure scheduling (Balakrishnan, 2010).

CPS Network

The optimization problem was modeled as finding the path of least cost through a directed acyclic graph, which is a graph of nodes and unidirectional edges with no cycles. The dynamic programming algorithms depend on the valid construction of this graph, or the "CPS network", as defined by Balakrishnan and Chandran (2010). The construction of such a network satisfies the CPS constraint in that every path from the start node s to the end node t represents a sequence that obeys CPS and contains no repeated aircraft. Additionally, every possible sequence that meets these criteria is represented as an s-t path through the network.

To start, the CPS network generation algorithm we implemented creates a table that lists all the possible aircraft (represented by their index in the FCFS spot ready sequence) that could be placed at each position in the optimal spot release schedule. For the example below, n=4 and k=1:

Position	1	2	3	4
Possible aircraft assignments	1	1	2	3
	2	2	3	4
		3	4	

Using this table, the algorithm generates subsequences of aircraft that obey CPS constraints, which are represented as nodes in the CPS network. These nodes are placed into "stages," which represent the positions of the final optimal spot release sequence that the aircraft can be assigned to. These stages consist of groups of nodes whose last aircraft could possibly be assigned to the position that stage represents. For example, all nodes in stage 3 must have aircraft 2, 3 or 4 as their last aircraft. All previous aircraft in a node are used only to correctly draw edges between nodes in neighboring stages, as described later.

To create these nodes and place them in the correct stages, the algorithm begins with stage 1. For this stage and all subsequent stages, the number of aircraft that are represented by nodes in stage p is$min(2k + 1,\ p)$. For stage 1, this means all of its nodes have subsequences of length 1. Since the last aircraft of all nodes in stage *p* must be assignable to position *p* in the optimal spot release sequence with respect to CPS, stage 1 has just two nodes: one that represents just aircraft 1, and one that represents aircraft 2.

Next it builds stage 2, in which nodes represent subsequences of length two, according to the process described above. The algorithm constructs nodes for all possible subsequences of length two that follow the CPS constraint, and produces the following: 1-2, 1-3, 2-1, and 2-3, obviously omitting 1-1 and 2-2 since aircraft cannot repeat in a sequence. This process continues until nodes are generated for stage 4, or more generally, stage *n*, which in this case has nodes with subsequences of length three since $2k + 1 = 3 < p = 4$. Finally, dummy nodes *s* and *t* are added, which are the start and end nodes, respectively. The completed CPS network is shown in Figure 5.

Once the nodes are determined, they are connected with edges. Beginning with stage 1, the algorithm draws a directed edge from every node in stage 1 to every node in stage 2 that could follow it. In this example, these relationships are (1)→(1-2), (1)→(1-3), (2)→(2-1), and (2)→(2-

3). Then, it moves on to stage 2 and does the same for nodes in stages 2 and 3. More formally, edges are drawn from all nodes *i* in stage *p* to all nodes *j* in stage *p+1* where the last $min(2k, p)$ aircraft of *i* match the first $min(2k, p)$ aircraft of node *j*. See Figure 5 for all such valid relationships between nodes in adjacent stages.

Finally, there are some nodes that cannot be part of an *s*-*t* path because they are not reachable from both *s* and *t*. We used a graph search algorithm that first marks all nodes reachable from *s*, then reverses the edges in the network and does the same but starting from *t*. Any nodes not marked twice can safely be removed from the network, along with all their associated edges. The nodes that are removed from the network in Figure 5 are shaded gray. This pruned version of the original CPS network is the network on which we perform the dynamic programming recursion. The properties of the network that allow us to use it to solve the problem are proven by Balakrishnan and Chandran, who have used networks like this to solve other aircraft sequencing problems with CPS and dynamic programming (Balakrishnan, 2010).

Although our solution relies on Balakrishnan and Chandran's CPS network, they solved the problem of re-sequencing at the runway while our problem involved re-sequencing at the spot. Therefore, our initial solution made the adaptation of changing what the edge costs in the network represent and how they were set. Balakrishnan and Chandran set the costs of the edges to the required separation times between the last aircraft of the nodes they connected, and did so when they constructed the network. In our problem, edge costs correspond to the additional delay incurred by scheduling the last aircraft of the target node next in the spot release sequence. Our network starts with all edge costs set to zero, which are then implicitly calculated on the fly as the problem is solved since they rely on the solutions to the nodes from which they originate.

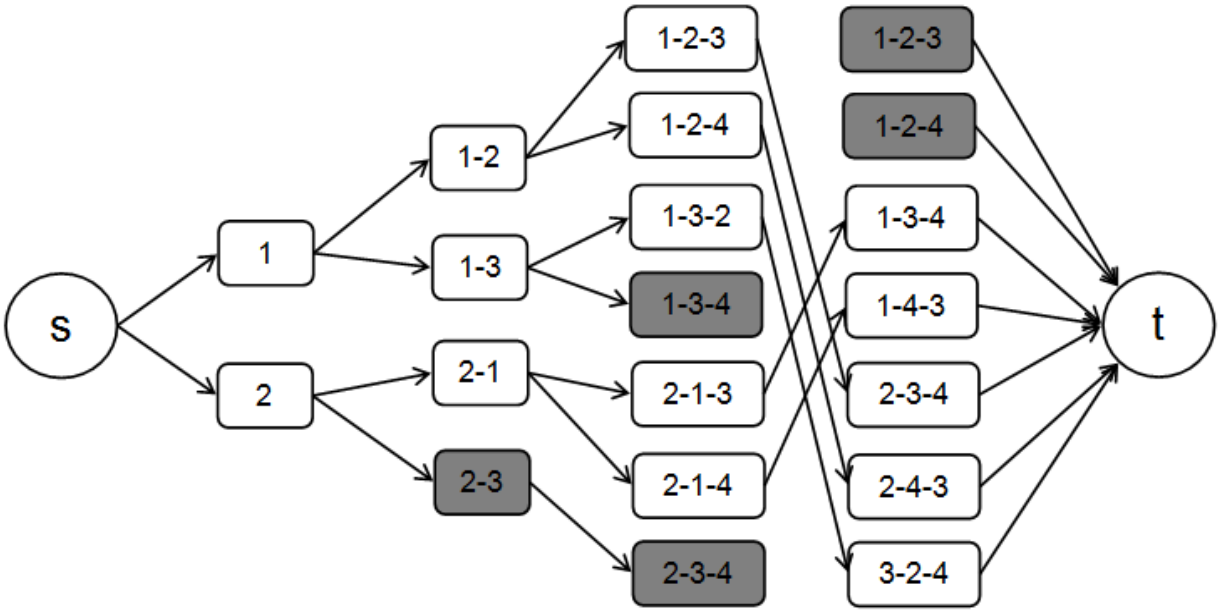


Figure 5: The directed acyclic graph on which the problem of optimally sequencing four takeoffs is solved

Variables for DP

n	the number of aircraft in the sequence
k	the value of the CPS parameter
last(i)	the last aircraft in the subsequence of node *i* of the CPS network
s_i	the spot ready time for *last(i)*
u_i	the unimpeded taxi time for *last(i)*
So_i	the actual release time of *last(i)*
$\Delta_{i,j}$	the required separation time between *last(i)* and *last(j)* at the runway
P(i)	the set of nodes that are predecessors of node *i*

Parameters for DP

$So_j(i)$	the spot release time for *last(j)* that minimizes delay if *last(i)* is its predecessor
D(i)	the minimum possible departure delay incurred by a sequence ending in *last(i)*

Change of variables between formulations of MILP and DP:

$$t_i = So_i + u_i$$
$$\alpha_i = s_i + u_i$$

Objective for DP

$$min\left(\sum_i (So_i - s_i)\right)$$

Dynamic Programming Recursion

Once we implemented the CPS network generation algorithm, we moved on to developing heuristics that used the network to solve the problem. These algorithms had to respect the following constraints:

- An aircraft cannot leave the spot before it is ready to.

$$So_i \geq s_i \quad \forall i$$

- Separation times are obeyed.

$$(So_j + u_j) - (So_i + u_i) \geq \Delta_{i,j} \rightarrow \quad So_j - So_i \geq \Delta_{i,j} - (u_j - u_i) \quad \forall i, j$$

- Constrained Position Shifting (CPS): aircraft can only shift zero, one, or two places from their original spot leave sequence (for departures only).

The last of these three constraints was taken care of by the construction of the network, leaving the other two to be enforced while applying the dynamic programming recursive equation.

The most basic situation to consider was that the spot release and takeoff sequences were the same, so it can be assumed that *last(j)* comes immediately after *last(i)* in the takeoff sequence if that is the order in which they appear in the spot release sequence. Therefore, the following must be true:

$$So_j \geq \Delta_{i,j} - (u_j - u_i) + So_i$$

In practice, the goal is to have So_j be as early as possible, and the earliest aircraft *last(j)* can possibly leave the spot is s_j. Additionally, the order of the current CPS subsequence being looked at must be preserved, meaning that *last(j)* could leave no earlier than So_i. We called this the "predecessor bound." Therefore, the time that *last(j)* should leave is a maximum of these three times, if preceded by node *i*:

$$So_j(i) = max(\Delta_{i,j} - (u_j - u_i) + So_i,\ s_j,\ So_i), i \in P(j)$$

These additional constraints resulted in the following recursive function D(·), which gives the lowest possible departure delay for a sequence ending in node *j*, minimized over all possible predecessor nodes *i*:

$$D(j) = min\{So_j(i) + D(i)\} - s_j, \forall i \in P(j)$$

However, the definition of *D(j)* assumes that the goal is to have *last(j)* not get to the runway until *last(i)* has already taken off and that the required separation time between the two has already passed. As mentioned above, this enforces that the spot release and takeoff sequences are the same. For the simplicity of this initial heuristic, we named it ***Simple Spot***. Where it falls short is where there exists the possibility that the two sequences could differ due to large differences in unimpeded taxi times, which could give a better solution.

Consider the following scenario:

	s_i	u_i	Aircraft Type
A	0	100	2
B	400	200	2
C	405	50	2

Applying the *D(·)* function to each node in the generated CPS network resulted in a minimum departure delay of five seconds, and with the spot sequence being A-C-B. In reality, the best spot sequence would have been A-B-C, with no departure delay. The reason why this occurs is because C is able to leave early enough so that it can fit between A and B in the takeoff sequence without incurring any delay at the runway.

Any time there is sufficient space between the spot ready times of two consecutive aircraft, where the third aircraft in the spot ready sequence is ready very soon after the second one is and has a shorter unimpeded taxi time to the runway, the ***Simple Spot*** heuristic computes an incorrect result if the third aircraft can fit between the first two in the takeoff sequence without consequences.

The fundamental problem with ***Simple Spot*** was that it did not take into account that the takeoff sequence could be different from the spot release sequence due to large differences in unimpeded taxi times between different aircraft. To recognize this possibility, two improvements to this algorithm were implemented: ***One Gap*** and ***All Gaps***. ***One Gap*** works exactly like ***Simple Spot***, except that it considers the current predicted takeoff sequence first. If it is determined that the

aircraft currently being scheduled can fit between the last two aircraft projected to take off without incurring any runway delay and respecting all other constraints, it is allowed to do so. Otherwise, it is scheduled to take off last. ***All Gaps*** is a modification of ***One Gap*** that looks at gaps between all aircraft in the projected takeoff sequence rather than just between the last two. While ***One Gap*** expanded on the original heuristic and ***All Gaps*** even more so, the resulting departure delays given by these algorithms were only a slight improvement over ***Simple Spot*** and were much higher than the runway departure delay incurred by just sending all aircraft from the spot to the runway immediately after they are ready to do so.

Although ***One Gap*** and ***All Gaps*** were developed to recognize that the spot release and takeoff sequences may differ substantially from each other due to differences in unimpeded taxi times, they only consider gaps in the takeoff sequence that are already large enough to fit subsequent aircraft into. That is, a gap between the takeoffs of previously scheduled aircraft *a* and *b* is only considered suitable if the aircraft currently being scheduled, *c*, can take off as soon as it gets to the runway without following *a* too closely with respect to the required separation time between the two of them, and if this can be done without *b* following *c* too closely. This approach fails to consider gaps in the projected takeoff sequence that are only slightly smaller than necessary to fit *c*. With a small adjustment to *b*'s takeoff time (i.e. an adjustment to its spot release time), the gap can be made wide enough to fit *c* between *a* and *b*, incurring far less delay than scheduling *c* to take off after *b*.

The first algorithm we developed to address this issue is ***One Gap Force***, which extends ***One Gap*** to consider "forcing" the gap between the last two aircraft in the current projected takeoff sequence to be large enough to fit the aircraft currently being scheduled. As it schedules each aircraft, ***One Gap Force*** computes the additional delay incurred by "forcing" *b* to leave the spot later to accommodate *c*, and then does the same for just scheduling *c* to take off after *b*. The spot release time for *c* that corresponds to the lesser of the two delays is then assigned to *c* as its optimal spot release time. If the two incur equal delays, then the one that places *c* between *a* and *b* in the takeoff sequence is preferred since it minimizes the time at which the next aircraft to be scheduled after *c* can take off. The one exception to this is if rescheduling *b*'s takeoff (and therefore spot release) time changes the spot release sequence enough to make it violate the CPS constraint, in which case the other spot release time is taken instead.

In implementing this algorithm and all subsequent algorithms that relied on "forcing," three other important changes had to be made:

1. Each node must keep its own optimal spot release sequence that is set once that node is solved for and not changed afterwards
2. The "predecessor bound" must be calculated from the stored spot release sequence of the predecessor node *i* currently being considered rather than just So_i
3. The optimal spot release sequence for the entire day is that which is stored in the node in stage *n* with the lowest departure delay

The first change was necessary because "forcing gaps" changes values that have already been solved for, and also has the potential to change the sequence in which aircraft are released from the spot, which is dependent upon the spot ready time, unimpeded taxi time, and aircraft type of the next aircraft being scheduled. Since nodes in the CPS network often have more than one next node, and since by construction of the network the last aircraft of those nodes must be different, they had the potential to have very different effects on the spot release sequence that had already been scheduled. Therefore, given node *i* with next nodes *x* and *y*, where *x* is solved for first, *x* cannot just adjust So_i as it sees fit because it would bias the results of *y*, which also relies on the value of So_i. Therefore, each node *i* must maintain its own optimal release sequence, which it creates by copying from what is determined to be its optimal predecessor node and then modifies when it schedules *last(i)*.

The second change arose from the fact that the previously scheduled sequence can change due to "forcing" gaps and may not follow the sequence that would normally result from tracing back through the nodes in the CPS network. For example, a node *i* in stage 3 could have been originally constructed to represent the subsequence 1-2-3, but whose optimal sequence could actually represent 1-3-2 due to forcing. This could happen if 3 is fit into the takeoff gap between 1 and 2, but in order to do so, 2's spot release time is adjusted enough to release it after 3 even though 3 is the aircraft currently being scheduled. This is perfectly acceptable since the new subsequence of 1-3-2 is valid according to CPS. However, if a node *j* in stage 4 that is connected to *i* then uses 3's optimal release time in its calculation of how early it is allowed to schedule aircraft 4, it is possible that it could schedule 4 before 2, creating the sequence 1-3-4-2, which violates 1-CPS since 2 would have moved twice from its original position. This is not a problem

if the forcing part of the algorithm does this since it is always validated against CPS before being accepted. However, scheduling the next aircraft to be the last takeoff can also create an invalid sequence in this way, and that, rightly so, is not validated against CPS because it would then leave the algorithm no alternative for assigning an optimal spot release time to the aircraft currently under consideration.

The solution we came up with to avoid invalid sequences is to consider the entire sequence of the predecessor node currently being looked at to compute the predecessor bound rather than just looking at the aircraft that was scheduled last. This can be done by starting at the last aircraft in the spot release sequence and moving backward, stopping once an aircraft is reached that *must* precede the aircraft currently being scheduled in order to preserve the validity of the sequence with respect to CPS. There are two conditions to validate that determine whether such an aircraft has been found. The first is that you have gone too far back in the sequence for the current aircraft. For example, if k=1 and you are currently scheduling aircraft 6, this condition would tell you to stop once you look at the aircraft in the fourth position of the optimal spot release sequence. This means aircraft 6 must follow that aircraft, i.e. appear at least fifth in the sequence. The second condition is that you have reached an aircraft that has already been moved k places later than its original position, and therefore you cannot schedule the current aircraft before it.

Once such an aircraft is found, the predecessor bound on the earliest time the aircraft currently being scheduled can be released can be calculated. An important note is that this implementation considers the possibility of releasing two aircraft at the same time, and considers whichever aircraft that has the lowest First Come First Serve index to be first in the optimal sequence, even though in practice this does not matter. Therefore, if the aircraft currently being scheduled has a higher FCFS index than the one it must follow, it is safe to set the predecessor bound to the optimal release time of the aircraft it must follow. Otherwise, the bound is set to one second later than the found aircraft's optimal spot release time. In the case that there are no aircraft in the sequence that must precede the aircraft currently being scheduled, any suitably low value such as the current aircraft's spot ready time or even zero can be set as the bound since the maximum of this value and the aircraft's spot ready time will be taken as the earliest it can be scheduled anyway. It may make more sense operationally in the future to consider spot release schedules

that do not schedule multiple aircraft to be released at the same time. The MILP implementation already enforces this restriction.

