STOCHASTIC PROGRAMMING METHODS FOR SCHEDULING OF AIRPORT RUNWAY OPERATIONS UNDER UNCERTAINTY

A Thesis
Presented to
The Academic Faculty

by

Gustaf Sölveling

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
August 2012

STOCHASTIC PROGRAMMING METHODS FOR SCHEDULING OF AIRPORT RUNWAY OPERATIONS UNDER UNCERTAINTY

This thesis is dedicated to my beautiful girls,

Jaleh, Helena, and Lovisa.

You are my everything!

ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisors Ellis Johnson and John-Paul Clarke for their professional guidance, continuous encouragement and their belief in me that always gave me confidence to continue my research. Without their deep knowledge, positive spirit and valuable suggestions always pointing me in the right direction, I would not have been able to reach this point.

I would also like to express my deepest gratitude to Senay Solak, who from the moment I first set my foot in the research lab until today has been a great inspiration, collaborator and friend. I would also like to thank the other members of my committee, Shabbir Ahmed and Joel Sokol, for their valuable insights and suggestions.

To all my friends in the Air Transportation Laboratory, I would like to say thank you for all your help and good luck with your future research and careers. I would like to thank the faculty, the staff and the students in ISyE who all have contributed to the great experience I have had at Georgia Tech.

I would like to thank my family and friends both in Sweden and in the U.S., especially Annika, Svante, Janet, and Samad, for their constant support and encouragement.

Although their presence has lead to a slight drop in research productivity, nothing in the world means more to me than my two wonderful daughters, Helana and Lovisa. Their arrival showed me what really matters in life, and their amazing personalities make me a very proud father.

Last, but certainly not least, I would like to thank my wife Jaleh from the bottom of my heart for making all of this possible. Without your love, patience and understanding I would not have been able to reach this far. You are my great love and my best friend. Thank you!

TABLE OF CONTENTS

DEDICATION . . . . **iii**

ACKNOWLEDGEMENTS . . . . **iv**

LIST OF TABLES . . . . **viii**

LIST OF FIGURES . . . . **xi**

SUMMARY . . . . **xiii**

I INTRODUCTION **1**

1.1 Airport Runway Scheduling 1

1.2 Stochastic Programming 7

1.2.1 Two-Stage Stochastic Programming 8

1.2.2 Stochastic Branch and Bound 11

1.3 Thesis Contribution and Outline 11

II LITERATURE REVIEW **13**

2.1 Airport Runway Scheduling 13

2.1.1 Deterministic Methods 13

2.1.2 Robust/Stochastic Runway Scheduling 17

2.2 Machine Scheduling under Uncertainty 18

2.3 Stochastic Programming and Applications in Aviation 19

III COST FUNCTIONS FOR REDUCED ENVIRONMENTAL IMPACTS 21

3.1 Problem 21

3.2 Cost Functions with Environmental Impacts 24

3.2.1 Components of the Global Cost Function 25

3.2.2 Cost of Deviation from Latest Scheduled Operation Time 30

3.3 Simulations for Environmental Cost Analysis 31

3.4 Analysis and Policy Implications 32

3.4.1 Environmental Value of Optimization 32

3.4.2 Cost of Environmental Runway Scheduling to Airlines 36

3.4.3 Structure of Optimal Schedules 37

IV TWO-STAGE STOCHASTIC RUNWAY SCHEDULING 39

4.1 Problem Description 39

4.1.1 Two-Stage Scheduling Process 39

4.1.2 A Generic Modeling Framework for a Simplified Version of SRSP . 41

4.2 Two Alternative Formulations for SRSP-R and SRSP 44

4.2.1 Network Formulation of SRSP-R 44

4.2.2 Slot Formulation of SRSP-R 48

4.2.3 Distinctions of SRSP 50

4.2.4 Extension of the Two Formulations to SRSP 52

4.3 Valid Inequalities for SRSP 55

4.4 Solution Methodology for SRSP-R and SRSP 59

4.4.1 Scenario Decomposition 60

4.4.2 The Sample Average Approximation Implementation 63

4.5 Computational Framework and Experimental Setup 65

4.6 Computational Study and Analysis 68

4.6.1 Analysis-I: Efficiency of Flow vs. Slot Formulations for SRSP-R and SRSP 69

4.6.2 Analysis-II: Impact of Improvement Procedures for SRSP 77

4.6.3 Analysis-III: Comparison of SRSP vs. SRSP-R 83

4.6.4 Analysis-IV: Practical Performance 88

V SINGLE-STAGE STOCHASTIC SEQUENCING 92

5.1 Stochastic Branch and Bound 94

5.1.1 Algorithm 95

5.1.2 Partitioning 98

5.2 Algorithm Enhancements 99

5.2.1 Dynamic Sample Size Update and Statistical Testing 99

5.2.2 Termination Criteria 101

5.2.3 Heuristic Solution Methods 102

5.3 Implementation Details 103

5.3.1 Upper Bound Estimation 103

5.3.2 Lower Bound Estimation 103

5.4 Computational Study 105
5.4.1 Experiment Setup 105
5.4.2 Reference Experiment 107
5.4.3 Sample Size Selection 109
5.4.4 Dynamic Sample Size 111
5.4.5 Termination Criteria 113
5.5 Implementation for Runway Scheduling 115
5.5.1 Runway Scheduling Model 115
5.5.2 Solution Procedure for Runway Scheduling Model 117
5.5.3 Lower Bound Estimation 119
5.5.4 Upper Bound Estimation 123
5.6 Computational Study for Runway Scheduling 123
5.6.1 Input Data and Parameters 124
5.6.2 Configuration 1: General Layout 125
5.6.3 Configuration 2: Departure Scheduling with Crossings 130

VI CONCLUSIONS AND FUTURE RESEARCH 133

6.1 Conclusion for Two-Stage Sequencing Model 134
6.2 Conclusion for Single Stage Model 135
6.3 Directions for Future Research 136

LIST OF TABLES

1 Aircraft weight class categories. 2

2 Separation requirements, in seconds, for arrival operations on a single runway. 3

3 Separation requirements (in seconds) for two parallel runways. 23

4 Separation requirements (in seconds) at runway crossing. 23

5 External costs of aircraft emissions. (\$/lb) 28

6 Emission rates (lb/hr) based on the Boeing Fuel Flow Method 2. 28

7 Noise cost, \$/minute, for three different cost levels. 29

8 Crew and passenger delay costs. (\$/minute) 30

9 Arrival and departure rates for the different schedules. 31

10 Average arrival and departure rates after realization. 32

11 Annual estimate of the environmental value of optimization for 30 major airports in the U.S. 34

12 Same runway separation requirements for arrival operations. 42

13 Same runway separation requirements, in seconds, for departure operations. 42

14 Separation requirements, in seconds, for operations on close parallel runways. The leading operation is performed on one runway and the trailing operation is on the parallel runway. 42

15 Parameters used in the implementation of the scenario decomposition procedure. 63

16 Overall fleet mix at the 35 OEP airports in 2009. The fleet mix is obtained from 12 months of Aviation System Performance Metrics (ASPM) data. . 66

17 Description of instance groups used for model evaluation. 66

18 Computational results comparing the slot based formulation to the network flow based formulation of deterministic instances of SRSP-R. 71

19 Computational results comparing the slot based formulation to the network flow based formulation of deterministic instances of SRSP. 73

20 Computational results comparing the slot based formulation to the network flow based formulation of SRSP-R using scenario decomposition. 75

21 Computational results comparing the slot based formulation to the network flow based formulation of SRSP using scenario decomposition. 76

22 Computational results showing the impact of using valid inequalities (50) and (51) for the slot based formulation of SRSP. 78

23 Computational results showing the impact of using valid inequalities (50) and (51) for the network flow based formulation of SRSP. 79

24 Computational results showing the impact of the heuristic upper bound calculations. 80

25 Computational results for showing the impact of prioritization for the slot based formulation of SRSP over four different cases. 82

26 Computational results for showing the impact of prioritization for the network flow based formulation of SRSP over four different cases. 82

27 Computational results comparing the deterministic model of SRSP-R to the deterministic model of SRSP for the slot based formulation. 83

28 Computational results comparing the SRSP-R model to the SRSP model for the slot based formulation. 84

29 Computational results comparing the deterministic model of SRSP-R to the deterministic model of SRSP for the flow based formulation. 85

30 Computational results comparing the SRSP-R model to the SRSP model for the network flow based formulation. 86

31 Computational results comparing the characteristics of the full two hour algorithm to a truncated version that solves the problem in 20 minutes. The slot based model is used in the implementations. 89

32 Computational results comparing the best sequence found in the full two hour algorithm compared to the best sequence found in the truncated version that solves the problem in 20 minutes. 90

33 Computational results comparing the impact of runtime for the truncated SAA algorithm implementations. 90

34 Computational results showing the percentage of instances for which the best sequence, as identified by the 2 hour run, is found. The increase, relative to the 2 hour run, in the upper confidence interval limit is also shown for cases where the best sequence is the same and when it is not. 91

35 Sequence-dependent setup times for five categories of jobs, C1,. . . ,C5. The rows indicate the leading job and the columns indicate trailing job. 106

36 Triangular delay distribution, in seconds, from earliest possible release time r^0. 107

37 Parameters used for the reference experiment. 107

38 Average runtime change (39a) and average objective value change (39b) for the makespan objective for different values of N^0 and N^1. The table also shows the average number of samples used for the optimal sequence (39c) and the average number of iterations required by the algorithm (39d). . . . 110

39 Average runtime change (40a) and average objective value change (40b), for the makespan objective for different values of N^0 and N^1. The table also shows the average number of samples used for the optimal sequence (40c) and the average number of iterations required by the algorithm (40d). . . . 110

40 Change in runtime and average system delay for $N^0 = 10$ and different values of N^1. 111

41 Impact of dynamically changing the sample size for the makespan objective. In Tables 42a and 42b the change is with respect to the base case with $N^1_{\min} = 1, N^- = 2$ and $q^E = 10$. 112

42 Impact of dynamically changing the sample size for the system delay objective. In Tables 42a and 42b the change is with respect to the base case with $N^1_{\min} = 1, N^- = 2$ and $q^E = 10$. 112

43 Change in average objective function value, compared to $T^{\max} = 30$, and the percentage of instances in which no complete sequence was found for different time limits. 113

44 Change in average runtime, average objective function value and the percentage of instances in which no complete sequence was found for two different termination criteria. 114

45 Aircraft weight class distributions used for schedule generation. 124

46 Triangular delay distribution, in minutes, from earliest possible release time r^0. 125

47 Parameters used in the algorithm applied to runway scheduling. 125

48 Comparison between aircraft sequences obtained from the stochastic branch and bound algorithm (*Opt.*), the deterministic scheduling model (*Det.*) and the FCFS scheduling policy (*FCFS*). 126

49 Change in average makespan when using a deterministic model compared to using the stochastic branch and bound algorithm for four different input distributions. *Distr. 1* has no uncertainty whereas *Distr. 4* has the highest level of uncertainty. 128

50 Comparison between aircraft sequences obtained from the stochastic branch and bound algorithm (*Opt.*), the deterministic scheduling model (*Det.*) and the FCFS scheduling policy (*FCFS*). The result is for the special case when only departure operations and crossings are considered. 131

51 The preferred models with respect to run times for the restricted SRSP (SRSP-R) and SRSP for different input schedules. 135

LIST OF FIGURES

1 Example of uncertainty levels in arrival prediction and pushback delay. . . . 6

2 Network used in CPS algorithm presented in *Balakrishan and Chandran (2010)*. 16

3 A two-runway configuration similar to runways 22L and 22R at DTW. . . . 22

4 Cost components for arriving (top) and departing (bottom) aircraft. 26

5 Average reduction in environmental costs when an optimization based schedule is used. The savings are over a FCFS policy. 33

6 Environmental costs per flight based on schedules optimized with and without explicit consideration of the environmental factors. The values are according to the base emission cost estimate. 35

7 Analysis of environmental and non-environmental costs per flight for optimal and FCFS schedules. 36

8 Cost of including the environmental impact in optimization. 37

9 Maximum position shifts in the optimized schedule with respect to the FCFS schedule. 38

10 Network representation for the flow based formulation 45

11 Sample arrival and departure delay cost functions for Boeing 737, representing the Large aircraft weight class. 51

12 Hartsfield-Jackson Atlanta International Airport operates two-pairs of close parallel runways where 26L and 27R are used for departures, while 26R and 27L are used for arrivals. 67

13 Comparison of run times when the problem is solved using scenario decomposition versus when the deterministic equivalent formulation is used directly. The higher values correspond to the deterministic equivalent solution. . . . 69

14 Impact of the objective weights for SRSP-R. 87

15 Example of an incomplete branching tree for 4 jobs. 98

16 Average objective for three different scheduling methods: stochastic branch and bound algorithm (*Opt. Sequence*), optimal deterministic sequence (*Det. Sequence*), and a FCFS policy (*FCFS Sequence*). The error bars indicate the 95% confidence interval for the average objective value. 108

17 Average (with 95% confidence interval) and median runtime for instances solved by the stochastic branch and bound algorithm. The bars show the percentage of instances in which we are not able to find a solution within 30 minutes of runtime. 108

18 Upper and lower bound estimates for an instance with 9 jobs. 114

19 The runway configuration in this study consist of a pair of closely spaced parallel runways with one crossing where arrivals taxiing to the gate cross the departure runway. 115

20 Average makespan for three different scheduling methods: stochastic branch and bound algorithm (*Opt. Sequence*), optimal deterministic sequence (*Det. Sequence*), and a FCFS policy (*FCFS Sequence*). The error bars indicate the 95% confidence interval for the average makespan. 127

21 Average (with 95% confidence interval) and median runtime for instances solved by the stochastic branch and bound algorithm. The bars show the percentage of instances in which we were not able to find a solution within 20 minutes of runtime. 128

22 Average makespan for different levels of uncertainty around the nominal runway time. The triangular distributions are given as (lower limit, mode, upper limit) 129

23 Average makespan for three different scheduling methods in the special case where only departures and runway crossings are considered: stochastic branch and bound algorithm (*Opt*), optimal deterministic sequence (*Det*), and a FCFS policy (*FCFS*). The error bars indicate the 95% confidence interval for the average makespan. 131

24 Average (with 95% confidence interval) and median runtime for instances solved by the stochastic branch and bound algorithm. The bars show the percentage of instances in which we were not able to find a solution within 20 minutes of runtime. The result is for the special case when only departure operations and crossings are considered. 132

SUMMARY

Runway systems at airports have been identified as a major source of delay in the aviation system and efficient runway operations are, therefore, important to maintain and/or increase the capacity of the entire aviation system. The goal of the airport runway scheduling problem is to schedule a set of aircraft and minimize a given objective while maintaining separation requirements and enforcing other operational constraints. Uncertain factors such as weather, surrounding traffic and pilot behavior affect when aircraft can be scheduled, and these factors need to be considered in planning models. In this thesis we propose two stochastic programs to address the stochastic airport runway scheduling problem and similarly structured machine scheduling problems.

In the first part, we develop a two-stage stochastic integer programming model and analyze it by developing alternative formulations and solution methods. As part of our analysis, we first show that a restricted version of the stochastic runway scheduling problem is equivalent to a machine scheduling problem on a single machine with sequence dependent setup times and stochastic due dates. We then extend this restricted model by considering characteristics specific to the runway scheduling problem and present two different stochastic integer programming models. We derive some tight valid inequalities for these formulations, and we propose a solution methodology based on sample average approximation and Lagrangian based scenario decomposition. Realistic data sets are then used to perform a detailed computational study involving implementations and analyses of several different configurations of the models. The results from the computational tests indicate that practically implementable truncated versions of the proposed solution algorithm almost always produce very high quality solutions.

In the second part, we propose a sampling based stochastic program for a general machine scheduling problem with similar characteristics as the airport runway scheduling problem. The sampling based approach allows us to capture more detailed aspects of the problem, such as taxiway operations crossing active runways. The model is based on the stochastic branch and bound algorithm with several enhancements to improve the computational performance. More specifically, we incorporate a method to dynamically update the sample sizes in various parts of the branching tree, effectively decreasing the runtime without worsening the solution quality. When applied to runway scheduling, the algorithm is able to produce schedules with makespans that are 5% to 7% shorter than those obtained by optimal deterministic methods.

Additional contributions in this thesis include the development of a global cost function, capturing all relevant costs in airport runway scheduling and trading off different, sometimes conflicting, objectives. We also analyze the impact of including environmental factors in the scheduling process.

CHAPTER I

INTRODUCTION

Models explicitly considering uncertainty are an important area in the field of optimization. There are many real-world applications in which the input parameters to the model are not known with certainty at the time of model execution and decision making. One such area is the scheduling of aircraft operations at an airport. Because the costs associated with aircraft operations are relatively high, efficient operations can lead to significant cost and environmental savings. The most critical part of aircraft operations at airports is the runways, which have been shown to be bottlenecks in the entire aviation system. In this thesis, we consider stochastic optimization models for scheduling aircraft operations on the runway. The problem is introduced in the next section, followed by an introduction to stochastic programming.

1.1 Airport Runway Scheduling

It is reported that in 2007, nearly one in four scheduled airline flights arrived more than 15 minutes late to its destination (Ball et al. 2010). Furthermore, a third of these late arrivals were a direct result of the traffic demand exceeding the capacity of the aviation system. When the aviation system is operating under constrained conditions, any reduction in system performance leads to significant flight delays causing inconvenience and added cost to passengers and airlines. The potential for significant delays due to reduced system performance combined with a predicted increase in air traffic demand calls for the development of new and enhanced mathematical models that can assist in utilizing the available resources more efficiently. Air transportation authorities in the United States and Europe are investigating various technologies which they hope will increase capacity and efficiency in air transportation. Some of these efforts focus on runway operations planning and include tactical algorithm development for arrival and departure scheduling, taxiway routing and gate assignments (Eurocontrol 2011; Federal Aviation Administration 2011).

Table 1: Aircraft weight class categories.

Category	Max Certified Takeoff Weight	Example
Heavy	$\geq$ 300,000 lbs.	B747, B777, and A340
Large	$\geq$ 41,000 lbs. $<$ 300,000 lbs.	B737, A320, and ERJ145
Small	$<$ 41,000 lbs.	Most General Aviation and Executive Jets

The runway system is identified as the major source of delay in the departure process (Idris et al. 1998) and congestion in the terminal area constitutes a significant portion of all flight delays for arriving aircraft (Balakrishan and Chandran 2010). Consequently, increased runway capacity is critical to provide increased overall capacity in the air traffic system. Because infrastructure investments, such as new runways, are very costly and in many cases not feasible, improved planning and scheduling methods are a vital part to increased runway capacity.

Given a set of aircraft, whether we consider arrivals, departures or a combination, the objective of a runway scheduling algorithm is to set the time when each aircraft should begin using the runway. For arrivals, we denote the time at the runway to be the time of touchdown and for departures we denote the time at the runway to be the beginning of the takeoff roll. Each aircraft has an earliest and latest time at which it can be scheduled. In addition to time windows, separation requirements between each pair of aircraft must be enforced. In order to ensure safe distances between aircraft, the Federal Aviation Administration (FAA) mandates separation minima based on wake vortex categories for both arrival and departure operations. As the wake vortex categories are based on the maximum certified takeoff weight, the term aircraft weight classes is generally used. There are three aircraft weight categories specified in the official air traffic specifications document (Federal Aviation Administration 2010b), with the addition of two special cases concerning specific aircraft models, Boeing 757 and Airbus A380. The aircraft weight class categories can be seen in Table 1.

The pairwise minimum separation requirements between aircraft are typically given as a

Table 2: Separation requirements, in seconds, for arrival operations on a single runway.

		Trailing			
		H	7	L	S
Leading	H	96	138	138	240
	7	96	108	108	198
	L	60	72	72	162
	S	60	72	72	102

distance, but they can be converted to temporal separation requirements using representative landing and departure speeds. A matrix showing the temporal separation requirements for arrival operations on a single runway can be seen in Table 2, where the leading operation is given by the rows, and the trailing operation is given by columns.

As can be seen in Table 2, the separation requirements are not symmetrical. This asymmetry implies that the sequence of operations determines the time it takes to complete a set of operations, i.e. the makespan of the sequence. As an example, consider four arrivals, of which two are heavy (H) and two are large (L). The sequence H-L-H-L gives a makespan of 336 seconds from the first arrival to the fourth, whereas the sequence H-H-L-L gives a makespan of 306 seconds. This example shows that an additional 30 seconds of runway capacity can be gained by simply shifting a single pair of aircraft. This observation, in a more general sense, is the core of the runway scheduling problem. By taking advantage of the asymmetric separation requirements, mathematical models for runway scheduling can generate efficient runway schedules, which lead to increased runway capacity and reduced delay for the aircraft using the runway. On the other hand, the asymmetrical separation requirements make the problem NP-hard from a theoretical point of view.