The third change is needed because the spot release sequence created by following the path of least cost through the network will probably differ from the actual best sequence found, again due to "forcing gaps," which changes the sequences. Therefore, instead of tracing backwards through the network to recover the optimal spot release sequence, it instead is simply the sequence stored in the node in stage *n* that is the shortest distance from *s*, i.e. the lowest departure delay.

With all this in mind, the next "forcing" algorithm we implemented was ***All Gaps / One Gap Force*** (AG/OGF). This algorithm simply improves ***One Gap Force*** by also considering the spot release time suggested by ***All Gaps*** for each aircraft and takes the release time that results in the least amount of additional delay. Note that the ***All Gaps*** component of this algorithm must also use the new strategy of computing the predecessor bound rather than using So_i as it normally does.

Our next improvement to ***One Gap Force*** was ***All Gaps Force*** (AGF), which is to OGF as ***All Gaps*** is to ***One Gap***. It takes the logic of forcing gaps from OGF and extends it to considering all possible gaps in the takeoff sequence that could be forced to be wide enough to fit the aircraft currently being scheduled. An important difference, though, is that it is not enough to only adjust the spot release time of the following aircraft of the gap, *b*, because *b* is no longer guaranteed to always refer to the very last aircraft in the takeoff sequence since all gaps are now being considered. In addition to *b* being rescheduled, the algorithm must then determine if any runway delay is incurred by moving *b*'s takeoff time too close to the aircraft following it. If this is the case, then that aircraft must also be moved until there is no longer any conflict. This process must continue until there is no runway delay. The added delay is now the sum of all the additional delays incurred by rescheduling however many aircraft needed to be to avoid runway delay, added to whatever delay the aircraft currently being scheduled may also experience. To calculate the best sequence under AGF, the best resulting CPS-valid sequence should be maintained while trying each and every possible gap, which should then be compared to just scheduling the aircraft currently under consideration after the last takeoff.

Finally, one last improvement to this family of Dynamic Programming heuristic algorithms that we considered and implemented was ***Lenient All Gaps Force***. This algorithm operates the same way as ***All Gaps Force***, but keeps track of two sequences that violate CPS in addition to keeping track of a valid sequence. The first of these two invalid sequences is the one that is closest to being valid of all the invalid sequences that are tied for having the lowest departure delay, and the second is the sequence with the lowest departure delay of all the invalid sequences that are closest to being valid. How close an invalid sequence is to being valid is calculated simply by summing how far away each aircraft in the sequence is from being in a valid position according to CPS. For example, if aircraft 6 is in position 3 with k=1, then 2 is added to the sum since it is two positions away from position 5, the closest valid position it can occupy. Nothing is added to the sum for aircraft that are already within k positions of their original position. The motivation for such an algorithm was the possibility that intermediate invalid sequences could correct themselves due to forcing and end up having better departure delays than those that are constantly ensured to be valid throughout the scheduling process. A summary of the various DP algorithms and their relationships to one another is shown in Figure 6.

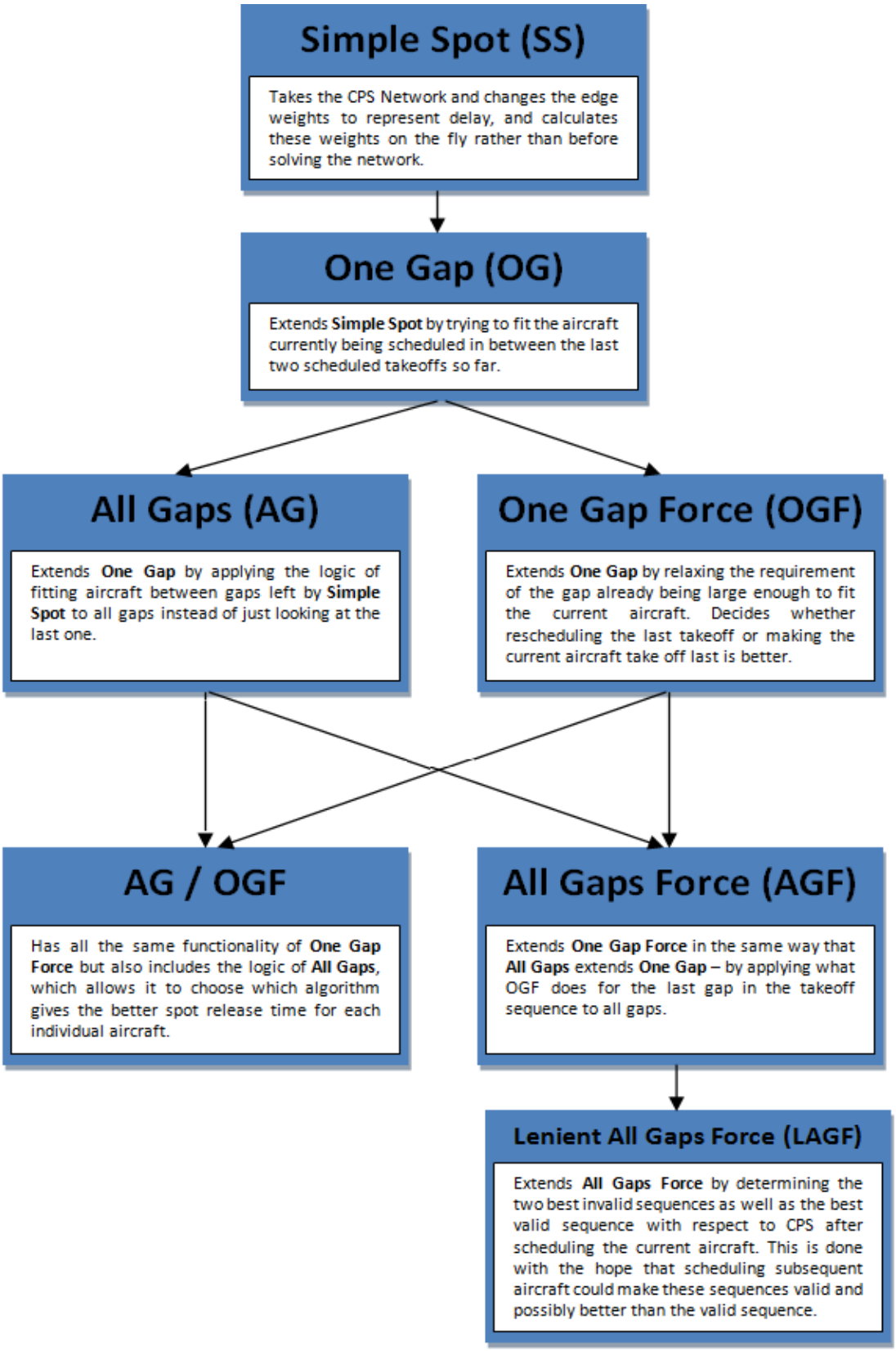


Figure 6: Summary of relationships between the different DP algorithms

Implementation

The above heuristic algorithms were all implemented using the Java programming language, whose object-oriented properties allowed for quick and easy extensions of earlier algorithms to develop the more complicated algorithms. The language features that were mainly responsible for this were interfaces/abstract classes, composition and polymorphism.

Coding to interfaces and abstract classes allowed for classes that needed to perform similar tasks but in different ways to easily be integrated into the program. Different algorithms were implemented by changing which classes the main class, *DPSequencer*, used to carry out these tasks needed to solve the problem. This often involved just changing which subclass of the *CostStrategy* class was used, whose role was to determine the optimal previous node for a given node and that node's optimal distance from the start node *s*.

CostStrategy, along with other helper classes, were used by the *DPSequencer* class according to the Strategy pattern. This allowed *DPSequencer* to handle the tasks that were common to all the DP heuristics, only calling on the strategy classes it was composed of for tasks that differed from algorithm to algorithm. As more types of strategy classes were added, we decided to also use the Abstract Factory pattern. This pattern involves passing just one object to the *DPSequencer* class that retrieves its helper objects from the factory object passed in. What this did was ensure that there would only ever need to be one argument to *DPSequencer*'s constructor, and that each algorithm would have a factory object associated with it. This meant that strategy objects would only be packaged in factories together in ways that made coherent and correct algorithms, rather than allowing any combination of strategy objects to be passed in to a *DPSequencer* constructor that accepts a strategy object of each type that it requires.

By taking advantage of the object-oriented properties of Java as well as the Strategy and Abstract Factory design patterns, we were able to quickly implement and test new DP heuristics without writing much additional code. Testing was done with the JUnit framework, which allowed us to verify that changes made in the new algorithms did not cause them to behave differently than we expected. We also wrote code that validated the new spot release schedules produced by our algorithms against each of the constraints and made this part of the optimization process so that we were guaranteed to know whether each and every result was valid. This allowed us to

eventually develop DP heuristics that did better than our baseline. Also, because each algorithm has its own factory object and because we chose subclassing instead of modifying the code whenever we implemented a new algorithm, we were able to run any of our heuristics that we had already developed, even the earliest ones, at any time. This allowed and continues to allow for comparisons between all the algorithms we developed for other test data we did not have time to run the algorithms on, and also allows for improvements to some of the earlier algorithms in ways that take a different path than that taken by the later algorithms that we implemented.

3.3 Stochastic Model with Mixed Integer Linear Programming and Dynamic Programming

To model uncertainty in the optimized models, once the optimal departure sequence was determined, we perturbed the taxi times from the last spot to the runway for each aircraft. These perturbations were chosen randomly from a triangular distribution modeled on the 10^{th} percentile of taxi times from DFW data, an example of which is shown in the figure below. We ran 500 simulations, each with different taxi times created by the model, for both of the algorithms being used to solve the problem (MILP and DP). We then took the average departure delay for each algorithm, along with an average of how much the sequence changed from the optimal sequence and an average of how many separation times were violated and by how many seconds. The results were fairly close together and so we did not feel the need to delete any outliers. This was carried out by using the stochastic unimpeded times along with the optimal spot leave times that were found during the deterministic optimization to get stochastic runway ready times. The stochastic runway ready times determined the FCFS sequence which the aircraft will take off in with the appropriate amount of separation. Additionally the stochastic unimpeded taxi times were added to the original spot ready times and again the flights were sent in FCFS. This modeled a stochastic world without optimization. We repeated this whole process with the data

of even distribution of aircraft types.

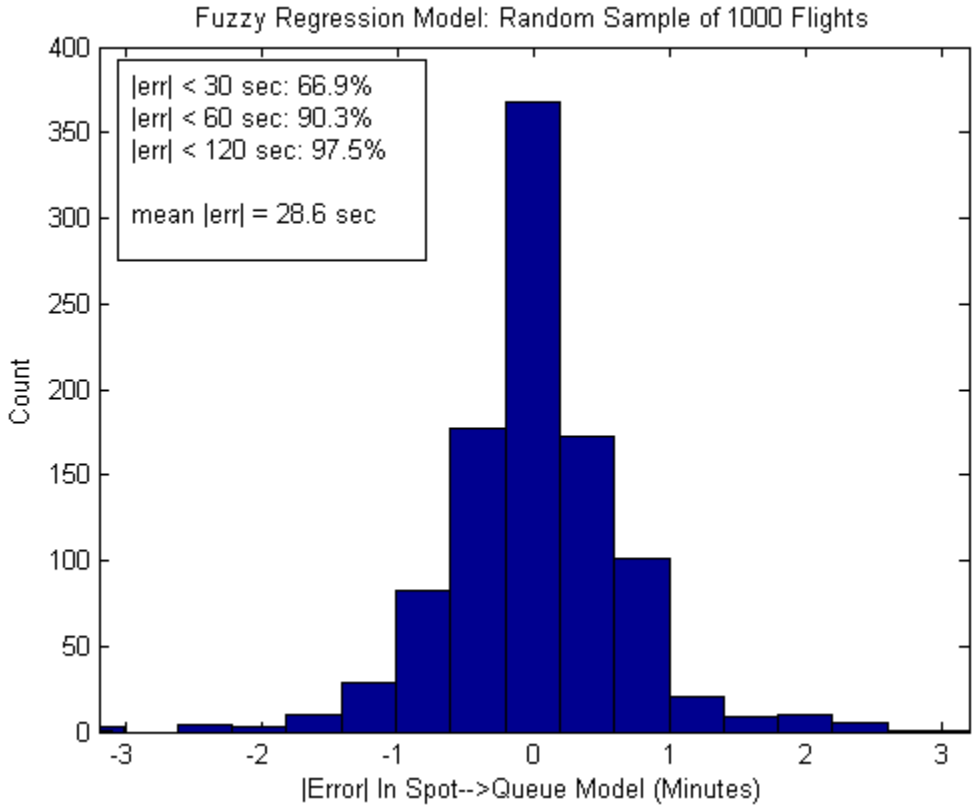


Figure 7: Triangular distribution modeled from Dallas/Fort Worth Airport data

3.4Analyze Effects of Uncertainties in Taxi Times

Adding in stochastic taxi times sometimes resulted in different sequences than the optimized sequence at the runway, and more frequently situations where aircraft incurred delay at the runway due to arriving at the runway too soon after a previous aircraft took off. The stochastic FCFS sequence at the runway was the sequence that was compared to the optimal runway sequence that the deterministic optimization found. The departure delay for the stochastic optimization was the sum of the stochastic takeoff times minus the original predicted runway ready times without holding at the spot.

Once we completed and ran the implementations of our algorithms, we had five sets of sequences and departure delays to compare for both of the aircraft distributions that we looked at. The first two were deterministic models, with and without optimizing. The next two were stochastic models, with and without optimizing. The fifth set was taken from the real takeoff

times captured at DFW. Comparing these results helped us determine if optimization with MILP or DP still had significant benefits when uncertainties like taxi times were taken into account in the model. This gave an idea of how the methods would perform as part of a decision support tool. These results are shown averaged in Table 6 in the Results Chapter and for each day individually in Appendix D.

We measured robustness in terms of the effects that uncertainties had on the departure delay and the order of the optimal sequence compared to those of the deterministic model. In order to examine the operational feasibility of the two methods, we measured their running times on desktop computers. This ensured that any runtimes we achieved would be achievable on hardware available to air traffic controllers in the control tower. Through the comparison of robustness of the solution and feasibility of the MILP and DP methods, we determined whether the benefits of optimizing the sequence are still significant when a stochastic variable, such as taxi times, is taken into account.

4.0 Results

Optimizing the takeoff times and sequences proved to be quite challenging, and therefore due to time restrictions we were unable to add in arrival crossings to the optimization. Therefore our results focus solely on departure-only optimizations. This section describes the difficulties that we encountered while implementing the Mixed Integer Linear Programming and Dynamic Programming algorithms. We then present the results that we obtained from our taxi time uncertainty analysis.

4.1 Difficulties in Implementation for Mixed Integer Linear Programming

The biggest challenge in the implementation of the Mixed Integer Linear Programming optimization was the need to bin the flights into smaller windows of time in order to make the problem solvable in less than five minutes. Table 2 shows timing results and departure delays for all four days with different bin sizes using no Constrained Position Shifting (CPS) constraint. We chose to display the timing results of no CPS because there are more sequence options that the solver has to search through and so the runtime is longer than with a CPS constraint. As the bins became larger, the departure delay improved, however the runtime became slower. The optimization was able to finish for three of the days with larger bin sizes, but the 14th of June took five minutes to solve the optimization with 30-minute bins and crashed MATLAB when it attempted to solve for hour-long bins. For this reason we decided to use 15-minute bins for the MILP optimization in order to get better departure delay without going over our runtime goal. As mentioned in the methodology, the bins are dependent on each other and so it was necessary to save information from the previous bin to use while solving for the current bin.

	June 14th (485)		June 15th (382)		June 16th (393)		June 18th (404)	
Bin Size	Time	Delay	Time	Delay	Time	Delay	Time	Delay
10 minutes	55.4	19726	36.31	11780	35.02	8835	35.7	10513
15 minutes	62.31	19190	39.12	11606	39.39	8617	40.17	9804
30 minutes	313.95	18894	60.73	11612	51.34	8465	58.32	9819
1 hour	DNF	N/A	104.76	11536	94.91	8441	101.46	9765
2 hour	DNF	N/A	DNF	N/A	194.65	8341	457.87	9743
4 hour	DNF	N/A	DNF	N/A	871.4	8401	DNF	N/A
8 hour	DNF	N/A	DNF	N/A	5059.1	8312	DNF	N/A
Full Day	DNF	N/A	DNF	N/A	DNF	N/A	DNF	N/A

Table 2: Timing and departure delay results for four days of data with different bin sizes (in seconds)

Unlike the Dynamic Programming algorithm, the Mixed Integer Linear Programming optimization is able to run without a constraint on position shifting. Due to this the algorithm was coded first with the other constraints and then Constrained Position Shifting (CPS) was added in later. Due to binning, this proved to be quite difficult. We wanted to constrain position shifting at the spot but not at the runway. The bins were divided by runway ready times; however this caused a problem when determining the spot sequence of one bin. Differing unimpeded taxi times created the possibility that a flight earlier in the spot leave sequence could appear in a later bin because it had a very large taxi time and it would therefore be ignored during the present bin optimization. One idea was to switch the optimization constraints and binning boundaries to be based on spot leave times. This would create the same problem when determining separation times at the runway as there could be an aircraft that takes off near a flight in the current bin, yet it does not show up until the next bin. The only solution is to either have large enough bins or sparse enough data so that this type of overlap does not happen. CPS sometimes failed in between bins, but was obeyed inside of bins and so only as many as eight flights out of a whole day disobeyed CPS and only by a position or two. This created an algorithm with a relaxed CPS constraint that gave similar results to one with an actual CPS constraint.

4.2 Difficulties in Implementation for Dynamic Programming

The main difficulty in optimizing the takeoff sequences with Dynamic Programming was that the problem did not appear to have an optimal substructure. That is, we were unable to define a relationship between steps n and $n+1$ of solving the problem such that the optimal solution for step n would be guaranteed to lead to the optimal solution for step $n+1$. Such a recursive relationship between adjacent steps is crucial for a DP algorithm to produce an exact, optimal solution.

Much research has been done using DP to solve the Departure Scheduling Problem, though all solutions we could find for the problem dealt only with re-sequencing aircraft that are already at the runway rather than re-sequencing at the spot instead to indirectly create favorable runway sequences. Because of this important difference, past research has looked at applying CPS directly to the takeoff sequence since that is the only sequence considered, but in our problem, the CPS applies instead to the spot release sequence, creating an indirect CPS constraint on the runway sequence.

In order to avoid the complexity of an indirect CPS constraint, we applied Balakrishnan's and Chandran's CPS network to the spot release sequence, which ensures that every valid spot release sequence with respect to CPS is tested, and no others. However, due to the nature of our problem, it created a scenario where the decisions made in scheduling the optimal sequence for *n* aircraft were not the decisions that would have been made for an optimal schedule of *n+1* aircraft, meaning there was no guarantee that there would be a reliable recursive relationship between the optimal solutions of adjacent steps in the problem. After spending much time trying to devise a DP algorithm that could create such a relationship, we decided to develop a series of heuristics instead due to time constraints. These heuristics were built to use the CPS network of Balakrishnan and Chandran, which we recognized as the best representation of a network that could be used by a DP algorithm that takes all the operational constraints of departure scheduling into account.

There was no ready-made solution for building the CPS network prescribed by Balakrishnan and Chandran, so we used our own method that used a network of its own to generate the CPS network. We later implemented a slightly more efficient algorithm that was iterative in nature, using the generated nodes of the previously constructed stage to build the nodes in the next stage. Table 3 shows the running times for how long it took to generate CPS networks for different *n* and *k*, since the particular DP algorithm being used has no bearing on the generation of the network. The first four *n* are the number of flights for each of the days we looked at. The time to generate networks for 50 aircraft was also considered since it is closer to the typical number of aircraft that would be scheduled at once in practice.

	N=485	N=382	N=393	N=404	N=50
K=1	.016	.013	.013	.014	.004
K=2	13.953	10.792	11.082	11.792	1.391

Table 3: Running time for generating CPS networks for given n and k (in seconds)

As Balakrishnan and Chandran state, such networks need not even be created on the fly, but could be stored in files that are read in based on the given *n* and *k* (Balakrishnan, 2010). This could be especially useful if there are cases where 3-CPS would be possible, since it would eliminate the large running time required to generate such a network. This assumes a DP algorithm that could solve such a network in a reasonable amount of time.

Due to the lack of optimal substructure, the original heuristic, called ***Simple Spot***, performed much worse than simulated FCFS. However, through various improvements as explained in the Methodology, we arrived at solutions that produced results that were slightly better than our baseline, simulated FCFS. The running time and departure delay results of each of these algorithms are shown in Tables 4 (1-CPS) and 5 (2-CPS). The best departure delays for each day are in bold.

	June 14th (485)		June 15th (382)		June 16th (393)		June 18th (404)	
	Time	Delay	Time	Delay	Time	Delay	Time	Delay
SS	.022	42388	.020	21987	.026	21884	.026	19301
OG	.292	29644	.226	16583	.253	12435	.252	13620
AG	.363	27324	.276	15882	.297	11412	.304	12525
OGF	.459	21638	.375	12695	.398	9267	.397	10792
AG/OGF	.638	20427	.731	12695	.532	**9017**	.543	**10205**
AGF	.751	**19797**	.717	**12416**	.506	9037	.510	10374
LAGF	1.624	**19797**	1.226	**12416**	.992	9037	1.016	10374

Table 4: Running time and departure delay at the spot for various DP algorithms for 1-CPS (in seconds)

	June 14th (485)		June 15th (382)		June 16th (393)		June 18th (404)	
	Time	Delay	Time	Delay	Time	Delay	Time	Delay
SS	.110	33172	.099	19370	.093	19408	.094	16287
OG	8.823	24683	4.347	15446	5.268	11111	5.705	11693
AG	13.816	23670	7.861	14746	8.715	10234	9.311	11349
OGF	12.341	**19056**	7.030	**12006**	8.414	**8414**	8.275	10037
AG/OGF	22.489	**19056**	14.081	**12006**	14.048	**8414**	14.140	**9776**
AGF	36.261	19541	19.933	12162	20.580	8615	19.149	9883
LAGF	107.227	19147	58.922	12162	58.212	8615	57.729	9887

Table 5: Running time and departure delay at the spot for various DP algorithms for 2-CPS (in seconds)

Interestingly, ***All Gaps Force*** (AGF) and ***Lenient All Gaps Force*** (LAGF) come up with the same exact results in all but a few instances, though AGF does so considerably faster. While these two algorithms are the best in general for 1-CPS, ***All Gaps / One Gap Force*** (AG/OGF) performs better than them both for all four days for 2-CPS. Even ***One Gap Force*** performs better than the last two for three out of the four days, for which it is tied with AG/OGF. This could be due to the benefit of using increasingly complex algorithms is reduced when more shifting is allowed, or that the algorithms that allow more forcing allow for more opportunities to make an

intermediate decision that hurts the rest of the sequence more than it helps the aircraft currently being scheduled.

It appears there is a relationship between the number of aircraft to sequence and the average delay for each aircraft in the optimized sequence, with the exception of 6/15, whose 382 flights experience an average delay higher than that of the 393 flights on 6/16 and the 404 flights on 6/18. A more sophisticated metric that measures how clustered spot ready times are could explain the observed differences in average delay between the four dates above better.

4.3 Robustness and Feasibility of Mixed Integer Linear Programming and Dynamic Programming Optimizations

We use departure delay to show the optimality of each takeoff schedule since minimizing departure delay was our objective function. Comparing the departure delays associated with deterministic and stochastic unimpeded times allowed us to determine the effects of uncertainties and whether our optimizations are robust enough to use in the real world. We also looked at the change in sequences at the runway as another measure of the effects of uncertainties. Operational feasibility of the optimizations was determined by runtimes.

Departure Delays per Aircraft, in seconds								
		Baseline		Deterministic		Stochastic		
Distribution of aircraft type		Real World	Simulated FCFS	DP Optimization	MILP Optimization	Original	DP Optimization	MILP Optimization
DFW	CPS = 0	155.67	30.05	37.07	36.02	29.90	39.94	38.96
	CPS = 1			31.02	30.96		34.20	34.13
	CPS = 2			29.60	29.75		32.73	32.77
	NO CPS			N/A	29.39		N/A	32.39
Even (25%)	CPS = 0	N/A	80.79	100.90	96.16	80.48	105.73	100.93
	CPS = 1			74.25	72.92		79.05	77.75
	CPS = 2			66.02	67.03		70.73	72.72
	NO CPS			N/A	64.85		N/A	69.73

Table 6: Delay per aircraft results averaged over the four days of data (in seconds)

Table 6 shows the delays per flight averaged over the four days of data: June 14th, 15th, 16th, and 18th. The first column in the table is the real-world comparison. This was computed by using the actual takeoff times that were detected at DFW on that day and subtracted our simulated runway ready times. The reason for using simulated runway ready times was because we wanted to

compare our results with real-world results that involved unimpeded taxi times. Had we used the real times that the aircraft arrived at the runway, there would be an amount of system delay that was incurred during taxiing that would be lost. Some of the takeoff times were before our simulated runway ready times. Although this is impossible in real life, it makes sense because the unimpeded times were taken from the 10^{th} percentile of taxi times and so 9% of flights would end up taxiing faster than we predicted. We included these negative delay times into the departure delay as it would make up for runway ready time predictions that might be earlier in real life. The real-world departure delay is so large because the controllers are not telling aircraft to take off as soon as they can while obeying separation times, and there are arrival crossings which we are not taking into account.

Columns 2-4 show the results for the deterministic world. This means that the unimpeded taxi time predictions were assumed to be correct and unchanging. The departure delays were again computed by subtracting the simulated runway ready times from the assigned or optimal takeoff times. The "Original" column is the departure delay associated with aircraft taking off as they are ready in a First Come First Serve sequence at the runway and enforcing minimum separation times. The difference between this departure delay and the real-world departure delay is the amount of extra time that the aircraft were waiting unnecessarily at DFW that day, some of which is due to arrival crossings. The next two columns show the DP and MILP optimization results for departure delay. The difference between these results and the original departure delay shows the benefit of re-sequencing aircraft at the spot.

The last three columns in each table are the departure delays for the stochastic world. The original column uses the original spot leave times and sequence along with the stochastic unimpeded taxi times and averages the departure delay of sending them FCFS at the runway over 500 iterations. The optimization columns use the optimal spot leave times found by the DP and MILP optimizations and calculates the departure delay again using FCFS at the runway. These are the columns that show the effects of taxi time uncertainties and whether it is beneficial to use these optimizations in real-world scenarios. If the optimization departure delays are significantly less than the original stochastic delays then it is still beneficial to use our optimizations in uncertain circumstances.

The departure delay is calculated for real, deterministic, and stochastic worlds while also changing six different factors. The first change is the aircraft distribution. Since DFW is on average about 88% large aircraft, there is less of a benefit of re-sequencing. To show how large of a benefit re-sequencing can have, we uniformly assigned the DFW aircraft new aircraft types. The even distribution of aircraft types includes close to 25% each of small, large, heavy, and B-757. Using DFW's distribution and an even distribution we also looked at the effect of CPS. No CPS means that there was no restriction on position shifting anywhere on the surface of the airport. This is expected to have better results as it is the most flexible. Though the departure delay is better without CPS, the controllers' workload is increased without it. Therefore, we looked at CPS equal to zero, one and two and determined how much delay this added compared to no CPS. A low CPS value is more realistic for real-world applications. Separate delay results from the four days of data can be found in Appendix D. The major difference in delays between each day comes from the difference in the amount of flights.

The change in runway sequence adds another dimension to the measure of the robustness of the optimizations. We measured sequence change three different ways. First, we looked at the sum of spaces that each flight shifted from the optimal takeoff sequence due to taxi time perturbations. Next we looked at relative sequence change and weighted relative sequence change. The relative sequence change was measured by the number of aircraft pairs where aircraft *i* was supposed to come before aircraft *j* in the optimal sequence, but instead *j* came before *i* in the final sequence. The weighted relative sequence change added in weights for how many positions these pairs were separated by. We also looked at how many aircraft arrived too soon after the aircraft before them which would violate separation requirements and how many seconds early the aircraft arrived. These results are shown in Appendix E for each day of data.

We ran the optimizations on Dell desktops running Linux that had four dual-core processors with 4GB of RAM. The MILP optimization took on average 45 seconds in total to run on a full day of data when the data was split into 15-minute bins. This time includes the time it takes to sort the data, divide it into bins, and save the results from each bin. YALMIP took around 15 seconds to set up the MILP formulation and around 1-8 seconds to solve it, depending on CPS. The DP optimization took under a second to build the graph and solve it for CPS equal to zero and one and about 30 seconds for CPS equal to two.

5.0 Discussions

This chapter will interpret our optimization results, focusing mainly on the effects of taxi time uncertainties. We first begin with how we measured the success of our project. We also point out strengths and limitations to both the methodology and the results. We interpret our results to show whether the deterministic optimizations would be beneficial in real-world applications. Finally, we conclude this section with suggestions for future work that can extend upon this project.

5.1 Measures of Success

Our goals for the MILP optimization changed as we progressed through the project. Our original goals were to be able to optimize departures and arrivals and look at the effects of uncertain taxi times on these optimizations. We also wanted to impose a position shifting constraint at the spot to maintain a sense of fairness among different airlines. This constraint posed more problems than we expected, and so we were unable to attempt arrival crossings in addition to departure takeoffs. In addition to this we realized that there is no clean way to impose Constrained Position Shifting (CPS) for the MILP optimization while also splitting the day of data into smaller time periods. Despite these setbacks we were still able to get departure results with MILP using a relaxed CPS constraint and look at the effects of uncertainties, which was our most imperative goal.

Our goals for the DP optimization were similar, and also changed due to the difficulty of optimizing departure takeoffs only. The major problem encountered was the difficulty in defining a recursive relationship for the optimal schedule for n aircraft and for that of $n+1$ aircraft. Such a relationship is vital for a DP algorithm to produce optimal results, and the lack of such a relationship showed in the results of the first attempted algorithm, ***Simple Spot***. We decided that our time would be better spent developing heuristics that approximated the optimal schedule rather than searching for an exact solution, so we developed several DP heuristics that came increasingly close to simulated FCFS in terms of departure delay. While optimizing at 1-CPS never produced results that improved on simulated FCFS, 2-CPS did for several of the heuristics, showing that DP could be used to solve for an entire day without binning in a

reasonable amount of time. From there, we were able to look at the effects of uncertainties as we did with the MILP.

5.2 Strengths and Limitations of Methods

Our weekly meetings with our advisors and supervisors provided us with a good structure to show progress and receive feedback. We were able to discuss ideas of possible ways to attack the problem and possible reasons for our optimizations not producing the results we expected. It was also beneficial for each of us to work on one optimization tool while communicating issues with the other partner. Testing on smaller data sets is always a wise idea, especially when they are solvable by hand which is useful for checking that the optimization came to the correct answer. These smaller data sets, however, were not always helpful for testing the MILP because most of the bugs came from binning the data into smaller time periods, which is unnecessary for small data sets. The smaller data sets occasionally revealed bugs in the DP heuristics, but most of the bugs came up while working with larger data sets.

Using real data from DFW and taxi time distributions modeled on real data was useful in that it showed how our methods would affect departure delay for an actual 24-hour schedule of departures. However, June 15th was missing a substantial amount of flights in the middle of the day, and the other three days were missing roughly 10% of their flights each. Such factors decrease the realism of our results. Additionally, DFW does not experience high demand, has a very homogeneous mix of aircraft (an average of roughly 90% of "large" aircraft over the four days of data), and space out their takeoffs for longer than seems necessary. These factors all made it more difficult to see the actual effects of our optimizations. Higher demand and a more heterogeneous mix of aircraft would have provided more opportunities for optimization, as shown in our even distribution results. The fact that the real-life takeoffs were spaced out farther than they needed to be means that just having them take off as soon as they safely can would have made vast improvements in delay, overshadowing the further improvements made by our optimizations.

We addressed the homogeneity of the aircraft types by randomly assigning aircraft types to the aircraft in the schedule for each day to create an even distribution. While this helped us see what extra opportunities heterogeneity gives, it was limited by the fact that we were unable to model

realistic taxi times because our taxi time model was based on DFW's aircraft mix. This is expanded upon in Section 5.4. We intended to address the lack of demand by simulating higher demand, but did not have enough time to do so.

5.3 Interpretation of Results

The DP and MILP algorithms were both suboptimal due to discrepancies between spot sequences and runway sequences, which will be explained in more depth later in this section. DP heuristics were used to optimize the sequence and MILP finds local optimizations for each bin. The MILP optimization also used a relaxed CPS constraint that alters the results slightly in a positive direction. This explains why they do not give the same exact optimization delays, but the results are fairly close between the two and the optimizations can still be compared. The timing results proved that the optimizations are operationally feasible in real-time situations. For a full day of flights, DP becomes operationally infeasible when CPS is set to three positions or higher. MILP can run without CPS, but it gives better results when there are fewer bins, although it becomes operationally infeasible with too large of bins. The size of bins that the optimization can handle was determined to be largely dependent on the data. The data from the 14th has spot leave times that are very close together, which creates more options for the optimal sequence.

We came across several unexpected results, and the cause of these was differing sequences at the spot and at the runway. We set our baseline to be First Come First Serve at the runway with enforced minimum separation requirements, because this would be the easiest improvement in practice. DFW already uses FCFS at the runway, but they do not enforce that a flight takes off as soon as it safely can. We expected any of our optimizations to be able to do better than simulated FCFS and so we were surprised when they did not. We quickly realized that this was because simulated FCFS is a CPS of zero at the runway, but we were imposing CPS at the spot. We attempted to achieve simulated FCFS departure delay with our MILP optimization using CPS equal to zero at the spot, thinking that these should give the same answers, but were unable to.

To investigate why a FCFS at the spot (CPS=0) was giving worse results than runway FCFS we took the runway FCFS takeoff times and subtracted the unimpeded taxi times in order to get spot leave times, as if simulated FCFS was a metering optimization result. What this showed us was in order to achieve simulated FCFS at the runway, the spot sequence was changing up to three

positions. This would be illegal in the CPS=0 optimization, and so runway FCFS delay may not be achieved or surpassed unless the CPS at the spot allows for as much flexibility as the runway FCFS requires. An example of how FCFS at the runway can shift the spot sequence is shown in Tables 7 and 8 below. In this example the optimization with CPS equal to zero at the spot gives an additional delay of 16 seconds compared to FCFS at the runway in order to preserve the sequence at the spot. This situation happens because flights 2 and 3 have very close spot leave times but flights 1 and 2 have very close runway ready times. When FCFS at the runway puts the minimum separation time between flights 1 and 2, flight 2 gets pushed back and its optimal spot leave time ends up being behind that of flight 3 which shifts the spot sequence.