The airport runway scheduling problem does not have a clearly defined objective. On the one hand, it is important to maximize the throughput on the runway to allow for as many aircraft as possible to use the available resources. On the other hand, it is important to impose fairness between all users of the system. This can be achieved by minimizing delays for individual aircraft. Other objectives exist, but these are the most common objectives used in runway scheduling models. There are occasions when the two objectives are in direct conflict. Consider an example where a single aircraft has to be delayed for an extended amount of time to achieve maximum throughput. All other users of the system experience less delay as throughput is increased, at the expense of the delayed aircraft. Clearly, this is not desirable from a fairness perspective. In Chapter 3 we develop a global objective function that captures both objectives simultaneously. More importantly, the global objective function considers all relevant costs associated with runway scheduling, including environmental costs and fuel costs. In addition, we compare the impact of introducing environmental cost into the objective function. The global objective function is later used in the stochastic model developed in Chapter 4.

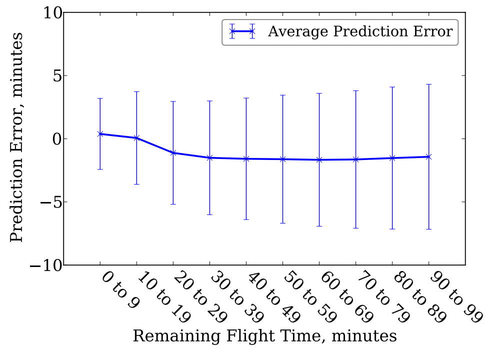
The problem of scheduling runway operations has received great attention in the literature over the past 35 years. Despite the fact that more than 60 articles have been published on the topic of arrival scheduling and planning (scheduling of departure operations has received far less attention in the literature), very few of these methods have been implemented and used in practice (Mesgarpour, Potts, and Bennell 2010). Reasons for this include:

- The problem is theoretically hard to solve, thus, models that include all operational constraints cannot be used to solve the general problem optimally and sufficiently fast.
- There is a discrepancy between the decision problem (what is the best way to schedule operations) and the control problem (how can air traffic controllers achieve the scheduled sequence of operations) that is in many cases overlooked. Researchers often focus on the decision problem without consideration of the control problem, which can lead to models giving solutions that cannot be implemented in practice.
- Many of the models presented in the literature focus on the static problem, i.e. when all the information is known upfront. In reality the runway scheduling problem is dynamic in the sense that new information is continuously fed to the planner who has to make decisions in close to real time.
- To assist air traffic controllers, the solution to the runway scheduling problem must be obtained fast.

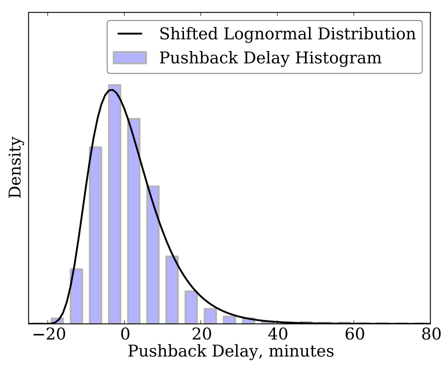
The models developed and presented in this thesis do not attempt to overcome the above shortcomings. The objective of this work is rather to investigate an area of runway planning and scheduling that has received very limited attention in the literature, namely runway scheduling under uncertainty. Almost all previous work has treated the problem as a deterministic problem, and assumed that the input parameters are known with certainty at the time of decision making. However, due to unpredictable factors such as weather, pilot behavior, surface traffic and other circumstances, deviations from the estimated input parameters are inevitable.

Given its similarities to other sequencing problems in areas such as manufacturing, the runway scheduling problems can be framed in terms of machine scheduling terminology. To simplify the categorization of different types of machine scheduling problems, Graham, Lawler, Lenstra, and Kan (1979) established the three-field notation $\alpha|\beta|\gamma$ to describe different versions of machine scheduling problems. The first parameter, α, describes the machine environment, the second parameter, β describes the job characteristics, and the γ parameter indicates what objective is minimized. Even when multiple dependent runways are considered, the runway scheduling problem can be cast as a machine scheduling problem on a single machine, i.e. $\alpha = 1$. The time-windows for specific aircraft (earliest time and latest time) and the separation requirements are denoted as release time, due dates, and sequence-dependent setup times, respectively. These characteristics are defined in the β-field. For the stochastic model developed in Chapter 5 we consider both throughput and system delay as objective functions in the γ-field.

Arrival and departure operations at airports contain a significant level of uncertainty due to the probabilistic nature of trajectories and other operational factors. Despite this fact, stochastic versions of the runway scheduling problem have previously never been considered in the literature. For the more general machine scheduling problem there exist models that do take uncertainty into account, but none of these models capture all the complexities included in airport runway scheduling. The earliest time an aircraft can be scheduled on the runway is an important input parameter common to all runway scheduling models. Up to this point, all runway scheduling models considered in the literature have treated the earliest time at the runway as a deterministic quantity. In reality however, there are several



(a) Sample mean and sample standard deviation for arrival prediction observations. The prediction error is calculated as the estimated arrival time subtracted from the actual arrival time.



(b) Distribution of pushback delay. The push back delay is calculated as the difference between the actual turn time and the scheduled turn time for those flights where there is no arrival delay.

Figure 1: Example of uncertainty levels in arrival prediction and pushback delay.

factors that can delay an aircraft. Arrivals can encounter unanticipated weather conditions or airspace congestion whereas departures can be delayed at the gate or during taxiing. Two examples of uncertainty levels can be seen in Figure 1. Figure 1a shows the error in arrival time prediction as a function of when the prediction was made. The data comes from FAA's Enhanced Traffic Management System. A similar study for departures, seen in Figure 1b, shows the distribution of pushback delay. The analysis was performed using date from the Bureau of Transportation Statistics database on on-time performance. One year of operations at a single airport are included in the study. For further details, the reader is referred to Solveling, Solak, Clarke, and Johnson (2011a).

In this thesis we develop two stochastic programs aimed at improving runway efficiency and utilization by explicitly considering uncertainty in the runway scheduling models. This is something that, to the best of our knowledge, never has been done before. These models are likely to result in better resource utilization than a first-come, first-served policy (FCFS) or deterministic optimization procedures under certain conditions, i.e. for traffic scenarios with high demand for arrivals and departures.

In the first model, we develop an efficient solution methodology to improve the solution of a large scale, stochastic integer programming based scheduling model. Two novel formulations are developed and tight valid inequalities are derived to strengthen these formulations.

We also analyze the impact of different objective function structures on computational efficiency and highlight the implications for similar scheduling problems.

To be able to capture more detailed aspects of the runway scheduling problem, such as taxiway operations crossing active runways, we also propose a sampling based stochastic program for the runway scheduling problem and other machine scheduling problems of similar structure. We propose extensions to an existing solution methodology, which makes it computationally tractable and can be used as an alternative to existing deterministic models.

1.2 Stochastic Programming

In contrast to traditional deterministic models, where the input data to the model is considered to be known, the input data to a stochastic model is not know with certainty at the time the model is evaluated. A Stochastic Program (SP) is an optimization model where the uncertain input parameters have known probability distributions. The most common stochastic programs are recourse models. In a general recourse model, high level decisions are made before uncertain parameters are realized. After some of the uncertain parameters have been realized, corrections (recourse) to higher level decisions are made and new, lower level decisions are made. The simplest, and most studied, recourse model is the two-stage model, where a first stage decision is made, uncertainty is realized, and a recourse decision is made in the second stage. In a two-stage stochastic program we are interested in finding the first stage decision that minimize the expected cost over all possible realizations and recourse decisions. In Chapter 4 we develop a two-stage stochastic integer program for the airport runway scheduling problem. Anagnostakis (2004) has identified a decision hierarchy for this particular type of problem that we exploit further. Recognizing that the makespan of an aircraft sequence-depends on the aircraft weight classes involved, and not on characteristics of individual aircraft, we find an aircraft weight class sequence in the first stage and assign individual aircraft to positions in the sequence in the second stage. This gives rise to a two-stage stochastic integer program, which we solve using scenario decomposition based on Lagrangian relaxation. In Chapter 5 we use a different approach to solving a stochastic integer program. The stochastic branch and bound algorithm is a sampling based approach where stochastic upper and lower bounds are generated for first stage decisions. Similar to deterministic branch and bound algorithms, the solution space is divided into smaller subsets and organized into a branching tree. In contrast to deterministic branch and bound algorithms, branches in the tree cannot be pruned as we do not know the exact bounds. We propose an approach to significantly increase the performance of the algorithm by focusing the computational effort on the parts of the branching tree that are most likely to contain the optimal solution.

1.2.1 Two-Stage Stochastic Programming

The structure of a general two-stage stochastic linear program is shown in Model (1)

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x + \mathbb{E}_\xi\left[Q(x,\xi)\right] \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{1}$$

where x is the vector of first stage decision variables and $\xi = (q, T, W, h)$ is the data to the second stage model. The function $Q(x, \xi)$ is the solution to the second stage problem, defined in Model (2).

$$\begin{aligned} Q(x,\xi) = \min_{y\in\mathbb{R}^m} \quad & q^T y \\ \text{s.t.} \quad & Tx + Wy = h \\ & y \geq 0 \end{aligned} \tag{2}$$

If the probability distribution ξ has finite support, there are only a finite number of outcomes of the uncertain events. Letting Ω be the set of all possible outcomes, also referred to as scenarios, the stochastic program in Model (1) can now be formulated as a deterministic problem, although it generally contains a huge number of decision variables. The new problem, referred to as the deterministic equivalent problem, can be seen in Model (3), where the parameter ϱ_ω is the probability of scenario $\omega \in \Omega$.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x + \sum_{\omega \in \Omega} \varrho_\omega q_\omega^T y^\omega \\ \text{s.t.} \quad & Ax = b \\ & T_\omega x + W_\omega y^\omega = h_\omega \\ & x \geq 0, y^\omega \geq 0 \end{aligned} \tag{3}$$

Classical solution methods to the large scale Linear Program (LP) in Model (3) are based on cutting plane algorithms. The first cutting plane algorithm for two-stage stochastic programs is the L-shaped method (Slyke and Wets 1969). This is an adaptation of the well know Benders' decomposition (Benders 1962), in which the problem is decomposed into a master problem representing the first stage decision and several second stage problems, one for each scenario. Improvements to the solution methodology include multi-cut algorithms, where several cutting planes are added in each iteration (Birge and Louveaux 1988). More recent algorithms for stochastic linear programs (SLP) are based on the regularized decomposition method (Ruszczyski 1986) and trust-region methods (Kiwiel 1990). By limiting the step-size in the master problem, improved computational efficiency can be obtained, thus leading to fewer iterations and faster convergence.