Given data						
Aircraft #	Aircraft type	Spot ready time(s)	Spot sequence	Taxi times(s)	Runway ready time(s)	Runway sequence
1	2	4000	1	375	4375	1
2	2	4150	2	250	4400	2
3	2	4170	3	350	4520	3

Table 7: Initial data for the problem

	Results for FCFS at runway			Results for FCFS at spot (CPS=0)		
Aircraft #	FCFS at runway takeoff time(s)	Proposed spot leave time (s)	Spot leave sequence	CPS=0 at spot takeoff time (s)	Proposed spot leave time (s)	Spot leave sequence
1	4375	4000	1	4375	4000	1
2	4435	4185	3	4435	4185	2
3	4520	4170	2	4536	4186	3

Table 8: Comparison of the results given by FCFS at the runway and FCFS at the spot (CPS=0)

Another unexpected result came from not being able to enforce CPS with MILP. The reason for this, as stated before, was due to binning. The problem with binning is that the optimization is unaware of any flights that are not in the bin that is currently being solved for. Due to different taxi times between flights, the runway and spot sequences can vary greatly. In order to properly enforce CPS at the spot and separation requirements at the runway it is necessary to have all

flights that are next to each other in the spot sequence along with all flights that are next to each other in the runway sequence in the same bin. We began with separating the bins by runway ready times. This ensured that the runway sequence was preserved but it did not guarantee that the spot sequence was preserved. To compensate for this we attempted to add in flights whose spot leave time was among the times that were currently in the bin. These flights however, would have runway ready times that were later than the rest of the bin's runway ready times, possibly later than those in the next bin, and therefore could cause the runway sequence to be lost. If we continued this process of adding in runway ready times that were within the bin boundaries and then adding in spot ready times that were within the bin boundaries we could eventually come close to having the whole day of flights in the current bin and the solver would be unable to optimize it. An example of this overlap in bins is shown in Table 9.

Bin 1		Bin 2	
Spot ready time(s)	Runway ready time(s)	Spot ready time(s)	Runway ready time(s)
5000	5250	5200	5401
4900	5300	5125	5405
5155	5310		

Table 9: Spot and runway ready times overlap for flights in two bins

In this example Bin 1 has an original runway ready time boundary of 5400 seconds. The runway ready times of Bin 2 are all higher than the runway ready times of Bin 1, but the second spot ready time in Bin 2 is less than one of the spot ready times in Bin 1. While the optimization is solving for Bin 1, it would be unaware that the third flight is actually fifth in the spot sequence out of all six flights. To fix this we can put the second flight of Bin 2 into the first bin and this will correct the spot sequence. The other effect that moving this flight to the first bin will have is that the runway sequence will now be lost because the first flight in the second bin is ready at the runway before the second flight in this bin. Therefore both flights must be moved to the first bin in order to fully maintain both the runway and spot sequence. With real data this could continue until the bins were too large to handle. The data from DFW only ran into this problem a few times during a day, and so we let the flights break CPS instead of making the bins larger. We determined that this would have a smaller effect on the results and operational feasibility than adjusting the bins.

The main goal of this project was not just to optimize using MILP and DP but to see what happens to these optimizations when uncertainties are added into the model. The optimization delays under uncertainties were worse than the delays associated with not optimizing under uncertainties for both MILP and DP, using simulated FCFS as our baseline. The reason for this is that the goal of the optimizations is to find optimal spot leave times based on optimal takeoff times and to hold the aircraft at the spot until their optimal spot leave times. In a deterministic world the aircraft would then taxi unimpeded and be able to take off immediately. With stochastic unimpeded times, the aircraft no longer arrived at the runway when they were supposed to or even in the order that they were supposed to. This created delay at the runway which, when added to that at the spot, was more than that experienced at the runway by the simulated FCFS sequence with stochastic taxi times. This result shows that a deterministic optimization is not beneficial in a stochastic world. Contrary to these results, an even distribution of aircraft types allows for more benefits to be gained from re-sequencing and was robust under uncertainties.

From the results of the sequence analysis, it appears that the order of the spot releases has little bearing on how much the sequences are changed by, as measured by three different sequence change metrics. Stochastic taxi times rarely ever caused the sequence to change, and only by very little when it did. Additional delay incurred at the runway was mostly due to aircraft simply arriving at the runway too early with respect to the takeoff before it, even while still staying in the same position as in the originally projected takeoff sequence. The average change in unimpeded times for each of the days was about 8 seconds, which explains why the sequences were for the most part preserved. Also each aircraft that violated separation times did so by 10 seconds on average.

Date	Minimum	Average	Maximum
June 14th	-40	8.16	50
June 15th	-103	8.52	102
June 16th	-45	8.32	46
June 18th	-38	7.88	46

Table 10: Changes to the taxi times due to uncertainties for the four days of data (in seconds)

5.4 Strengths and Limitations of Results

As described above, while our results are based on real data, there are limitations to that data that make them a little less realistic and that give results that may not be as good as they could be. Additionally, we were unable to also optimize arrival crossings. This would have allowed us to better compare our delays to the delays that are actually seen at DFW. Although we discovered that the deterministic optimization was robust for the even distribution of aircraft, we are not sure whether or not this would be true in practice. The taxi time model that we used is heavily based on the spot group of the aircraft and the aircraft type. Since DFW has so few aircraft that are small, heavy, or are B757s there was not enough data to get an accurate prediction for their unimpeded taxi times. In addition it is unknown how DFW would assign an even aircraft mix to different spots, and so we kept the spot assignments as they were. In order to get a reliable result from an even distribution, we would have to look into an airport that actually uses an even distribution, for example Boston Logan, and do the same analysis that we did on DFW there.

We found another limitation to our results when we were analyzing the change in unimpeded taxi times due to uncertainties. All four days had a consistent average change of 8 seconds per aircraft; however the maximum and minimum changes were very different for one of the days. The 15th of June had maximum and minimum changes of 102 and -103 seconds respectively, whereas the other days had maximums and minimums closer to plus and minus 45 seconds. We looked into this and there did not seem to be any errors in the code or the data, but we also could not come up with a reason for why this day would have larger amounts of taxi time changes. These larger changes also caused the sequence to change for three or four flights, whereas the other days did not experience any sequence changes due to uncertainties. These large changes were consistently assigned to the same four flights on the 15th, but as stated before we were unable to figure out why.

5.5 Future Work

The next step to our project would be to add in arrival crossings to the optimization and again see the effects of uncertainties since this was our original plan. The first step to do this for the MILP algorithm would be to make a function that can determine whether an aircraft is a departure or arrival aircraft. An arriving aircraft's ready to cross time is analogous to a departure’s runway

ready time. Arrivals are also assigned a number from 1 to 6, which represents their crossing queue, and is analogous to departure weight classes. These characteristics determine separation time requirements at the runway. After an aircraft takes off, arrivals must wait based on how close their crossing queue is to where the departure took off from, i.e. the end of the runway. Crossing 6 is the closest to where departures take off and so aircraft crossing in queue 6 must wait 60 seconds. Two arrivals can cross the runway in different queues almost simultaneously, and so they are given a separation time of two seconds. If they cross in the same queue then the second aircraft must wait 10 seconds for the one in front of it to finish crossing. Similarly a departing aircraft must wait 12 seconds for an arriving aircraft to finish crossing. Since some of these separation times are quite smaller than the departure separation times, it is necessary to keep track of the last departure flight to make sure that separation times between departures are obeyed even if there are arrival crossings in between. The CPS constraint would not constrain arrivals in any way. For DP, all the above constraints would also apply. However, we were unable to determine how to fit arrival crossings into our DP heuristics and leave that to future researchers.

Once arrivals are added into the optimization, effects of uncertainty can be measured in the same way as the departures only optimization. System delay would be computed as the sum of the differences between when flights were ready at the runway and when they were assigned to either takeoff or cross.

Another operational constraint to be considered is priority departures, or those that must leave by a certain time. This could be the case if a certain aircraft needs to get to its destination airport by a certain time to keep its airline's schedule intact and prevent delay at other airports. This added constraint would further limit the extent to which a given spot release sequence could be optimized.

In addition to optimizing at the spot, a second optimization at the runway could be executed if enough aircraft build up there due to delay at the runway caused by taxi time uncertainties. This could be useful for our algorithms. Most likely, a tradeoff would have to be developed between how much a second optimization would save in delay and how much extra work a second optimization would make for air traffic controllers.

Next, stochastic variables other than unimpeded taxi times could be considered. For example, how closely separation times are adhered to at the runway. As mentioned previously, DFW does not tend to have its departures to take off as soon as they safely can, and it could be the case that other airports allow them to take off sooner than they are supposed to. Additionally, the accuracy of spot ready times would be another stochastic variable to consider, as that could have substantial effects on deterministic optimizations.

Alternative runway layouts could also be considered, where arrivals cross the runway differently than they do for the runway that we studied in this project. For example, a peripheral taxiway from the arrival runway to the terminal would eliminate the need for some arrivals to cross the departure runway at all. An optimal tradeoff between the delay saved and the increase in fuel burn resulting from sending some aircraft along the longer peripheral taxiway rather than straight across the departure runway could be beneficial.

Finally, if all of the stochastic variables make the use of deterministic optimizations in real-world situations unreasonable, then stochastic optimizations would be the next logical extension. This type of optimization would take into account uncertainties and perhaps give an optimal takeoff time window, so that it held up better to taxi time and other uncertainties.

References

Balakrishnan, H. and Chandran, B., "Efficient and Equitable Departure Scheduling in Real-Time: New Approaches to Old Problems," *USA/Europe Air Traffic Management R&D Seminar*, Barcelona, Spain, June 2007.

Balakrishnan, H. and Chandran, B., "Algorithms for Scheduling Runway Operations under Constrained Position Shifting," *Operations Research*, Vol. 58, No. 6, November-December 2010.

Ball, M., Barnhart, C., Dresner, M. et al. The National Center of Excellence for Aviation Operations Research (NEXTOR), (2010). *Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the united states* Retrieved from http://www.nextor.org/pubs/TDI_Report_Final_11_03_10.pdf

Chen, D., Batson, R. and Dang, Y., *Applied Integer Programming*, John Wiley & Sons, Inc., 2010.

Dantzig, G. The generalized simplex method for minimizing a linear form under linear inequality restraints. Pacific J. Math. Volume 5, Number 2, pp. 183-195. 1955.

Dasgupta, S., Papadimitriou, C. and Vazirani, U., *Algorithms*, McGraw-Hill, 2006.

Ishutkina, Mariya (2011, September 15th). Personal interview.

Jordan, Richard (2011, September 23rd). Personal interview.

Griffin, K. *et. al*. "Evaluating Surface Optimization Techniques Using a Fast-time Airport Surface Simulation", 2010.

Gupta, G., Malik, W. and Jung, Y.C., "A Mixed Integer Linear Program for Airport Departure Scheduling," *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO).* AIAA, Hilton Head, South Carolina, pp. 1-13, 2009.

Gurobi: An optimizer for MILP, MIQP, LP and QP . Robert Bixby, 2011. http://www.gurobi.com/.

FAA. "IFR Operations in the National Airspace System", 2008. http://www.docstoc.com/docs/837186/IFR-Operations-in-the-National-Airspace-System. Accessed 4 Oct 2011.

Hardaway, R. M. (1991). *Airport regulation, law, and public policy: the management and growth of infrastructure*. (pp. 52-55). Westport, CT: Quorum Books. Retrieved from http://books.google.com/

Howe, P. (2006, November 19). The 30-year saga of 14/32. *The Boston Globe*. Retrieved from http://www.boston.com/business/globe/articles/2006/11/19/the_30_year_saga_of_1432/

Ishutkina, M.A. "Analysis of the interaction between air transportation and economic activity: a worldwide perspective." PhD thesis, Massachusetts Institute of Technology, 2009. Accessed 19 Sep 2011. http://hdl.handle.net/1721.1/49882

Lee, H., Tradeoff evaluation of scheduling algorithms for terminal-area air traffic control. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2008. Accessed 30 Aug 2011. http://hdl.handle.net/1721.1/45254

Malik, W., Gupta, G. and Jung, Y.C., "Managing Departure Aircraft Release for Efficient Airport Surface Operations," *American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control (GNC) Conference and Modeling and Simulation Technologies (MST) Conference*, Toronto, Canada, pp. 1-15, August 2010.

Marchi, R. "Reframing the Runway: A case study of the impact of community organizing on news and politics," JOURNALISM: THEORY, PRACTICE & CRITICISM, vol. 6(4):465-485, Sage, November 2005.

McCall, E., "Performance results of the simplex algorithm for a set of real-world linear programming models", March 3, 1982. Communications of the ACM, 25, 207-212.

NASA Aviation Systems Division. "SARDA Spot and Runway Departure Advisor", 2011. http://www.aviationsystemsdivision.arc.nasa.gov/research/surface/sarda.shtml. Accessed 4 Oct 2011.

Paramount Business Jets. "National Airspace System". 2005-2011. Accessed September 2, 2011. http://www.paramountbusinessjets.com/charterterms/national_airspace_system.php.

Simaiakis, I., Khadilkar, H., Balakrishnan, H., Reynolds, T.G., Hansman, R.J, Reilly, B. and Urlass, S., "Demonstration of Reduced Airport Congestion through Pushback Rate Control," *Proceedings of the USA/Europe Air Traffic Management R&D Seminar*, MIT Technical Report, ICAT-2011-2, June 2011.

"Terminal Area Forecast Summary, Fiscal Years 2010-2030." Federal Aviation Administration. 2010. Accessed 31 Aug 2011. http://www.faa.gov/about/office_org/headquarters_offices/apl/aviation_forecasts/taf_reports/media/TAF%20Summary%20Report%20FY%202010%20-%202030.pdf

U.S. Department of Transportation, (2008). *Notice of amendment to policy statement regarding airport rates and charges* (FAA-2008-0036-0100) Retrieved from http://www.regulations.gov/

U.S. General Accounting Office, Accounting and Information Management Division. (1995). *Denver international airport: information on selected financial issues* (AIMD-95-230) Retrieved from http://www.gao.gov/products/AIMD-95-230

"U.S. rules against Boston airport fee plan favoring big planes." (1988, December 23). *Los Angeles Times*. Retrieved from http://articles.latimes.com/1988-12-23/news/mn-548_1_fee-structure

Venkatakrishnan, C.S. "Analysis and Optimization of Terminal Area Air Traffic Control Operations." PhD thesis, Massachusetts Institute of Technology, 1991. Accessed 1 Sep 2011. http://hdl.handle.net/1721.1/13716

YALMIP : A Toolbox for Modeling and Optimization in MATLAB. J. Löfberg. In Proceedings of the CACSD Conference, Taipei, Taiwan, 2004. http://users.isy.liu.se/johanl/yalmip/.