Two-stage Stochastic Integer Programs (SIP) have a structure similar to Models (1) and (2), with the complicating factor that some or all of the decision variables are restricted to be integer values. If the integer variables are limited to the first stage only, most of the theory and methods applicable to SLP can be used, with the exception that a Mixed Integer Program (MIP) needs to be solved in the first stage. Therefore, SIPs are assumed to have at least one integer variable in the second stage (Ahmed 2010). Analogous to SLP, the deterministic equivalent problem of the SIP can be formed and solved as a large scale MIP where cutting planes are included to take advantage of the problem structure (Guan, Ahmed, and Nemhauser 2009). Stage-wise decomposition methods for SIP include the integer L-shaped method (Laporte and Louveaux 1993) for two-stage problems with binary variables in the first stage. The method is later generalized to handle a first stage containing arbitrary variables and with integer recourse (Carøe and Tind 1998). Other methods based on stage-wise decomposition have been developed by Sen and Higle (2005), Sherali and Fraticelli (2002), and Ahmed, Tawarmalani, and Sahinidis (2004).

In the solution methodologies above, the problem is decomposed by stage, i.e. the decomposition scheme follows the same logical order as the information process. An alternative approach to formulate and solve the problem is by means of scenario decomposition. By creating copies of the first stage variables such that there is one copy for each scenario, the problem can be decomposed and solved independently for each scenario. However, to achieve an implementable policy it is required that the first stage variables take the same values across all scenarios. Model (4) shows a formulation based on scenario decomposition, where we explicitly add the constraint linking all first stage variables, also called the non-anticipativity constraint.

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \sum_{\omega\in\Omega} \varrho_\omega (c^T x^\omega + q_\omega^T y^\omega) \\ \text{s.t.} \quad & A x^\omega = b \\ & T_\omega x^\omega + W_\omega y^\omega = h_\omega \\ & x = x^\omega \\ & x^\omega \geq 0, y^\omega \geq 0 \end{aligned} \tag{4}$$

The idea of bundling scenarios for multistage (the generalization of two-stage models) stochastic programs was introduced by Rockafellar and Wets (1991) and solution methods based on dualizing the non-anticipativity constraints have been developed in Mulvey and Ruszczyski (1995) and Rosa and Ruszczyski (1996). Caroe and Schultz (1999) extended the idea and solution methodology to the case where some (or all) of the second stage variables are required to be integer. In the solution procedure, the nonanticipativity constraints are relaxed in a Lagrangian fashion, after which each scenario problem can be solved independently to obtain a Lagrangian dual solution. As the final solution requires the first stage variables to have the same value over all scenarios, this may be attained through the penalty components represented by the Lagrangian multipliers. The dual subproblem of finding optimal Lagrangian multipliers is solved using subgradient methods. For most integer problems a duality gap typically exists, requiring procedures to obtain good feasible first stage decisions.

1.2.2 Stochastic Branch and Bound

A slightly different approach for solving two-stage stochastic integer programs is the stochastic branch and bound algorithm. The algorithm is general in the sense that it does not require any particular structure on the variables, although it is well suited for problems with integer requirements on the first stage variables. Furthermore, there are no assumptions on the probability distributions, other than that samples can be drawn from the distribution. The initial model was developed for discrete decision variables (Norkin, Ermoliev, and Ruszczyński 1998) and it was soon thereafter generalized to handle global non-convex stochastic programs (Norkin, Pflug, and Ruszczyński 1998). Similar to deterministic branch and bound algorithms, the solution space is partitioned into smaller subspaces and organized in a branching tree. In contrast to traditional branch and bound algorithms, deterministic upper and lower bounds for nodes in the search tree are difficult to obtain. Therefore, stochastic upper and lower bounds are used to estimate the objective in each node.

1.3 Thesis Contribution and Outline

The contributions of this thesis have two dimensions, involving practical and theoretical issues. For the practical dimension, the contributions of this research are as follows: (1) Our proposed models aim at improving runway efficiency and utilization by implementing stochastic optimization procedures that are potentially tractable and implementable. Except for some general discussions, such models have not been previously considered in the literature. These models are likely to result in better resource utilization than FCFS or deterministic optimization procedures under certain conditions (2) We also note that our scheduling algorithm is an integrated arrival/departure scheduler with multiple interacting runways, as opposed to most other studies where only a single runway or a single type of operation is considered. This makes our models and analyses especially applicable in complex configurations where runway dependencies have an impact on the throughput of the dependent sets of runways, such as airport designs with close parallel runways. (3) Our inclusion of the environmental impact and costs in the optimization implementations are relevant from a sustainability perspective.

In addition to these practical implications, some theoretical contributions of this thesis are as follows: (1) We focus on computational performance and develop methods to improve the solution of a large scale, stochastic integer programming based scheduling model. (2) We develop two novel formulations and derive tight valid inequalities to strengthen these formulations. (3) We revive the stochastic branch and bound algorithm and propose statistical methods to dynamically change the computational emphasis for different parts of the branching tree. These improvements together with a careful implementation lead to a stochastic solution method for the runway scheduling problem, which can be used in close to real time for moderate to medium sized problems.

The outline of the thesis is as follows: In Chapter 2, previous work on the runway scheduling problem and related machine scheduling problems are presented along with a review of some stochastic models and solution procedures for stochastic models in aviation. In Chapter 3, we develop comprehensive cost functions for aircraft delay and use these functions to create a global objective function for the runway scheduling problem. The global objective function is later used in Chapter 4. We also investigate the impact of including environmental factors into the objective function. A two-stage formulation of the stochastic runway scheduling problem is presented in Chapter 4. We strengthen the formulation using valid inequalities, and propose a solution methodology based on scenario decomposition. In the computational study section, we evaluate the impact of various formulation implementations and analyze the value of stochastic scheduling. In Chapter 5, we present the stochastic branch and bound algorithm along with the proposed enhancements. We develop a solution strategy for stochastic machine scheduling problems and show that the algorithm works well for medium sized problems. We tailor the model to several versions of the stochastic runway scheduling problem and provide computational results showing the benefit of our proposed approach. A conclusion is provided in Chapter 6, where we also suggest directions for future research.

CHAPTER II

LITERATURE REVIEW

The literature review chapter is divided into three main sections. In the first part, we review earlier work on airport runway scheduling and attempt to emphasize various aspects of the scheduling process. The literature review shows that the problem is hard, from a strict mathematical sense and from a practical point of view. We conclude that the stochastic version of the runway scheduling problem has received very limited attention in the literature. In the second section we review models for machine scheduling problems and note that certain aspects of the runway scheduling problem have been captured in specific machine scheduling problems. In the last section we present previous work on stochastic programming models related to the formulations in this thesis and give examples of some stochastic programs applied to other areas of aviation.

2.1 Airport Runway Scheduling

2.1.1 Deterministic Methods

The deterministic runway scheduling problem has been well studied in the literature over the past 35 years. Almost all researchers refer to Dear (1976) as the first model published on the topic of optimized runway scheduling, and more than 60 articles have been published on the topic of arrival scheduling and planning. Scheduling of arrival operations has received far more attention in the literature than departure scheduling (Mesgarpour, Potts, and Bennell 2010).

We begin our literature review by focusing on *optimal* methods for airport runway scheduling. Bianco, Rinaldi, and Sassano (1987) and Bianco, Dell'Olmo, and Giordani (1997) introduce a combinatorial model for the arrival scheduling problem. The model is formulated as a machine-scheduling problem on a single machine with n jobs (aircraft), earliest release time for the jobs, and sequence-dependent setup times. In a special case, their formulation reduces to an Asymmetric Traveling Salesman Problem, showing that the runway scheduling problem is NP-complete (Garey and Johnson 1979).

Beasley, Krishnamoorthy, Sharaiha, and Abramson (2000) present a mixed integer program that has served as a base for describing the runway scheduling problem in many subsequent papers. In addition to providing the base formulation, they give computational results showing that their formulation works fairly well but that it is not capable of handling all real world instances within reasonable time limits. The main reason for this limitation is that the "big-M" construction to model non-convexities results in a weak LP-relaxation, an undesirable property in most solution methods for integer programs. Therefore, in the literature, the exact formulation has been used as a reference to compare the performance of heuristic methods rather than as a practical method to solve the scheduling problem in real-time.

When the sequence of operations is determined, the problem of optimizing the runway times, minimizing the objective and maintaining separation requirements, can be solved using an LP. This observation is explicitly stated in Ernst, Krishnamoorthy, and Storer (1999), who develop a modified simplex method to solve the timing problem. The modified simplex method is then embedded in a search heuristic that aims to find good sequences. Brinton (1992) suggests a branch and bound method where each level of the branching tree corresponds to a position in the sequence. Given n aircraft to be scheduled, the root node has n children, where each child corresponds to putting a specific aircraft first in the sequence. Each of these children then have $n - 1$ children, corresponding to the second position, and so on. If no pruning of the tree is made based on the objective bounds, this method would eventually enumerate all $n!$ sequences. Eun, Hwang, and Bang (2010) use a similar branch and bound scheme and incorporate a Lagrangian dual-decomposition method to calculate lower bounds on the objective, which significantly reduces the number of nodes in the search tree. For departure scheduling, Gupta, Malik, and Jung (2009, 2010) suggest a mixed integer formulation where aircraft are assigned to positions in the departure sequence. This formulation is combined with departure queue assignments, for increased departure throughput.