Appendix A- Acronyms

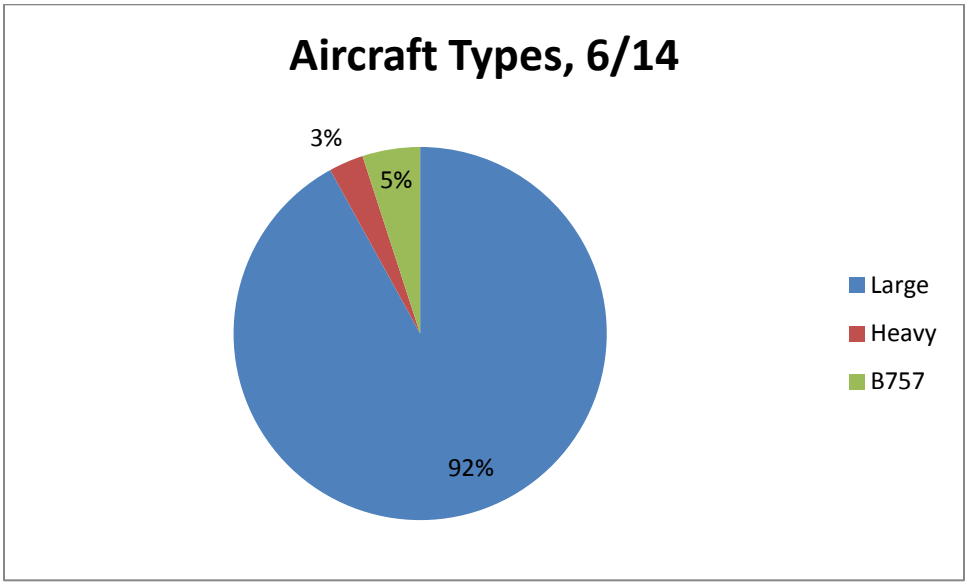
ATC- Air Traffic Control
B&B- Branch and Bound
B&C- Branch and Cut
CDT- Central Daylight Time
CPS- Constrained Position Shifting
DFW- Dallas/ Fort Worth International Airport
DP- Dynamic Programming
DSP- Departure Sequencing Problem
DST- Decision Support Tool
FAA- Federal Aviation Administration
FCFS- First Come First Serve
GC- Ground Controller
GDP- Gross Domestic Product
GMT- Greenwich Mean Time
IS-IS- Intermediate System to Intermediate System
LC- Local Controller
LP- Linear Programming
MILP- Mixed Integer Linear Programming
NAS- National Airspace System
NASA- National Aeronautics and Space Administration
NEXTOR- The National Center of Excellence for Aviation Operations
OR- Operations Research
OSPF- Open Shortest Path First
PACE- Program for Airfield Capacity Efficiency
RC- Ramp Controller
TSP- Traveling Salesman Problem

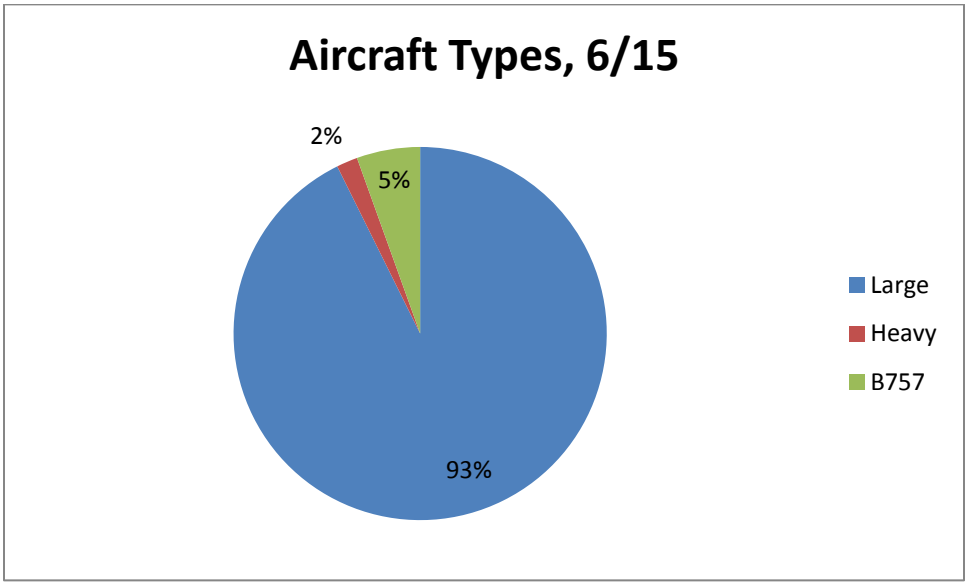
Appendix B- Glossary

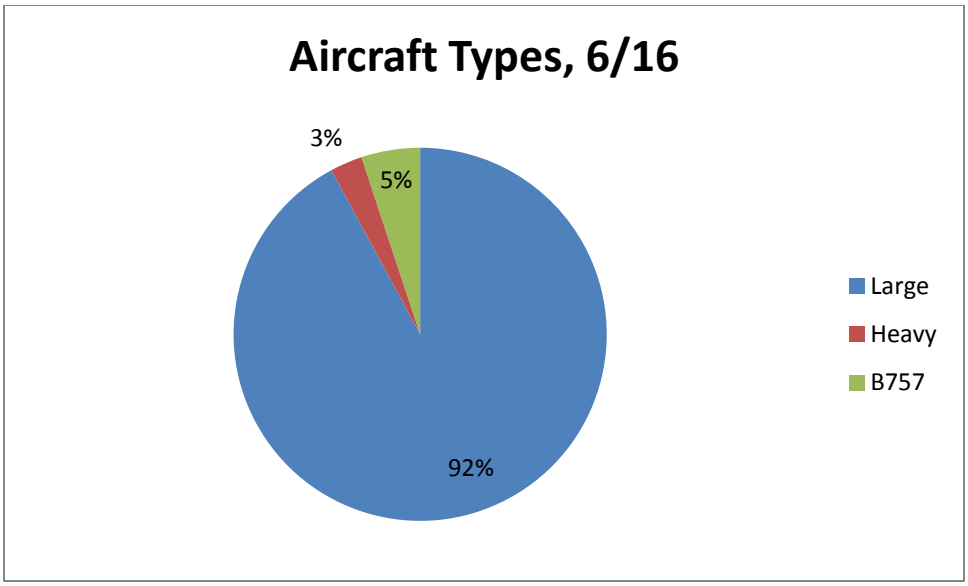
Constrained Position Shifting	Allowing an aircraft to shift by *k* or less spaces from its original place in the sequence
Directed acyclic graph	A graph whose nodes are connected by directed edges and which does not contain a cycle
Dynamic Programming	A method of solving problems by breaking them into simpler sub-problems whose solutions depend on the previous solutions
Efficiency	Achieving the capacity of the airport by minimizing system delay
Equity	Maintain fairness between airlines by constraining how many positions aircraft may shift from their original positions
Ground Controller	The ATC role responsible for managing aircraft on the ground, mainly in the taxiway, to ensure safety
Local Controller	The ATC role responsible for managing the runway, directing aircraft when to take off and when to cross the runway
Makespan	The time at which the last aircraft took off
Maximum delay	The most delay that any one aircraft incurred on a runway within a fixed period of time
Metering	A decision support tool that holds aircraft at the gate or spot so that they do not experience congestion or delay while taxiing or at the runway
Mixed Integer Linear Programming	A method of planning activities or solving for variables, at least one of which is an integer, by minimizing or maximizing an objective function while obeying certain constraints
National Airspace System	Controls all matters related to airspace above the United States
NP-hard	Problems for which it is difficult to find polynomial-time solutions
Objective function	Function which a program seeks to minimize or maximize
Operationally feasible	Solvable in a reasonable amount of time
Polytope	N-dimension geometric object with flat sides, for example a soccer ball
Queue	Area where aircraft wait in a line before they can take off or cross the runway
Ramp Controller	The ATC role responsible for managing the ramp, telling aircraft when to push back from the gate and taxi to a spot
Re-sequencing	A decision support tool which changes the order of when aircraft leave the spot, take off, or cross the runway in order to minimize delay due to separation times

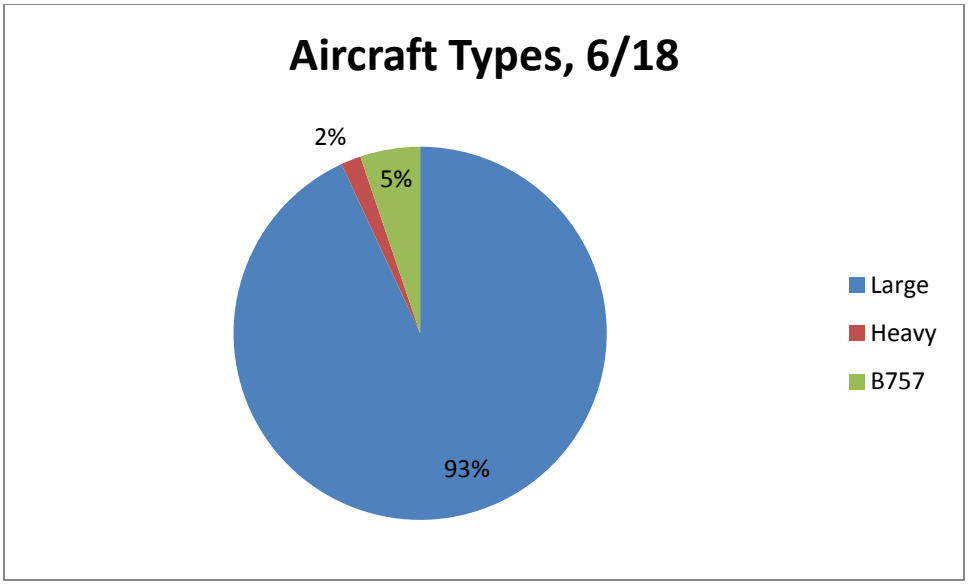
Separation times	Times between aircraft takeoffs that are mandated based on FAA's separation distance requirements, which are dependent on aircraft type, to ensure safety
Sequence Reliability	The sum of the probabilities between all pairs of adjacent aircraft in a given sequence that the aircraft in such pairs will violate the minimum separation time required between the two of them
Sequence Robustness	How well a sequence of aircraft responds to uncertainty in terms of delay
Sequence Weakness	The probability that a given sequence of aircraft will result in a minimum separation time violation
System/departure delay	The sum of the delays that each aircraft experience while on the ground, calculated by when they took action minus when they were ready to take action
Throughput	The number of aircraft that takeoff during a certain time period, which is a measure of an airport's capacity
Tractable	Solvable in polynomial time