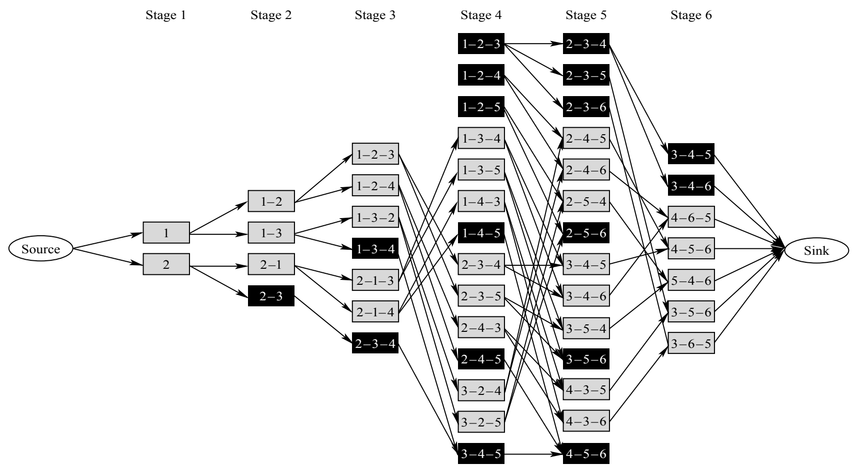
Recently, Sherali, Ghoniem, Baik, and Trani (2012) developed a MIP considering mixed arrival and departure scheduling on a single runway or on close parallel runways. Given the structure of the separation constraints, the mixed operation problem is as hard or harder to solve due to violated triangle inequalities among the separation constraints. In cases where all separation constraints satisfy the triangle inequality, e.g. in pure arrival (or departure) scheduling, it is sufficient to consider separation between consecutive operations to enforce separation between all pairs of operations. This is not the case in mixed operations, which introduce additional complexity into the formulation.

In contrast to the models presented above, where optimality of a solution can be proven but the run time is usually long, *heuristic* methods are able to provide the decision maker with solutions in real-time or close to real-time. In the dynamic environment faced by air traffic controllers, fast computation times are necessary if a sequence is to be implemented. In practice, first-come, first-served is the sequencing method most widely used (Brentnall and Cheng 2009). In a FCFS scheme, operations on the runway are sequenced based on the time the aircraft first enters a given point or is expected to reach a point. FCFS is considered to be a fair way of scheduling aircraft; furthermore, it requires minimal intervention from air traffic controllers as no re-sequencing is required. On the other hand, FCFS sequences rarely give sequences with the best throughput and/or minimized aircraft delay (Bennell, Mesgarpour, and Potts 2011). All practical applications, e.g. Atkin, Burke, Greenwood, and Reeson (2008); Balakrishan and Chandran (2010); Rathinam, Wood, Sridhar, and Jung (2009), are based on heuristic solution methods.

Based on the above observations, Dear (1976) developed the concept of Constrained Position Shifting (CPS) . The core idea is that in a good (optimal or close to optimal) sequence that can be implemented in practice, the position of each aircraft is shifted by a limited number of steps from the nominal FCFS sequence. The maximum allowable number of position shifts is determined through the parameter τ. When τ is small, typically 1,2, or 3, the search space of sequences is significantly reduced and efficient algorithms can be developed. CPS algorithms based on dynamic programming are presented in Psaraftis (1980), Dear and Sherif (1991), Venkatakrishnan, Barnett, and Odoni (1993), and Trivizas (1998). In the most recent work, the CPS problem is solved as a shortest path problem on a CPS network (Balakrishan and Chandran 2010). An illustration of the CPS network with six flights and $\tau = 1$ can be seen in Figure 2. The algorithm is able to handle precedence constraints, time window restrictions, multiple objectives (minimize makespan or sum of delays). Using a discretized network, the authors extend their formulation to include arbitrary cost functions and cases where the triangle inequality for the separation constraint is violated.

Figure 1. CPS network for $n = 6$, $k = 1$ generated from possible aircraft assignments shown on top.

Position	1	2	3	4	5	6
Possible	1	1	2	3	4	5
aircraft	2	2	3	4	5	6
assignments		3	4	5	6	



Note. Nodes shaded in black do not belong to any source-sink path and hence can be pruned from the network.

Figure 2: Network used in CPS algorithm presented in *Balakrishan and Chandran (2010).*

In the dynamic scheduling problem, the decision maker is continuously updated with new information, which requires re-optimization of the problem. The dynamic problem was first introduced within the CPS framework in Dear and Sherif (1991) and later in Venkatakrishnan, Barnett, and Odoni (1993). The problem is explicitly stated in Beasley, Krishnamoorthy, Sharaiha, and Abramson (2004), who define a displacement function to link decisions in successive iterations. Not only do solutions need to consider the traditional operational constraints, but the solution in the previous iteration must also be taken into account when finding a good sequence of operations. The authors use a genetic algorithm to solve the dynamic problem.

Genetic algorithms have received increased attention in the last decade as an efficient way to generate good sequences. Examples of genetic algorithms for arrival scheduling can be found in Hu and Paolo (2008) and Hu and Paolo (2011). In a recent publication (Yu, X.Cao, and Zhang 2011), cellular automata is embedded in a two-step algorithm. First, a good sequence of aircraft is found using cellular automata with updating rules aimed to enforce separation and reach a good objective value. In the second step, a local search heuristic is applied to set the landing times. Hu and Chen (2005) attempts to link the static and dynamic problem using a receding horizon method (rolling horizon). They show how good solutions to the dynamic problem can be obtained by using models for the static problem.

While there are no papers that study and account for environmental impacts within the runway scheduling framework, some planning tools aimed at quantifying different components of this impact have been proposed in the literature. Kesgin (2006) develops estimation methods for aircraft landing, take-off and taxi emissions, while Celikel, Hustache, Lepinay, Martin, and Melrose (2005) study the levels of environmental impact in different operational scenarios. Monroe, Jung, and Tobias (2008) analyze the environmental effects of eliminating arrival aircraft stops at active runway crossings using simulation tools. Hsu and Lin (2005) discuss an airline network design model to determine optimal routes and flight schedules in response to airport noise charges, while Nero and Black (1998) address the environmental costs in hub-and-spoke network design. Finally, a departure planning tool that also accounts for emission costs is proposed by Brinton, Cook, and Atkins (2007).

2.1.2 Robust/Stochastic Runway Scheduling

Almost all of the runway scheduling methods present in the literature assume that the target runway time (arrivals) or earliest runway time (departures) are known with certainty. The exceptions include:

- *Chandran and Balakrishan (2007)* give a CPS algorithm that includes uncertainty in the estimated time of arrival (ETA) at the runway and attempt to find robust schedules. The uncertainty is due to (lack of) aircraft equipage, pilot response time and behavior, and weather. The uncertainty used in the computational study is: $\pm$

150 seconds for aircraft with precise Flight Management System (FMS) and $\pm$ 300 seconds for aircraft not equipped with precise FMS.

- *Hu and Paolo (2008)* introduce uncertainty in the dynamic version, between successive iterations. In the computational study, 20% of the aircraft have their ETA randomly altered by $\pm$ 5 minutes.

- *Atkin, Burke, Greenwood, and Reeson (2008)* develop a near-time departure planner where the arrival time to the departure holding area is uncertain. The main purpose of their paper is to investigate how the uncertainty influences the proposed scheduling algorithm.

- The potential value of a stochastic approach to scheduling runway operations has been discussed in some recent studies such as Solveling, Solak, Clarke, and Johnson (2011a) and Gupta, Malik, and Jung (2011), where the authors in the latter evaluate the impact of deterministic departure scheduling under uncertainty.

Other than Solveling, Solak, Clarke, and Johnson (2011a), no stochastic models have been developed and/or applied to the runway scheduling problem. In their work, a simplified two-stage model is used to establish the value of stochastic runway scheduling. The integer program uses continuous second stage variables and employs Benders' decomposition to solve the model. Due to the simplified structure, the objective function in the second stage is only an approximation of the true cost. In their study, they find that a stochastic model has potential to increase runway throughput, especially when the demand exceeds runway capacity.

2.2 Machine Scheduling under Uncertainty

Given the similarities between the runway scheduling problem and certain machine scheduling problems, some runway scheduling studies describe their approaches based on the literature in machine scheduling (Bianco, Dell'Olmo, and Giordani 1997; Bianco, Rinaldi, and Sassano 1987). In particular, the asymmetric separation requirements between aircraft can be viewed as sequence-dependent setup times in a machine scheduling problem on a single machine. Typical objectives for machine scheduling problems with sequence-dependent setup times include minimization of the sum of completion times and minimization of total tardiness. When makespan minimization is used as the objective, the single machine scheduling problem can be transformed into a Traveling Salesman Problem (TSP) (Zhang and Zheng 1996). Our models of the stochastic runway scheduling problem correspond to single machine scheduling problem with probabilistic release times (and due dates) and sequence-dependent setup times. Many machine scheduling models with sequence-dependent setup times have been considered in the literature. Examples of solution methods include genetic algorithms (Lee and Asllani 2004), greedy randomized adaptive search (Gupta and Smith 2006), and simulated annealing (Tan, Narasimhan, Rubin, and Ragatz 2000). More examples can be found in the survey by Allahverdi, Ng, Cheng, and Kovalyov (2008). Note that all of these models consider the deterministic case. In relation to the multi-criteria objective structure developed in Chapter 3, we note that multi-objective models have been considered for machine scheduling problems as well. For example, Eren and Güner (2006) minimize a weighted combination of completion times and tardiness for a single machine problem with sequence-dependent setup times.

Stochastic machine scheduling problems in the literature primarily focus on probabilistic processing times (Cai and Zhou 2005; Skutella and Uetz 2005; Soroush and Fredendall 1994). However, the modeling framework in aircraft sequencing relates to machine scheduling models with probabilistic release times (or due dates). Such models have received very limited attention in the literature. One of the few examples is Wu and Zhou (2008), where the authors use a dynamic programming algorithm to solve a single machine scheduling problem with stochastic due dates and processing times. The model considered in that study does not include sequence-dependent setup times, and the objective involves only the minimization of maximum tardiness.

2.3 Stochastic Programming and Applications in Aviation

Applications of stochastic programming range from areas such as energy, telecommunications, manufacturing, to logistics (Sen and Higle 1999). Although no previous work has been published on stochastic airport runway optimization, there are examples of stochastic models in other areas of aviation. Several models for air traffic flow management have been developed. Common for all of them is that weather is the uncertain factor. Solution methods for these stochastic programs include a “fix-and-relax” heuristic for the deterministic equivalent problem (Alonso, Escudero, and Ortuno 2000), Lagrangian based scenario decomposition (Chang 2010), and a “tight” representation of the deterministic equivalent problem allowing for a direct solution using a commercial solver (Agustín, Alonso-Ayuso, Escudero, and Pizarro 2012).

Several stochastic methods for the ground-holding problems, where aircraft can be held at their origins due to limited capacity at the destination, have been presented. Ball, Hoffman, Odoni, and Rifkin (2003) show that their stochastic integer program formulation can be solved using an LP and still provide integer solutions. Similar characterizations are made in Kotnyek and Richetta (2006) and Glover (2010). Mukherjee and Hansen (2007) develop a dynamic stochastic integer program to solve the single- and multiple airport ground holding problem.

In addition to air traffic flow management and ground holding, stochastic models have been developed for airline network revenue management (Möller, Römisch, and Weber 2007) and crew scheduling (Yen and Birge 2006). For the revenue management problem, the deterministic equivalent problem is solved directly, and for the crew scheduling problem, a branching algorithm in which the recourse model decides how branching is done is developed.

We conclude the literature review with applications of the stochastic branch and bound algorithm. The practicality of the algorithm is investigated in Gutjahr, Hellmayr, and Pflug (1999) and Gutjahr, Strauss, and Wagner (2000). In the former, the stochastic branch and bound algorithm is applied to a stochastic single-machine-tardiness problem. The computational results indicate that the algorithm works well for small instances and gives good approximations for larger instances. A modified version of the algorithm is successfully used in Gutjahr, Strauss, and Wagner (2000), where the deterministic subproblems are solved using a heuristic, which allows many more samples to be generated during the course of the algorithm.

CHAPTER III

COST FUNCTIONS FOR REDUCED ENVIRONMENTAL IMPACTS

Several deterministic optimization models have been developed for scheduling arrivals and departures on a single runway or multiple runways. Given that the problem is combinatorial in nature, all exact approaches are based on integer models. As discussed in Chapter 1, the airport runway scheduling problem does not have a clearly defined objective. In this chapter we develop a global cost function that trades-off the two most commonly used objectives, runway utilization and system delay. We include all relevant costs, including environmental cost, in the objective function. We also analyze the impact of considering environmental costs in airport runway scheduling. The chapter is structured as follows: In Section 3.1 we present the problem and the optimization model used for the analysis. In Section 3.2 the global cost function is derived. The simulation used for the analysis is presented in Section 3.3 and the analysis and policy implications are presented in Section 3.4.

3.1 Problem

In this study we assume that the scheduled operations involve two parallel runways with a crossing taxiway, see Figure 3. This configuration is based on operations at Detroit Metropolitan Wayne County Airport (DTW), runways 22L and 22R, and is similar to the configuration at most other major airports. Without loss of generality, we assume that the outer runway is dedicated to arrivals, while departures take place on the inner runway. Thus, an arriving aircraft has to cross the departure runway, interacting with departure operations while taxiing to its assigned gate. In addition, as with many other airports, we consider the option for an arriving aircraft to go around the departure runway at the expense of increased taxiing time and costs. It is assumed that the longer route is not congested, and thus consists of a fixed taxiing time. On the other hand, if an aircraft is scheduled to cross the departure runway, it may be subject to taxiing speed adjustments or idling before the crossing. No runway crossing occurs for the departing aircraft.

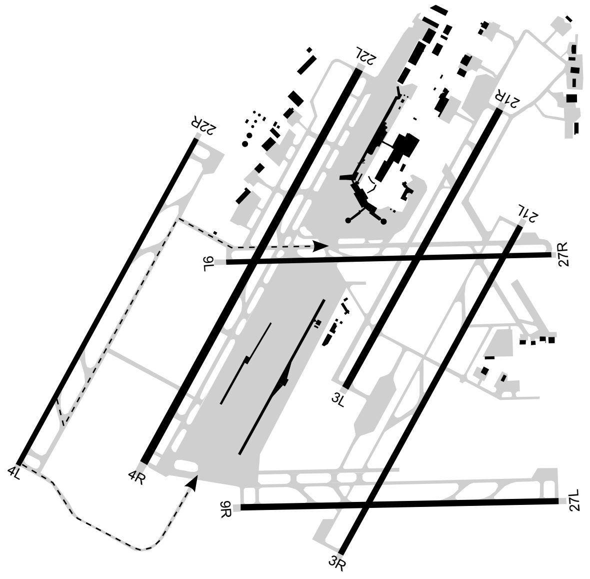


Figure 3: A two-runway configuration similar to runways 22L and 22R at DTW.

Given such a general runway configuration, we can consider a set of aircraft $\mathcal{A} = Arr \cup Dep$, where Arr and Dep represent the sets of arriving and departing aircraft. For each arriving aircraft $a \in A$, we assume that the following attributes are known at the time of decision making:

r_a :	scheduled arrival time for aircraft $a \in Arr$
l_a :	latest possible arrival time for aircraft $a \in Arr$

where the arrival time corresponds to the touchdown time for the aircraft. The latest arrival times are based on the possible airspeeds for the aircraft and other operational limitations. We consider a similar characterization for the departing aircraft, where the parameters are r_a, the scheduled departure or wheels-off time for aircraft $a \in Dep$, and l_a, the latest possible departure time for aircraft $a \in Dep$. We assume that an aircraft cannot depart before its scheduled time, and the upper bound l_a on the delay is imposed to prevent extensive delays at the gate. Additional parameters and bounds can be defined to model operational limitations such as maximum taxi times or queue times.

Key inputs for a runway operations scheduler are the separation requirements between different operations. These are based on safety measures imposed by the air traffic authority, and depend on the type of aircraft and operation. We define $s_{i,j}$ as the minimum required

Table 3: Separation requirements (in seconds) for two parallel runways.

		Trailing Operation							
		AH	A7	AL	AS	DH	D7	DL	DS
Leading Operation	AH	96	138	138	240	15	15	15	15
	A7	96	108	108	198	15	15	15	15
	AL	60	72	72	162	15	15	15	15
	AS	60	72	72	102	15	15	15	15
	DH	48	56	56	80	90	90	120	120
	D7	48	56	56	80	90	90	120	120
	DL	48	56	56	80	60	60	60	60
	DS	48	56	56	80	60	60	60	60

Table 4: Separation requirements (in seconds) at runway crossing.

		Trailing Operation							
		AH	A7	AL	AS	DH	D7	DL	DS
Leading Operation	AH	-	-	-	-	10	10	10	10
	A7	-	-	-	-	10	10	10	10
	AL	-	-	-	-	10	10	10	10
	AS	-	-	-	-	10	10	10	10
	DH	35	35	35	35	-	-	-	-
	D7	40	40	40	40	-	-	-	-
	DL	40	40	40	40	-	-	-	-
	DS	100	100	100	100	-	-	-	-

time separation between aircraft operation i followed by operation j, and present the values of these parameters in Tables 3 and 4.

The separation requirements in Table 3 are defined for operations in Instrument Flight Rules (IFR) conditions on a configuration with one runway for arrival operations and one runway for departure operations where the distance between the runways is less than 4300 feet. In Tables 3 and 4, the first letters of the row and column headers represent the type of operation, i.e. arrival or departure, while the second letters represent the aircraft weight class. The considered aircraft classes are heavy, Boeing 757, large and small, denoted by H, 7, L and S. Wake turbulence caused by the leading aircraft is the source of the separation requirements for runway operations (Bly 2005).

The runway scheduling problem consists of minimizing an objective function subject to these separation requirements. To this end, we define:

$x_{i,j}$: 1 if operation $i \in \mathcal{A}$ is scheduled before operation $j \in A$, 0 otherwise

t_a : time of operation for aircraft $a \in \mathcal{A}$

t_ℓ : time of the latest arrival or departure operation

An aircraft operation may refer to an arrival, a departure or a runway crossing action. If we let $g(\mathbf{t}, \mathbf{x})$ represent the overall cost function with environmental impact and let M be a sufficiently large constant, then the runway operations scheduling problem can be expressed as follows:

$$\min \quad g(\mathbf{t}, \mathbf{x}) \tag{5}$$

$$\text{s.t.} \quad t_a \geq r_a \qquad \forall a \in \mathcal{A} \tag{6}$$

$$t_a \leq l_a \qquad \forall a \in \mathcal{A} \tag{7}$$

$$t_\ell \geq t_a \qquad \forall a \in \mathcal{A} \tag{8}$$

$$t_j - t_i \geq s_{i,j} - M(1 - x_{i,j}) \qquad \forall i, j \in A \times A \tag{9}$$

$$t_i - t_j \geq s_{j,i} - M x_{i,j} \qquad \forall i, j \in A \times A \tag{10}$$

$$\mathbf{t} \in \mathbb{R}_+ \quad \mathbf{x} \in \mathcal{B}$$

Constraint sets (6) and (7) limit the times for which each aircraft can arrive/depart and constraint set (8) identifies the time for the last activity on the runway. Operations need to meet the mandated separation requirements, which are defined in a general form through constraint sets (9) and (10). These constraints are used to model the separation requirements for the aircraft at the runway-taxiway intersection, as well as the minimum and maximum taxi idling time at the runway crossing. Treating (5)-(10) as a core model, it is possible to include additional decisions into the runway-scheduling problem by introducing new variables and constraints.

3.2 Cost Functions with Environmental Impacts

An accurate and comprehensive representation of function $g(\mathbf{t}, \mathbf{x})$ is needed for the validity of the model. The objectives in the runway-planning problem are twofold: maximization of throughput by re-sequencing the arrival and departure streams, and minimization of operational and environmental costs associated with deviations from scheduled operation times. By putting a monetary value on these conflicting objectives, a global cost function can be constructed that consists of several parts, each of which involves some impact on the environment.

The overall cost function consists of a number of elements. First, for each arrival aircraft we consider the cost of deviating from the scheduled arrival time and the cost of deviating from the fuel-optimal arrival time, if different from the scheduled arrival time. For these latter aircraft we also consider the cost of additional time spent on the taxiway due to interactions with the departure operations. The second component captures the delay cost for departing aircraft that can be incurred either at the gate or on the taxiway. The third component captures the cost of runway throughput. Hence, we can express the objective function as:

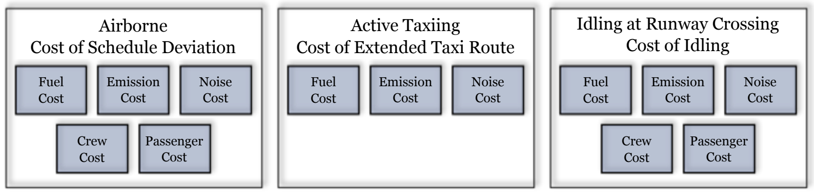
$$\begin{aligned} g(\mathbf{t}, \mathbf{x}) = & \sum_{a \in Arr} (\text{Cost of Schedule Deviation} + \text{Cost of Taxiway Operations}) + \\ & \sum_{a \in Dep} (\text{Cost of Delay}) + \text{Runway Throughput Cost} \end{aligned} \tag{11}$$

where the individual parts with an emphasis on environmental cost components are detailed below.