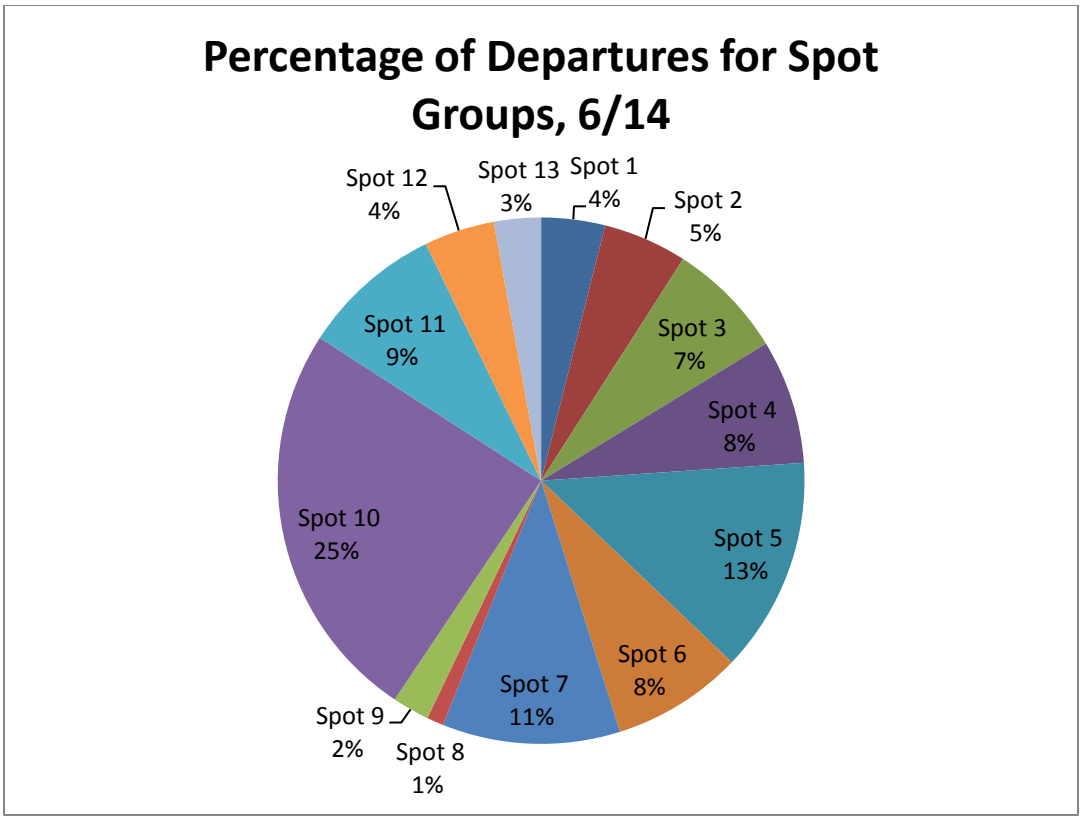
Appendix C- Charts for Data Analysis

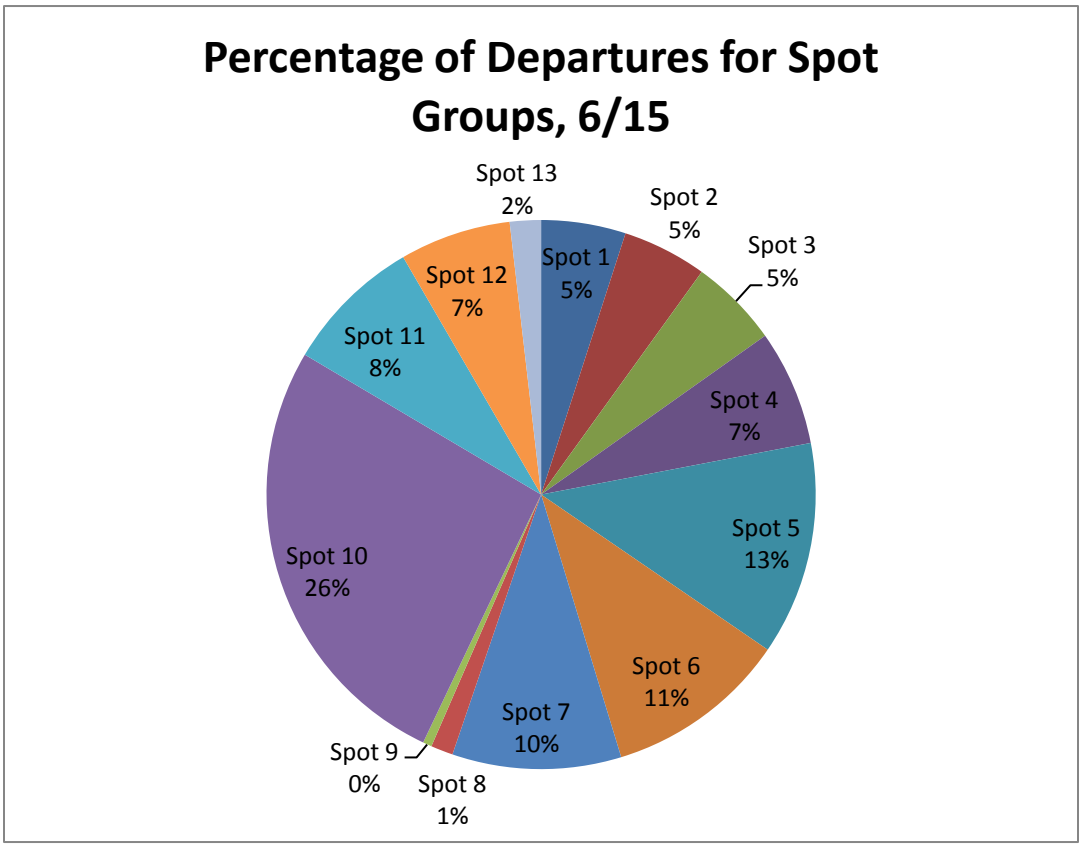


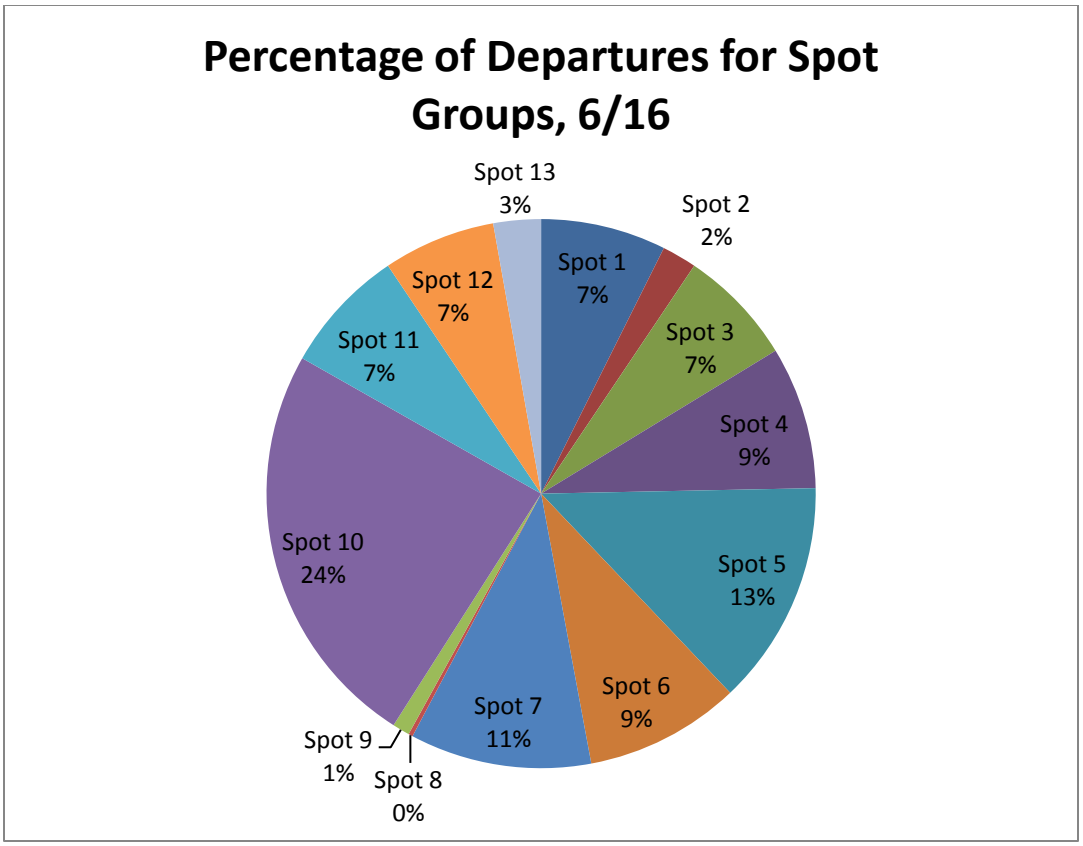


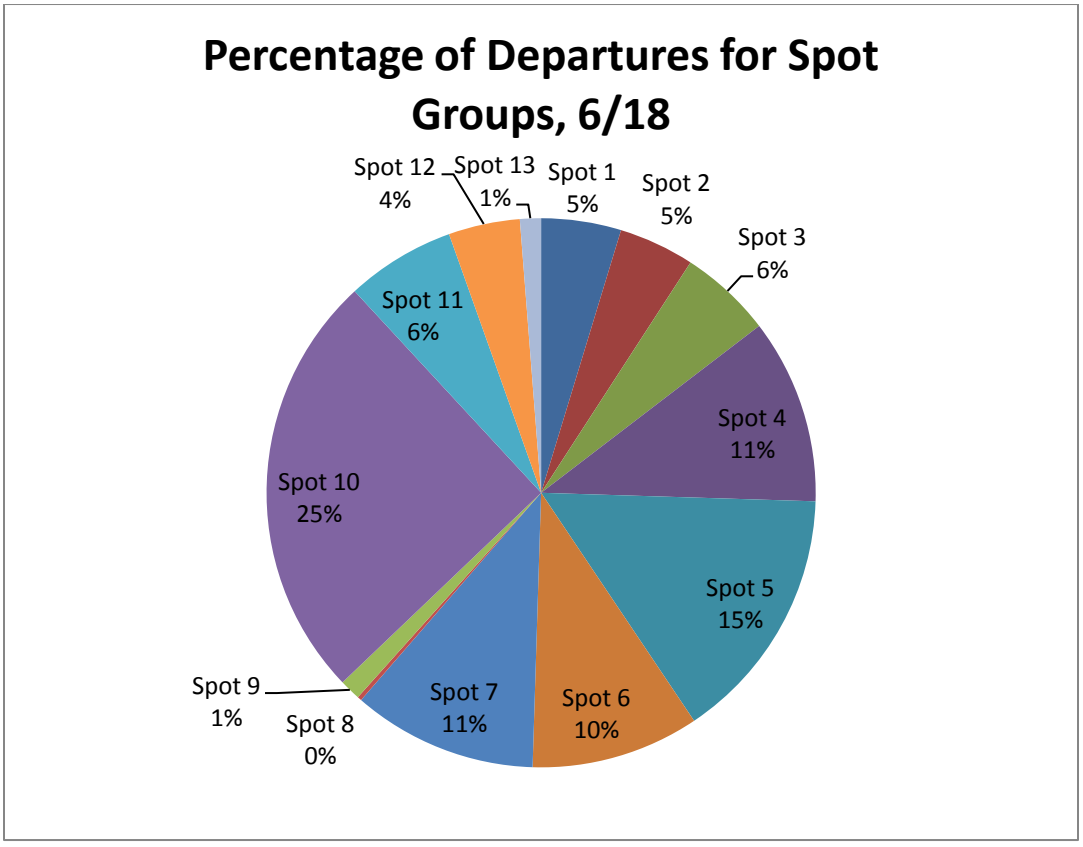

Aircraft Types, 6/16
3%
5%
92%
Large
Heavy
B757

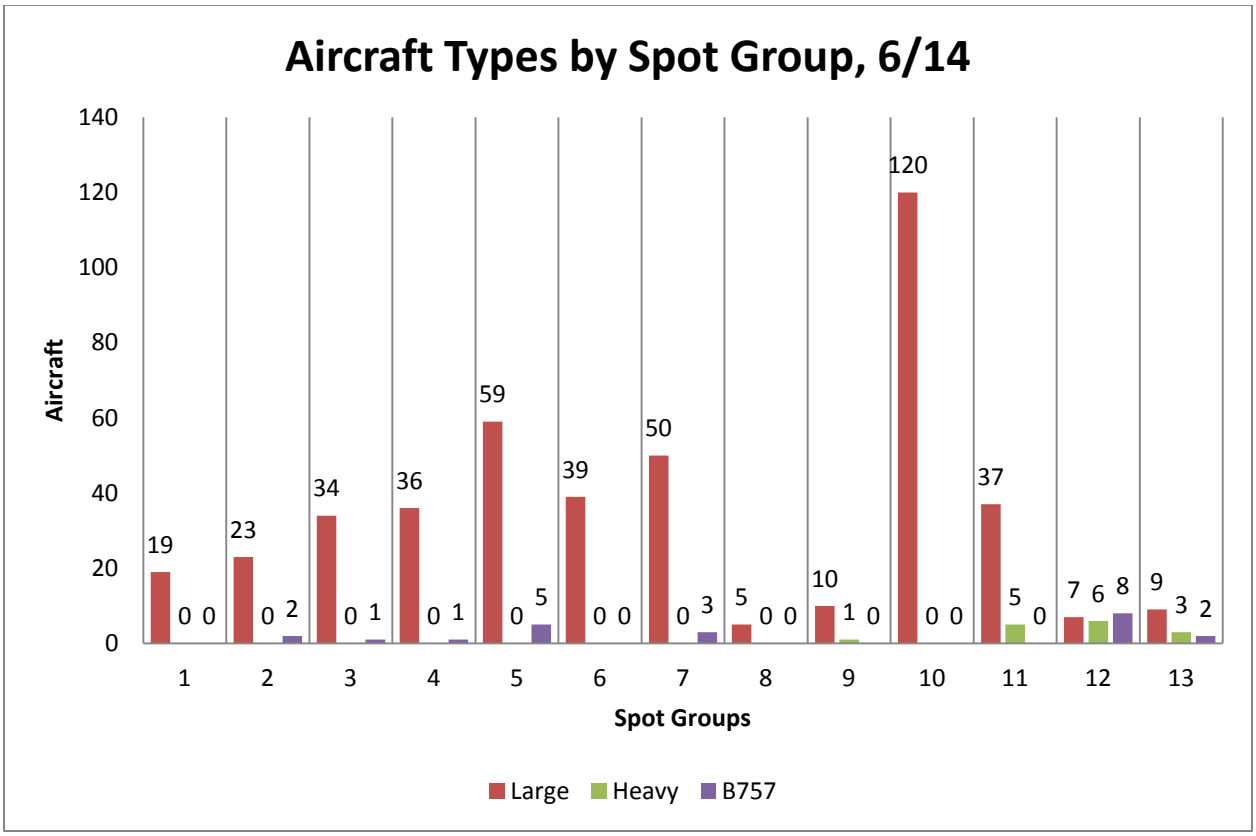

Aircraft Types, 6/18
2%
5%
93%
Large
Heavy
B757

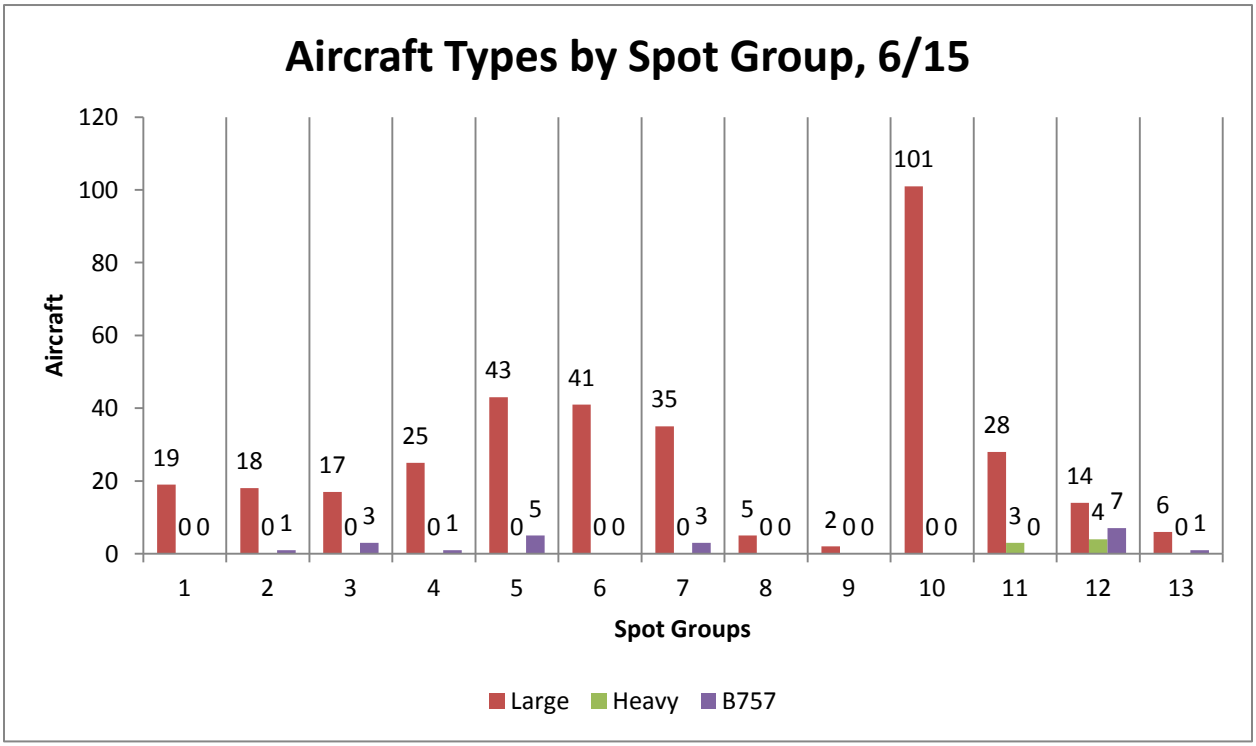

Percentage of Departures for Spot Groups, 6/14
Spot 12 4%
Spot 13 3%
Spot 1 4%
Spot 2 5%
Spot 3 7%
Spot 4 8%
Spot 5 13%
Spot 6 8%
Spot 7 11%
Spot 8 1%
Spot 9 2%
Spot 10 25%
Spot 11 9%

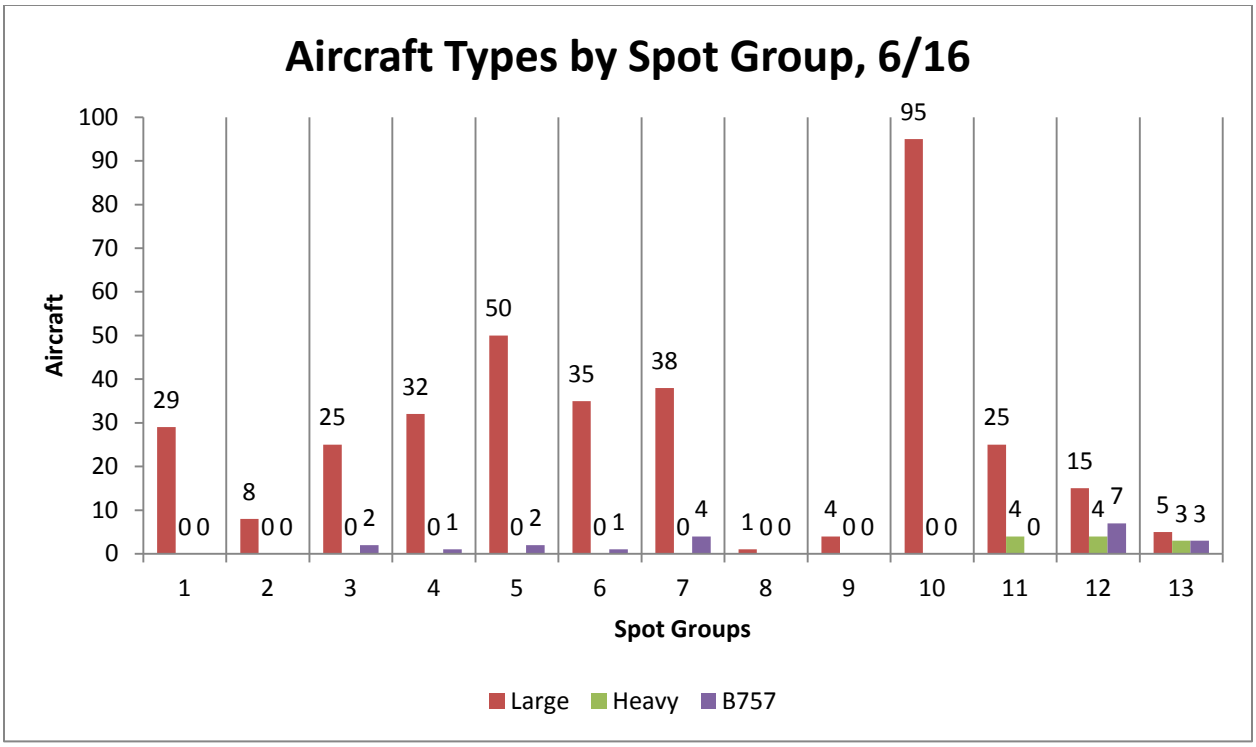

Percentage of Departures for Spot Groups, 6/15
Spot 13 2%
Spot 2 5%
Spot 3 5%
Spot 1 5%
Spot 12 7%
Spot 11 8%
Spot 4 7%
Spot 5 13%
Spot 10 26%
Spot 6 11%
Spot 7 10%
Spot 9 0%
Spot 8 1%

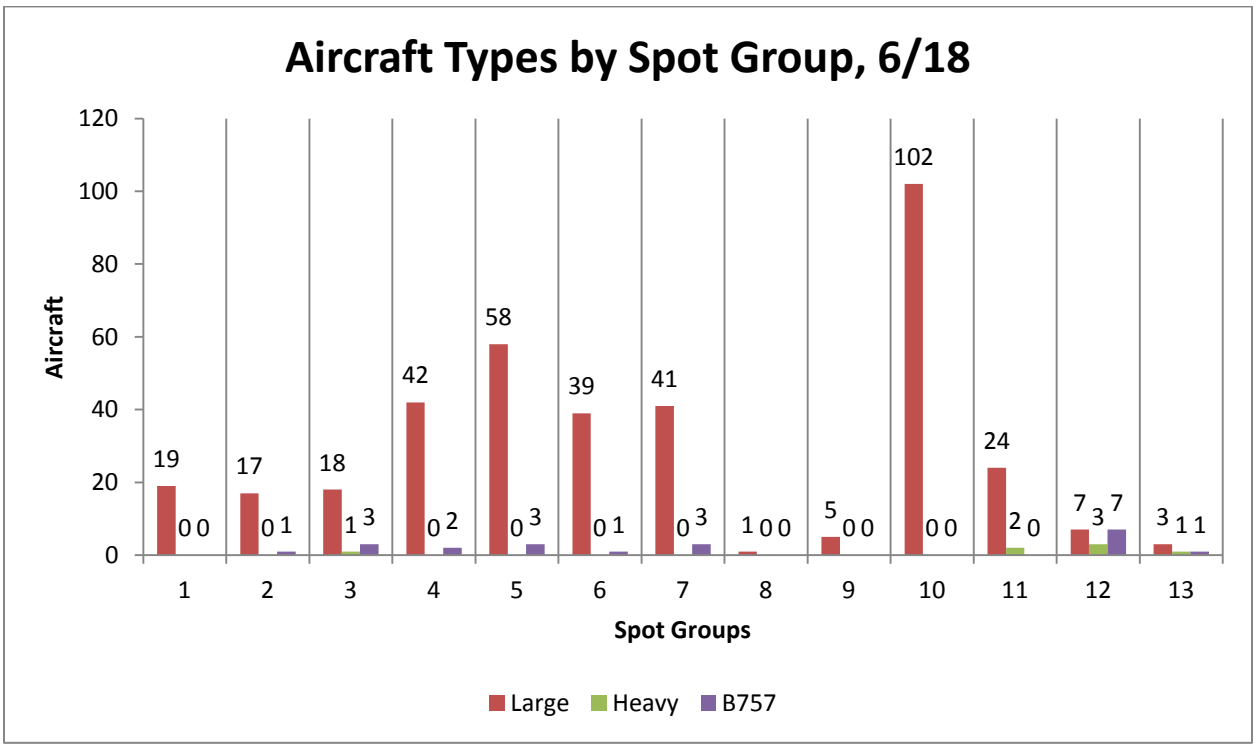

Percentage of Departures for Spot Groups, 6/16
Spot 13 3%
Spot 2 2%
Spot 1 7%
Spot 12 7%
Spot 3 7%
Spot 11 7%
Spot 4 9%
Spot 10 24%
Spot 5 13%
Spot 6 9%
Spot 7 11%
Spot 9 1%
Spot 8 0%