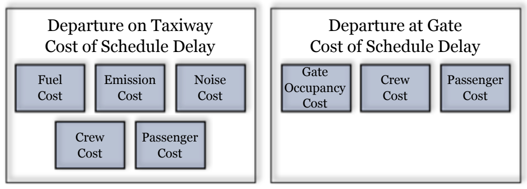
3.2.1 Components of the Global Cost Function

An important aspect of the global cost function is that it models the cost of deviations from a nominal schedule, i.e. the additional cost incurred by the scheduling process. Hence, all components modeling the costs for individual aircraft assume a zero cost if the aircraft arrives/departs on schedule, taxies according to the pre-assigned route and no delay is experienced due to runway crossings.

The planning model considers two decisions related to arriving aircraft that need to be captured in the global cost function. The first is the runway arrival time. The arriving aircraft is airborne when the decision is made and the deviation is easily calculated as the difference between the updated optimized arrival time and the scheduled arrival time. Two



(a) Cost components for arrival aircraft.



(b) Cost components for departure aircraft.

Figure 4: Cost components for arriving (top) and departing (bottom) aircraft.

actions can be taken to achieve the desired deviation, namely speed-change and vectoring. We assume that both actions are possible in the cruise phase, but only vectoring is allowed in the descent and landing phase.

The second decision deals with taxiing operations. Each arriving aircraft has a predefined taxi route based on the shortest unimpeded taxi time to the gate. For those aircraft crossing the departure runway, it may be necessary to idle while waiting for an opportunity to cross the runway. Based on the cost of idling and the fixed additional cost of taxiing the longer route around the departure runway, the model can change the initial taxi route. It is assumed that crew and passenger costs are accounted for in the planning stage of nominal taxiway operations. The three phases of the arrival operation with corresponding cost components are presented in Figure 4a.

A similar representation for departing aircraft can be seen in Figure 4b. By assumption, a departing aircraft cannot be scheduled earlier than the initially scheduled departure time, and thus, the cost function only needs to consider delays. Delay cost can be incurred either at the gate or on the runway, depending on whether the aircraft has pushed back from the gate or not. It is assumed that the engines are turned off when the aircraft is at the gate, therefore, no fuel, emission or noise cost is considered. Instead there is an opportunity cost for using the gate. For departures that have already pushed back we assume that the engine is idling and the delay is experienced on the taxiway.

3.2.1.1 Fuel costs

The additional fuel costs are proportional to the amount of additional fuel burned. For vectoring, taxiing and idling, the fuel flow is obtained for the standard four aircraft weight classes (Ren and Clarke 2008). Depending on what stage of the process an airborne arrival aircraft is in, the fuel flow for an appropriate flight level is used.

For speed adjustments, the fuel flow is obtained by calculating the required speed adjustment to achieve the deviation. Assuming an instantaneous speed-change, the additional fuel flow is calculated as the difference between the fuel flow with the new speed and the fuel flow with the initial speed. If the aircraft does not fly at fuel-optimal speed, a change in speed can lead to a decrease in fuel flow.

3.2.1.2 Emission costs

For the emission costs, we consider the emission of CO_2 caused by the burning of jet fuel, as well as the emission of other pollutants, such as SO_2, NO_x, CO, and HC. The emissions of CO_2 are proportional to the fuel flow with a factor of 3.14, i.e. 1 lb of jet fuel emits 3.14 lb of CO_2 (EIA 2009). The cost of CO_2 emissions is obtained by multiplying the additional fuel burn with $3.14c^{CO_2}$, where c^{CO_2} is the cost of CO_2 emission based on the current market price on the CO_2 emission trading market. In addition, Celikel, Hustache, Lepinay, Martin, and Melrose (2005) report three levels of estimates for aircraft emission costs by pollutant type, which are shown in Table 5. We use all three levels of estimates when conducting the sensitivity analysis of the environmental impact of schedules on variation in these costs.

For other pollutants, the emission rates are based on the Boeing Fuel Flow Method 2 (Dubois and Paynter 2006). Representative equipments in each weight class are used to approximate the emissions for all phases of the flight, including taxiing and idling (Table 6). Due to the wide variety of aircraft types in the small weight class, and the relatively small impact they have on the overall emissions from aircraft, we omit this class when estimating environmental cost. Instead, a fraction of the emission cost for the large weight class is used as an approximation. The unit cost for each pollutant is again based on the estimates in Table 5.

Table 5: External costs of aircraft emissions. ($/lb)

	Low	Base	High
CO	0.07	0.09	0.13
CO_2	0.007	0.024	0.042
HC	1.7	3.6	5.5
NO_x	2.9	4.1	6.9
SO_2	1.4	3.9	7.2

Table 6: Emission rates (lb/hr) based on the Boeing Fuel Flow Method 2.

	Phase 1			Phase 2			Phase 3		
	L	7	H	L	7	H	L	7	H
CO	8.1	5.1	14.6	60.1	503.6	366.4	54.6	455.4	307.9
HC	1.7	4.0	1.9	12.4	6.1	111.9	11.3	6.1	86.9
NO_x	53.8	32.2	94.4	4.6	3.2	8.3	5.1	3.8	6.2
SO_2	3.7	4.4	7.4	1.0	1.2	1.7	1.1	1.4	1.8

	Taxiing			Idling		
	L	7	H	L	7	H
CO	43.4	98.9	178.1	52	340.9	225.9
HC	9.0	5.3	40.9	10.8	5.9	56.7
NO_x	7.5	9.5	11.5	5.8	4.8	9.0
SO_2	1.4	2.4	2.6	1.2	1.6	2.3

Similar to the calculation of fuel flow for arriving aircraft, we divide emission costs into two components, depending on whether the maneuvers correspond to speed adjustment or vectoring. While the emission rates in Table 6 are used for vectoring based maneuvers, emission rates centered on the optimal cruise speed are used to calculate emission costs for speed adjustments.

In defining the cost of additional noise due to deviation in flight time, we use the estimates of Levinson, Kanafani, and Gillen (1999) who suggest an average noise cost of $0.043 per kilometer traveled. While this is a crude estimation, it enables us to calculate a noise impact measure based on the time deviation and the average speed of a given aircraft. The approximation does not directly model the complex relations that use community characteristics and property values in calculating noise costs of runway operations, which are discussed in Nelson (2004) without any specifics on the dollar amounts for the costs. However, the use of a linear relationship between flight time and noise costs in our optimization model can be justified. First our analysis considers the additional cost of operations due

Table 7: Noise cost, \$/minute, for three different cost levels.

	Weight class (passengers)			
Level:	S (70)	L (160)	7 (230)	H (270)
Low	1.9	4.4	6.4	7.5
Base	2.8	6.3	9.1	10.7
High	3.6	8.2	11.8	13.8

to inclusion of environmental factors, which are incurred by delaying or speeding the aircraft mostly through vectoring. The number of operations and aircraft velocities, therefore, remain constant and only the flight times change due to the vectoring maneuvers. Moreover, these changes are mostly small. Even if an exact relationship between flight time and noise costs were to be established, these need to be defined individually for each specific airport. Thus, we utilize the approximation of Levinson, Kanafani, and Gillen (1999) as a general relationship, and perform sensitivity analysis around that approximation to serve as a valid input to the runway scheduling algorithm. Using the value as a base cost level, we also consider low and high noise cost parameters to account for any inaccuracy in the approximation. The resulting costs can be seen in Table 3.2.1.2.

3.2.1.3 Crew and Passenger costs

To estimate the cost of delays for crew and passengers, we use the information found in Cook, Tanner, and Anderson (2004). The report includes delay costs for 12 specific aircraft types under three cost scenarios. For the purpose of estimating crew and passenger costs we consider the base scenario that includes costs for two types of delays; a short delay of 15 minutes and a long delay of 65 minutes. The costs are shown in Table 8. It is assumed that any delay of 15 minutes or less does not incur crew and passenger costs, resulting in the short delay costs being zero. Justifications for the no cost assumption for delays less than 15 minutes are detailed such that delays below this threshold are typically included as buffers in schedules. Moreover, the minimum connection time for passengers and minimum turn around time for aircraft are typically around 30 minutes. It is concluded that any crew and passenger costs for short delays will be incurred only in rare cases and even then their impact will be minimal.

Table 8: Crew and passenger delay costs. ($/minute)

	Crew Cost		Passenger Cost	
	Short	Long	Short	Long
S	0	575	0	1165
L	0	800	0	3344
7	0	893	0	5533
H	0	1926	0	8226

The crew cost portion of costs includes salaries and expenses for flight and cabin crew. The passenger cost portion includes hard cost (re-bookings, accommodations, etc.) together with soft cost (e.g. lost market share) for the airline.

Given these cost parameters, we can calculate the marginal cost for a long delay assuming that marginal delays grow linearly. The marginal cost for any delay is then calculated through interpolation. The cost of occupying a gate depends on the subsequent use of the gate, as the delay at the gate can potentially cause a delay for an incoming aircraft. The gate occupancy cost is based on the idling cost, including crew and passenger cost, for the incoming aircraft. Since the model only considers the arrival time to the runway, the time by which the arriving aircraft reaches the gate is approximated using the unimpeded taxi in time and the expected taxi delay.

3.2.2 Cost of Deviation from Latest Scheduled Operation Time

With a global cost function defined for each aircraft in the data set, a cost function for runway throughput can be developed. Let $\mathcal{A}'$ be the aircraft succeeding the aircraft in $\mathcal{A}$. For both sets $\mathcal{A}$ and $\mathcal{A}'$, the aircraft are ordered by scheduled departure or arrival time. For the purpose of constructing the cost function, we let $\bar{a} \in \mathcal{A}$ be the last scheduled aircraft. Furthermore, we let Γ_i represent the delay of $\bar{a}$ for delay level i. We begin by assuming that $\Gamma_0 = 0$, i.e. $\bar{a}$ is optimized such that $t_{\bar{a}} = r_{\bar{a}}$ and the last aircraft does not experience any delay.