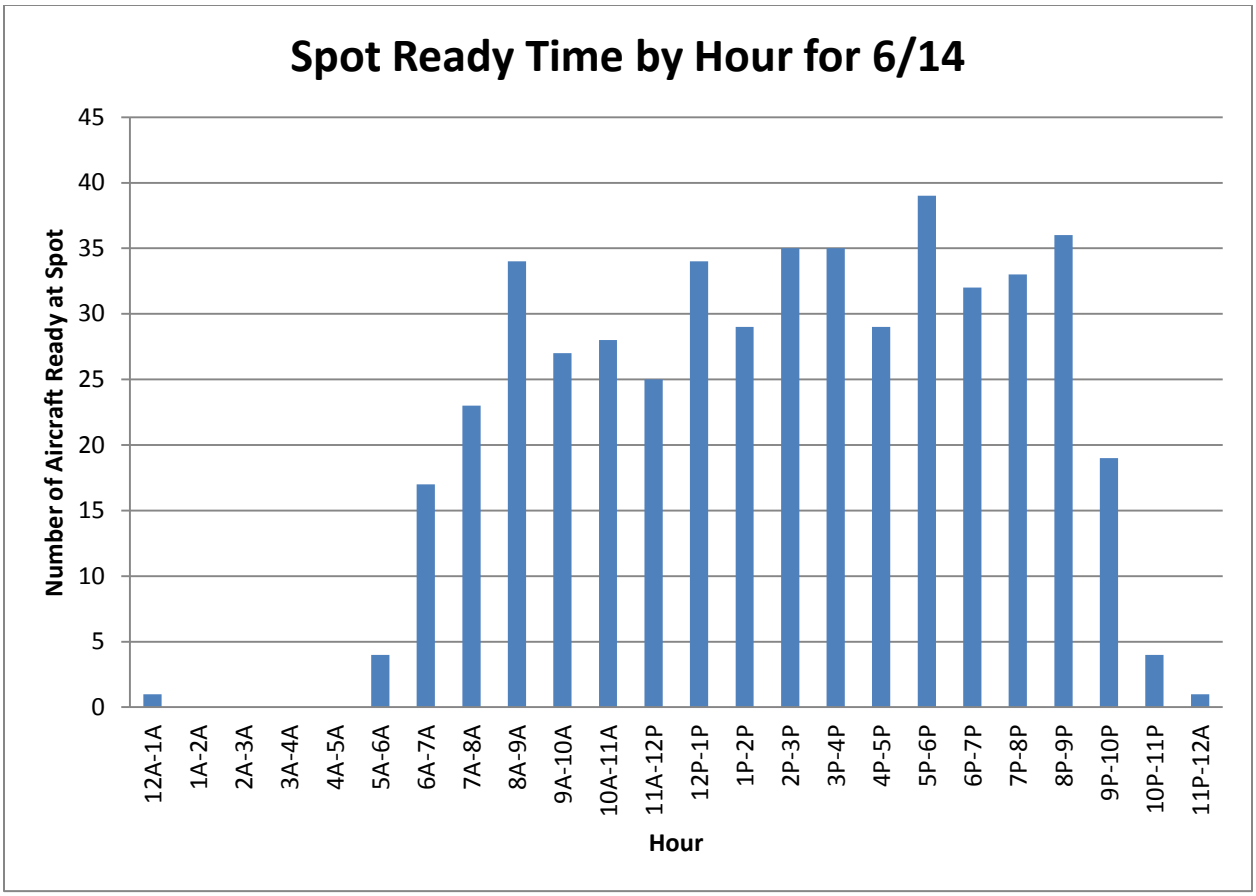

Percentage of Departures for Spot Groups, 6/18
Spot 12 4%
Spot 13 1%
Spot 1 5%
Spot 2 5%
Spot 3 6%
Spot 11 6%
Spot 4 11%
Spot 10 25%
Spot 5 15%
Spot 9 1%
Spot 8 0%
Spot 7 11%
Spot 6 10%

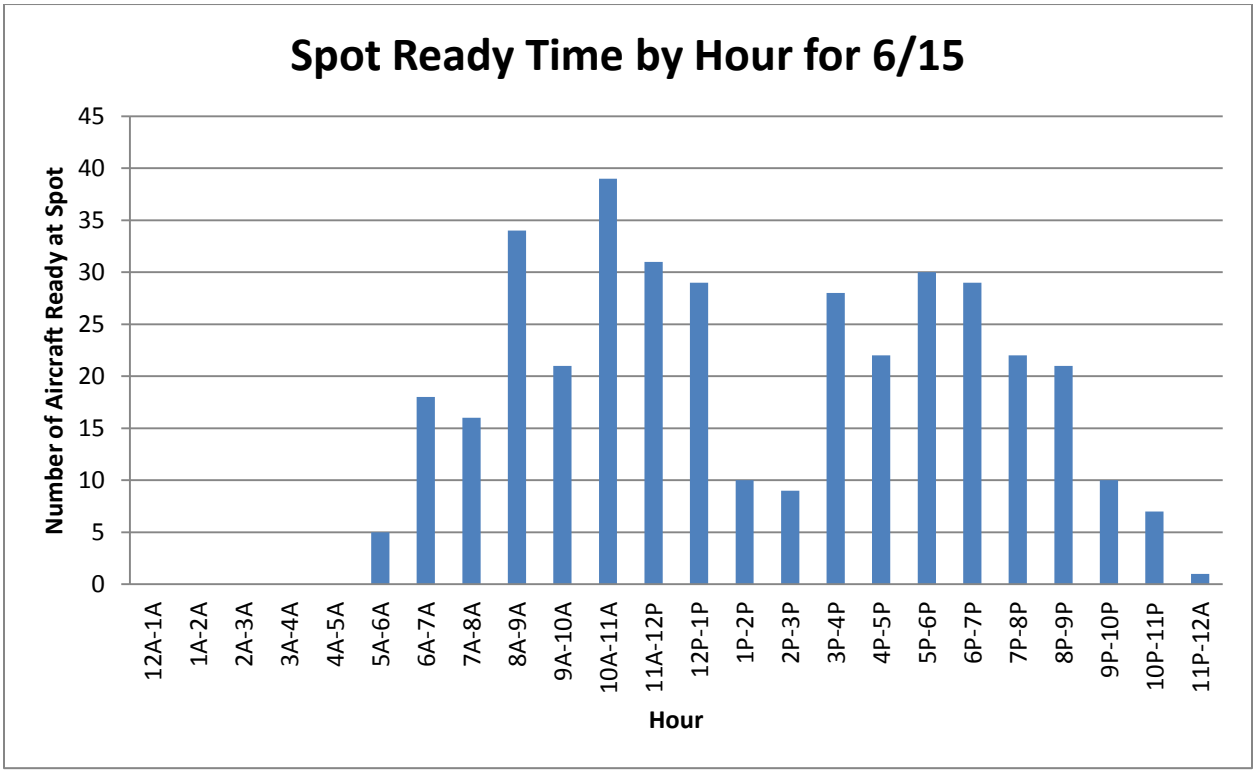

Aircraft Types by Spot Group, 6/14
Aircraft
Spot Groups
Large
Heavy
B757
19 0 0
23 0 2
34 0 1
36 0 1
59 0 5
39 0 0
50 0 3
5 0 0
10 1 0
120 0 0
37 5 0
7 6 8
9 3 2

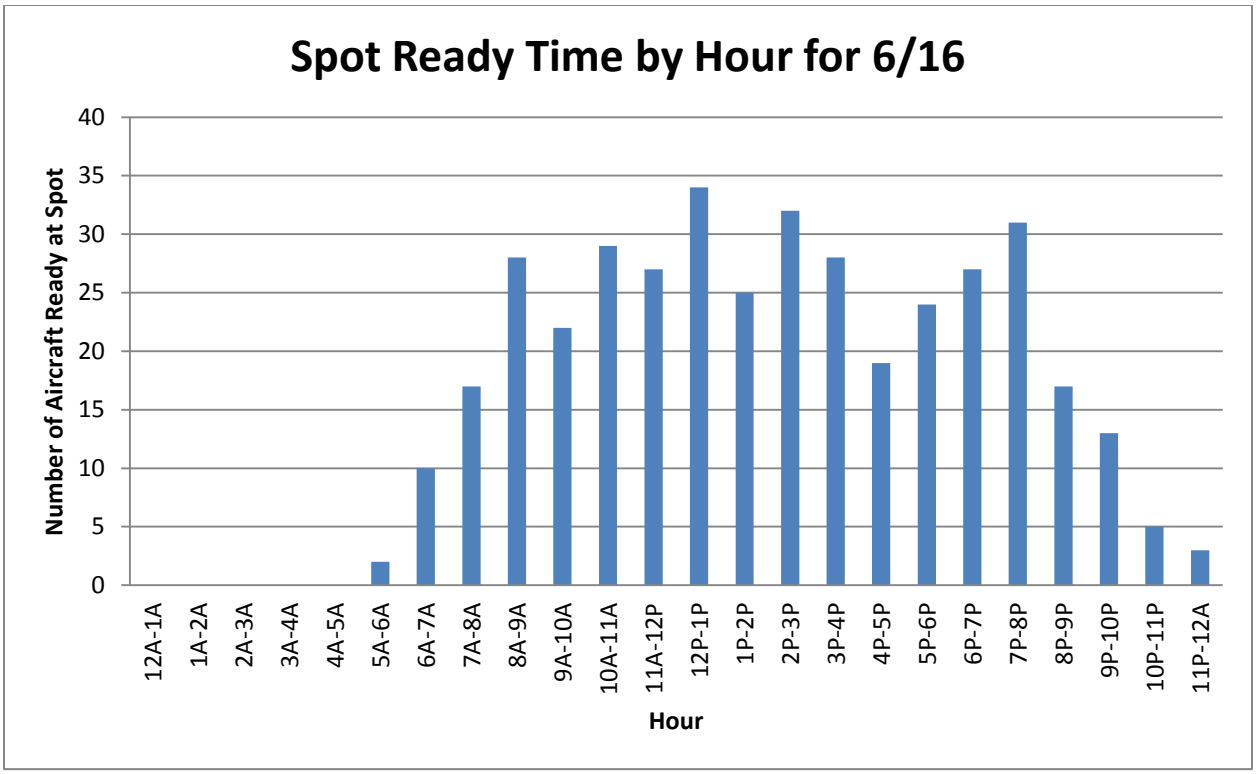

Aircraft Types by Spot Group, 6/15
Aircraft
Spot Groups
Large
Heavy
B757
19 0 0
18 0 1
17 0 3
25 0 1
43 0 5
41 0 0
35 0 3
5 0 0
2 0 0
101 0 0
28 3 0
14 4 7
6 0 1

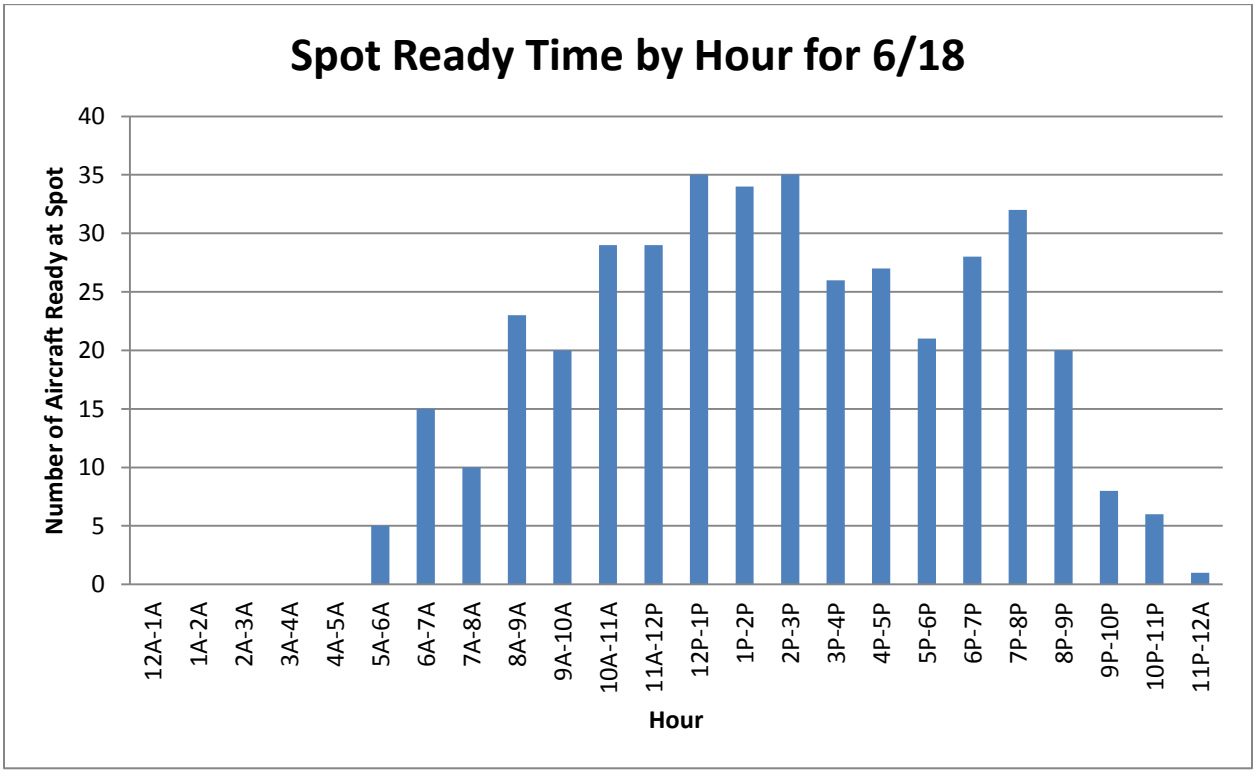

Aircraft Types by Spot Group, 6/16
Aircraft
100
90
80
70
60
50
40
30
20
10
0
29 0 0
8 0 0
25 0 2
32 0 1
50 0 2
35 0 1
38 0 4
1 0 0
4 0 0
95 0 0
25 4 0
15 4 7
5 3 3
1 2 3 4 5 6 7 8 9 10 11 12 13
Spot Groups
Large
Heavy
B757

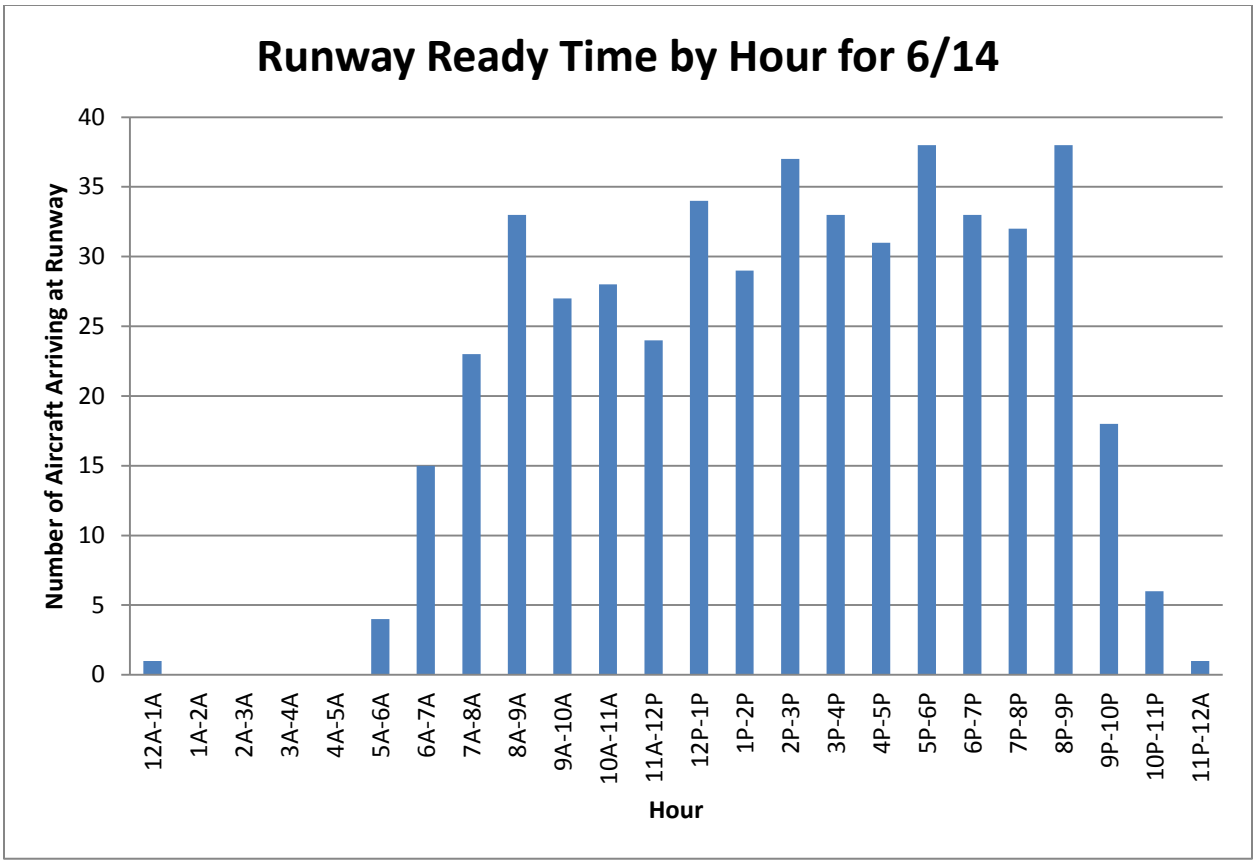

Aircraft Types by Spot Group, 6/18
Aircraft
120
100
80
60
40
20
0
19 0 0
17 0 1
18 1 3
42 0 2
58 0 3
39 0 1
41 0 3
1 0 0
5 0 0
102 0 0
24 2 0
7 3 7
3 1 1
1 2 3 4 5 6 7 8 9 10 11 12 13
Spot Groups
Large
Heavy
B757

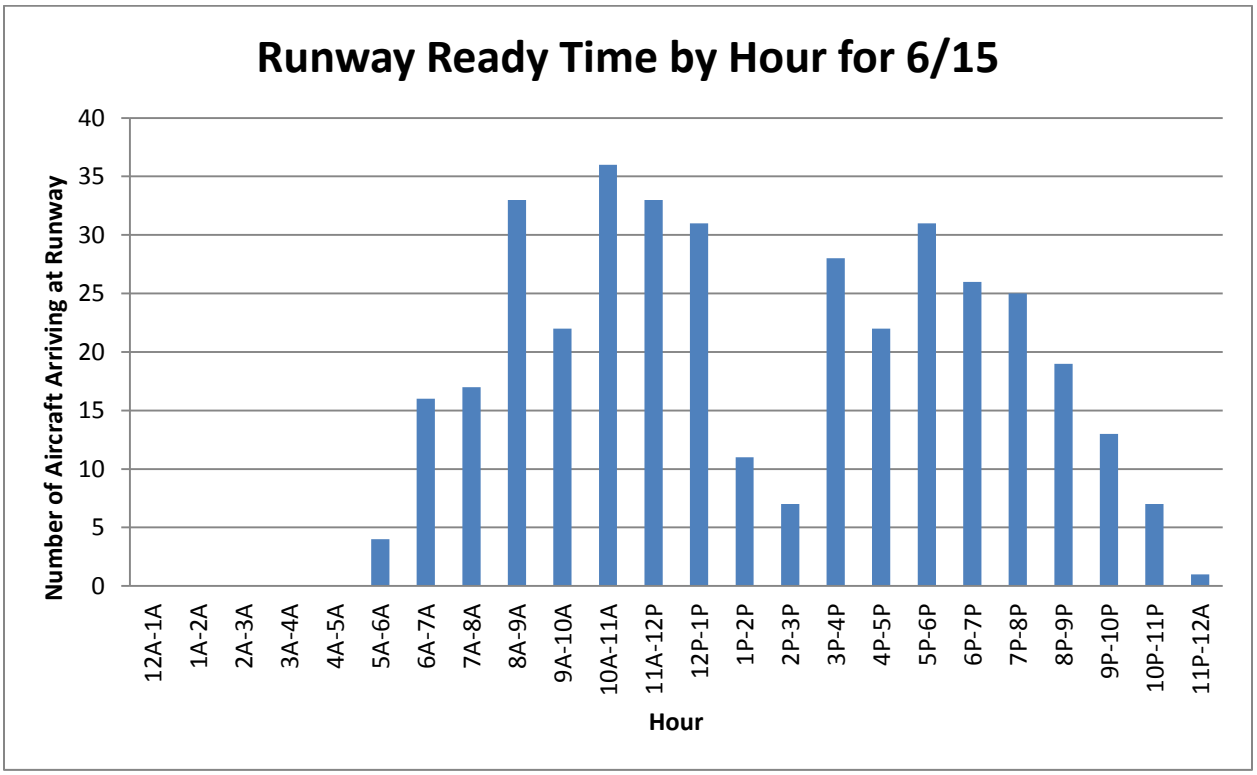

Spot Ready Time by Hour for 6/14
Number of Aircraft Ready at Spot
0
5
10
15
20
25
30
35
40
45
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour

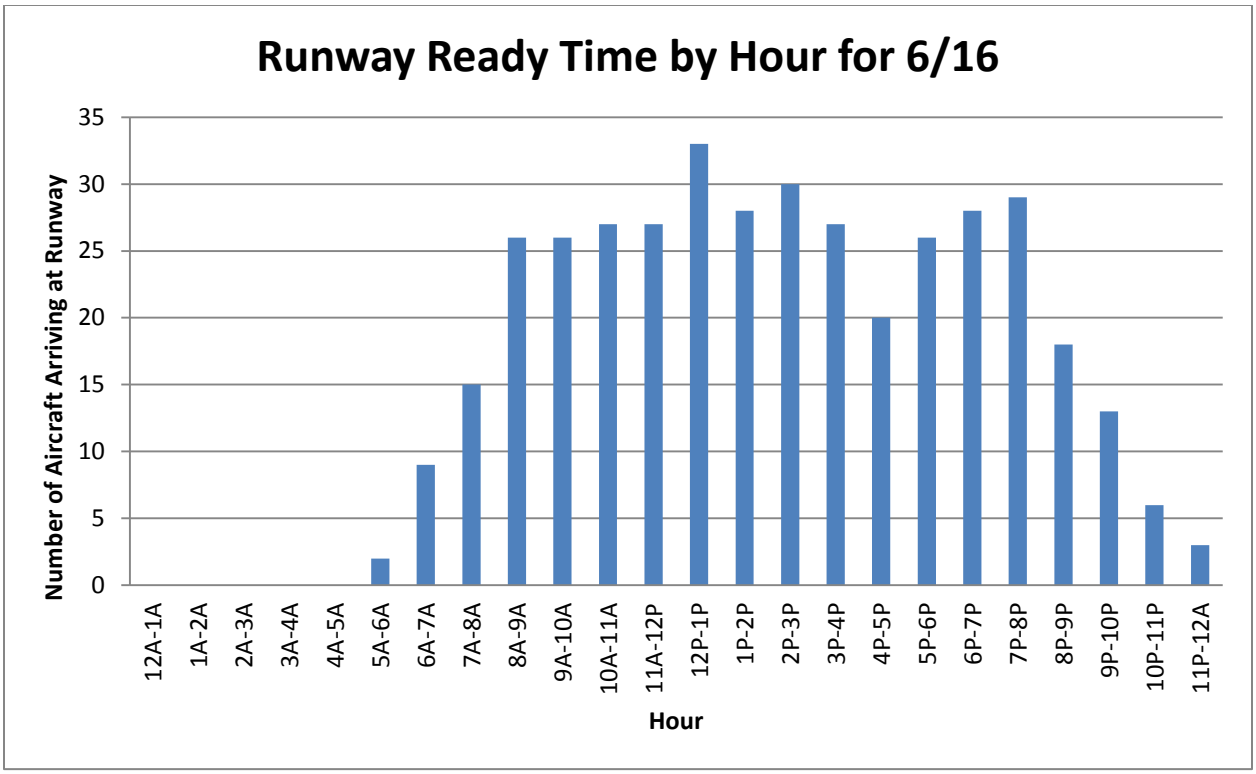

Spot Ready Time by Hour for 6/15
Number of Aircraft Ready at Spot
0
5
10
15
20
25
30
35
40
45
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour

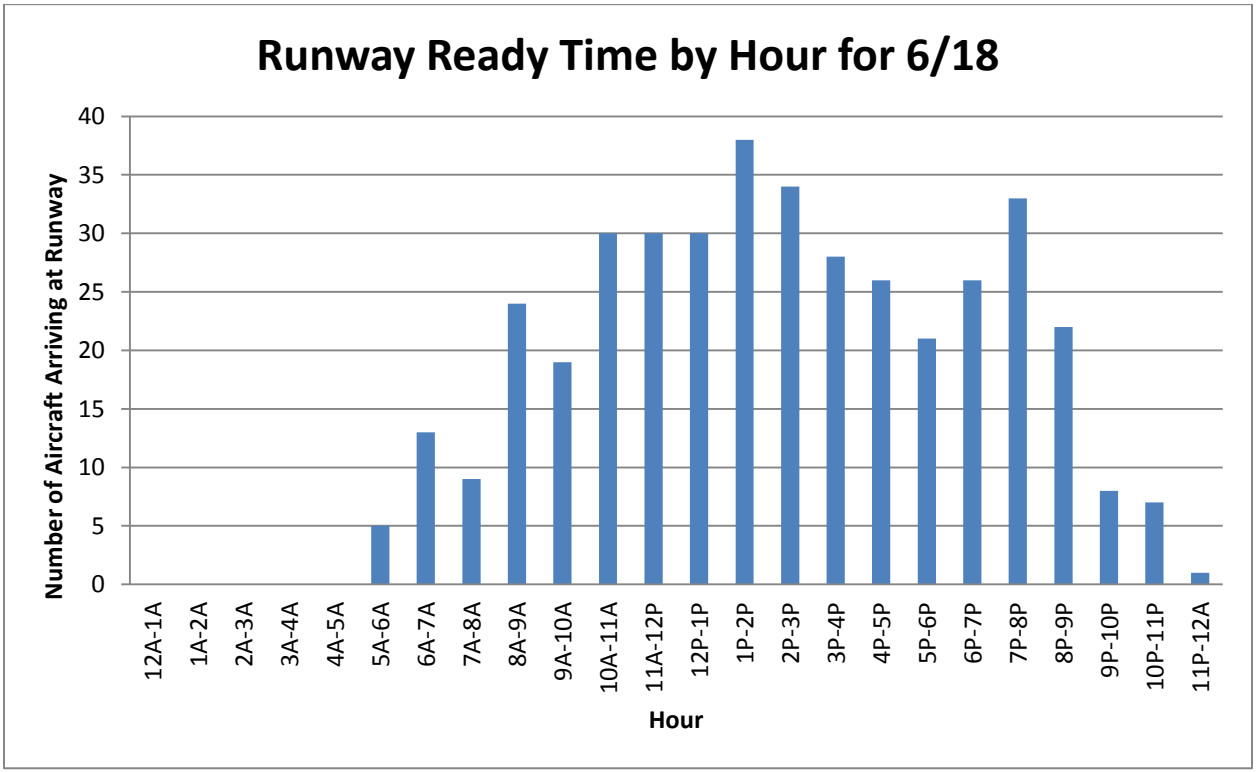

Spot Ready Time by Hour for 6/16
Number of Aircraft Ready at Spot
0
5
10
15
20
25
30
35
40
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour


Spot Ready Time by Hour for 6/18
Number of Aircraft Ready at Spot
0
5
10
15
20
25
30
35
40
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour


Runway Ready Time by Hour for 6/14
Number of Aircraft Arriving at Runway
0
5
10
15
20
25
30
35
40
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour


Runway Ready Time by Hour for 6/15
Number of Aircraft Arriving at Runway
0
5
10
15
20
25
30
35
40
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour


Runway Ready Time by Hour for 6/16
Number of Aircraft Arriving at Runway
0
5
10
15
20
25
30
35
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour


Runway Ready Time by Hour for 6/18
Number of Aircraft Arriving at Runway
0
5
10
15
20
25
30
35
40
12A-1A
1A-2A
2A-3A
3A-4A
4A-5A
5A-6A
6A-7A
7A-8A
8A-9A
9A-10A
10A-11A
11A-12P
12P-1P
1P-2P
2P-3P
3P-4P
4P-5P
5P-6P
6P-7P
7P-8P
8P-9P
9P-10P
10P-11P
11P-12A
Hour

Appendix D – Departure Delay Data for Individual Days

June 14th Departure Delays, in seconds								
		Baseline		Deterministic			w/Uncertainties	
Distribution of aircraft type		Real World	Simulated FCFS	DP Optimization	MILP Optimization	Original	DP Optimization	MILP Optimization
DFW	CPS = 0	89211	19237	23887	23278	19300	25882	24950
	CPS = 1			19797	20126	19306	21544	21964
	CPS = 2			19056	19416	19200	20789	21210
	NO CPS			N/A	19190	19100	N/A	20846
Even (25%)	CPS = 0	N/A	58288	76668	72777	58316	79255	75256
	CPS = 1	N/A		52437	51722	58312	55426	54765
	CPS = 2	N/A		45495	46881	58230	48562	49686
	NO CPS	N/A		N/A	45197	58292	N/A	47978

June 15th Departure Delays, in seconds								
				Deterministic			w/Uncertainties	
Distribution of aircraft type		Real World	Simulated FCFS	DP Optimization	MILP Optimization	Original	DP Optimization	MILP Optimization
DFW	CPS = 0	48757	12096	14304	13738	12039	15168	15338
	CPS = 1			12416	12213	11901	13579	13493
	CPS = 2			12006	11802	11882	13205	13117
	NO CPS			N/A	11606	12080	N/A	13078
Even (25%)	CPS = 0	N/A	27640	32561	31153	27507	34833	33235
	CPS = 1	N/A		25829	25433	27554	27682	27381
	CPS = 2	N/A		23723	23606	27542	25632	26578
	NO CPS	N/A		N/A	22673	27632	N/A	24740

June 16th Departure Delays, in seconds								
				Deterministic			w/Uncertainties	
Distribution of aircraft type		Real World	Simulated FCFS	DP Optimization	MILP Optimization	Original	DP Optimization	MILP Optimization
DFW	CPS = 0	67340	8617	10909	10402	8405	11629	11231
	CPS = 1			9037	8810	8670	10066	9812
	CPS = 2			8414	8417	8452	9263	9198
	NO CPS			N/A	8312	8517	N/A	9257
Even (25%)	CPS = 0	N/A	20704	24700	23553	20888	26352	25227
	CPS = 1	N/A		20016	19076	20808	21599	20691
	CPS = 2	N/A		17699	17825	20766	19004	19472
	NO CPS	N/A		N/A	17564	20784	N/A	19168

June 18th Departure Delays, in seconds								
				Deterministic			w/Uncertainties	
Distribution of aircraft type		Real World	Simulated FCFS	DP Optimization	MILP Optimization	Original	DP Optimization	MILP Optimization
DFW	CPS = 0	56588	10063	12579	12526	10018	13785	13308
	CPS = 1			10374	10365	10167	11723	11525
	CPS = 2			9776	9870	10070	11203	11008
	NO CPS			N/A	9804	9897	N/A	10708
Even (25%)	CPS = 0	N/A	27805	33971	32523	27269	35489	34223
	CPS = 1	N/A		25269	25107	27222	26834	26542
	CPS = 2	N/A		22932	23218	27473	24497	25265
	NO CPS	N/A		N/A	22476	27392	N/A	24148

Appendix E – Sequence Change and Separation Time Violation Results

The tables shown here contain the results for sequence change and separation time violations when stochastic taxi times are applied to deterministically optimized spot release schedules. The columns in the tables, from left to right, represent the sum of spaces that each flight shifted ("Sum"), the relative and weighted sequence changes ("Relative", "Weighted") as defined in the Results Chapter, the number of aircraft that arrived too early at the runway ("Sep Violations") and by how much on average ("Avg. Vio. Time"). These results were all averaged over 500 iterations of applying stochastic taxi times to the optimized spot release sequences.

We present the results for each day for both the Mixed Integer Linear Programming and Dynamic Programming methods, for both the aircraft mix at DFW for those days and the randomized even mix, and for different values of the Constrained Position Shifting variable.

June 14th, MILP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	0	0	0	118.45	1081.4
	CPS = 1	0	0	0	119.01	1070.8
	CPS = 2	0	0	0	118.13	1067.1
	NO CPS	0	0	0	120.94	1097.3
Even (25%)	CPS = 0	0	0	0	149.88	1437.4
	CPS = 1	0	0	0	149.13	1420.7
	CPS = 2	0	0	0	149.41	1375.2
	NO CPS	0	0	0	150.64	1412.1

June 15th, MILP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	5.5	3.67	4.67	80.6	866.8
	CPS = 1	5.34	3.61	4.61	81.41	859.26
	CPS = 2	5.4	3.65	4.65	81.55	868.5
	NO CPS	5.38	3.63	4.63	78.92	843.73
Even (25%)	CPS = 0	2	1	1	103.47	1111.8
	CPS = 1	2	1	1	106.16	1043.8
	CPS = 2	2	1	1	104.99	1154.6
	NO CPS	2	1	1	103.29	995.19

June 16th, MILP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	0	0	0	77.96	707.77
	CPS = 1	0	0	0	78.41	701.01
	CPS = 2	0	0	0	76.94	698.24
	NO CPS	0	0	0	79.02	735.95
Even (25%)	CPS = 0	0	0	0	100.91	935.71
	CPS = 1	0	0	0	100.37	928.49
	CPS = 2	0	0	0	99.27	898.83
	NO CPS	0	0	0	98.95	905.72

June 18th, MILP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	0	0	0	80.42	730.74
	CPS = 1	0	0	0	81.17	737.94
	CPS = 2	0	0	0	80.75	728.3
	NO CPS	0	0	0	79.46	707.03
Even (25%)	CPS = 0	0	0	0	105.74	964.98
	CPS = 1	0	0	0	103.85	950.27
	CPS = 2	0	0	0	101.84	926.99
	NO CPS	0	0	0	100.99	938.91

June 14th, DP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	0	0	0	118.05	1075.0
	CPS = 1	0	0	0	119.74	1085.9
	CPS = 2	0	0	0	120.35	1084.0
Even (25%)	CPS = 0	0	0	0	150.83	1395.5
	CPS = 1	0	0	0	150.64	1408.6
	CPS = 2	0	0	0	150.55	1390.0

June 15th, DP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	2	1	1	79.39	737.6
	CPS = 1	5.34	3.61	4.60	81.29	873.8
	CPS = 2	5.41	3.66	4.66	80.85	855.2
Even (25%)	CPS = 0	2	1	1	102.65	1096.1
	CPS = 1	0	0	0	105.27	1000.3
	CPS = 2	2	1	1	105.20	1028.5

June 16th, DP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	0	0	0	76.29	691.66
	CPS = 1	0	0	0	78.66	714.19
	CPS = 2	0	0	0	77.98	697.08
Even (25%)	CPS = 0	0	0	0	98.28	896.2
	CPS = 1	0	0	0	101.42	922.1
	CPS = 2	0	0	0	99.17	914.9

June 18th, DP						
Distribution of Aircraft Type		Sum	Relative	Weighted	Sep Violations	Avg. Vio. Time
DFW	CPS = 0	0	0	0	81.36	745.87
	CPS = 1	0	0	0	80.28	719.42
	CPS = 2	0	0	0	79.85	713.60
Even (25%)	CPS = 0	0	0	0	105.41	965.70
	CPS = 1	0	0	0	103.75	967.70
	CPS = 2	0	0	0	102.36	950.00