With this assumption, the aircraft in $\mathcal{A}'$ are scheduled according to a first-come, first-served policy until the FCFS policy gives a zero delay for an aircraft or all the aircraft in $\mathcal{A}'$ are scheduled. The cost of the delay for each aircraft in $\mathcal{A}'$ is calculated, and adding these costs together, the cost for violating the last scheduled runway time in $\mathcal{A}$ by Γ_0 minutes

Table 9: Arrival and departure rates for the different schedules.

Length of schedule [min]	Arrival rate [flights/hr]	Departure rate [flights/hr]	Cumulative rate [flights/hr]
120	21.2	25.8	47.0
105	24.3	29.6	53.9
90	28.5	34.6	63.1
75	34.2	41.6	75.8
60	43.0	52.2	95.2

is obtained. Repeating this with the assumption that the last aircraft in $\mathcal{A}$ is delayed by $\Gamma_0, \Gamma_1, \ldots, \Gamma_\theta$ minutes, a cost function for runway utilization can be constructed. The interval lengths Γ_i and parameter θ can be chosen to achieve a desired level of detail in the piecewise linear function that is created.

All the components in the global cost function $g(\mathbf{t}, \mathbf{x})$ are convex functions. In the mathematical model we approximate $g(\mathbf{t}, \mathbf{x})$ with piecewise linear functions which allows for modeling using standard techniques.

3.3 Simulations for Environmental Cost Analysis

The input to the simulations are based on actual flight schedules at major airports, and are representative of conditions during peak periods. More specifically, we initially consider the arriving and departing traffic at DTW for a 2 hour peak on September 26th, 2006. Using this schedule as a baseline, we create additional representative schedules for heavier volumes of traffic by "compressing" the baseline schedule to obtain higher operation rates. We compress the initial 120 minute schedule to schedules of 105 minutes, 90 minutes, 75 minutes and 60 minutes. The arrival and departure rates for the different schedules are presented in Table 9.

For each schedule, we simulate 40 randomly generated instances by creating a realization of the actual departure/arrival time for each aircraft. The realizations are based on pushback delay distributions, taxi time distributions, and for arrival aircraft, delay distributions while in transit. The pushback delay distributions and the transit time delay distributions are obtained by analyzing the Aviation System Performance Metrics (ASPM)

Table 10: Average arrival and departure rates after realization.

Length of schedule [min]	Arrival rate [flights/hr]	Departure rate [flights/hr]	Cumulative rate [flights/hr]
120	17.2	23.2	40.4
105	18.8	26.4	45.2
90	20.7	29.9	50.6
75	23.2	34.4	57.6
60	27.2	41.4	68.6

and Enhanced Traffic Management System (ETMS) data provided by the FAA, while taxi time distributions were based on the analysis of Simaiakis and Balakrishan (2009).

The initial schedule for the 2 hour peak is extended with an additional hour of flights to capture the impact of violating the latest scheduled time. After a schedule has been realized, the flights in the first 2 hours are included in the optimization, whereas the remaining flights are used to calculate the delay cost for subsequent operations. Similarly for the compressed schedules, we include all flights in the first x, $x \in \{105, 90, 75, 60\}$ minutes of the schedule in the optimization. The simulated arrival and departure rates are presented in Table 10.

As representative aircraft types in each weight class for the schedule, we use Boeing 767-300 for the heavy weight class, Boeing 757-200 for the B757 weight class, and Boeing 737-800 for the large weight class. Due to the wide spectrum of aircraft, e.g. jet and turbo-prop, in the small weight class we assume that the costs used for this weight class are 70% of what we use for aircraft in the large weight class.

3.4 Analysis and Policy Implications

3.4.1 Environmental Value of Optimization

The major policy related question involves the value of an optimized schedule from an environmental perspective. In other words, how much reduction in environmental costs can be achieved through an optimization based approach over the currently implemented FCFS system, and how does this value change with increasing schedule density? The response is given in Figure 5 where the environmental value of optimization is displayed for different levels of emission cost estimates. The overall savings vary between \$8/flight at low operation rates to more than \$25/flight at higher operation rates for the baseline cost estimates.

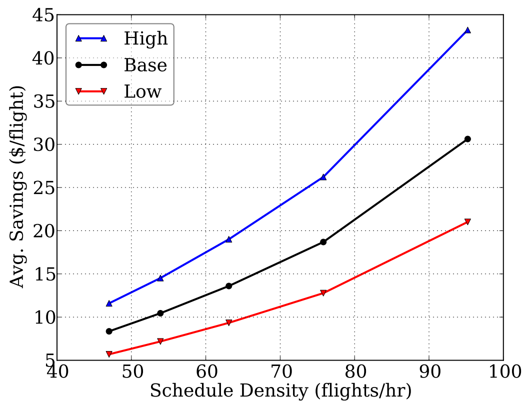


Figure 5: Average reduction in environmental costs when an optimization based schedule is used. The savings are over a FCFS policy.

These numbers can be used to estimate a lower bound on the annual environmental savings value for any major airport and for the entire air transportation system in the US during peak operation periods. To this end, we first estimate the annual number of commercial operations during a 2 hour peak period at 30 major airports based on the 2006 ASPM data shown in Table 3.4.1. Assuming an average savings value based on a peak operation rate of 40 flights per hour for two runways at each of these airports, we estimate the annual environmental savings based on the three different emission cost levels. These calculations suggest that annual savings in environmental costs can be between $9.4 and $19.1 million depending on the cost estimate level, if an optimization based scheduling policy is used during peak operation periods at major US airports.

Most environmental savings are realized due to the optimal scheduling of arriving aircraft. The main reason that arrivals produce most savings is because of the higher rates of fuel consumption and emissions during the descent phase of a flight. Indeed, despite the approximately equal rate of arrival and departure operations, the realized costs due to arrivals are 15 – 20 times higher than that of departures. On the other hand, this does not imply that the optimization results in significant increases in departure delays to enable decreases in arrival delays. Rather, the difference in costs implies that there is more potential for savings in environmental costs by optimally scheduling arrivals than departures.

Another issue involves the comparison of optimization procedures with and without

Table 11: Annual estimate of the environmental value of optimization for 30 major airports in the U.S.

Airport	Peak Period Operations	Savings in Environmental Costs due to Optimization		
		Low	Base	High
ATL	121,195	$690,088	$1,012,381	$1,405,256
ORD	120,113	$683,925	$1,003,339	$1,392,706
DFW	96,455	$549,216	$805,717	$1,118,392
DEN	78,942	$449,497	$659,427	$915,331
IAH	74,743	$425,589	$624,353	$866,646
LAX	67,784	$385,966	$566,224	$785,959
DTW	62,728	$357,172	$523,983	$727,325
CLT	59,219	$337,193	$494,673	$686,641
PHX	58,933	$335,567	$492,287	$683,329
PHL	58,015	$330,340	$484,619	$672,685
MSP	54,682	$311,358	$456,772	$634,032
EWR	53,134	$302,544	$443,841	$616,083
CVG	50,377	$286,849	$420,816	$584,123
JFK	48,985	$278,923	$409,189	$567,982
LGA	48,923	$278,567	$408,667	$567,259
BOS	48,787	$277,795	$407,535	$565,687
SLC	48,642	$276,969	$406,322	$564,003
LAS	46,320	$263,747	$386,925	$537,080
IAD	45,947	$261,622	$383,808	$532,752
MCO	43,808	$249,446	$365,945	$507,958
SEA	43,177	$245,852	$360,672	$500,639
MIA	39,679	$225,934	$331,453	$460,080
SFO	38,670	$220,188	$323,022	$448,378
DCA	36,457	$207,588	$304,538	$422,721
BWI	36,109	$205,607	$301,631	$418,685
CLE	34,998	$199,279	$292,349	$405,801
STL	34,937	$198,931	$291,838	$405,092
MDW	33,331	$189,789	$278,426	$386,475
MEM	30,608	$174,285	$255,681	$354,904
PDX	28,403	$161,726	$237,257	$329,330
TOTAL		$9,361,551	$13,733,692	$19,063,331

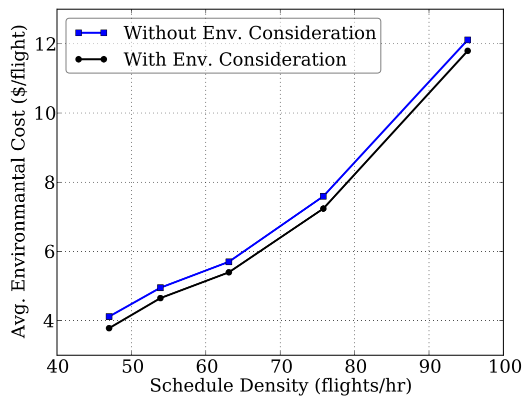


Figure 6: Environmental costs per flight based on schedules optimized with and without explicit consideration of the environmental factors. The values are according to the base emission cost estimate.

the environmental components. In other words, what is the value of considering environmental costs explicitly in an optimization model? Is minimizing fuel consumption and crew/passenger costs alone enough to minimize the environmental impact? To this end, we compare the environmental costs of optimal schedules in the two cases, and analyze the difference over the range of schedule densities considered. Explicit inclusion of environmental costs in the optimization reduces the environmental cost of the optimal schedule only minimally for each aircraft, i.e. around \$0.3/flight. This result holds for all schedule densities as shown in Figure 6.

This relatively high environmental value in fuel-optimal schedules is due to the direct relationship between fuel burn rates and emissions of aircraft. Hence, using optimal policies based on only the operational costs of airlines, which include fuel costs, will result in schedules that are very close to optimal schedules that account for environmental components. On the other hand, based on the number of peak period operations in Table 3.4.1, an environmental cost savings of \$0.3 per flight translates to about \$470,000 in annual savings on the environmental impact of airport operations. This amount can be viewed as a lower bound on the value of explicit consideration of environmental components in optimization based runway operations planning.

We also consider the relationship between environmental and non-environmental costs in

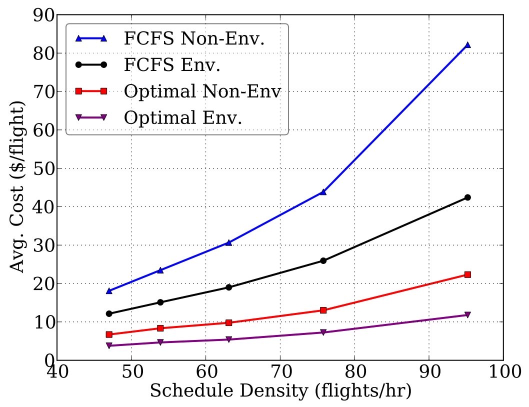


Figure 7: Analysis of environmental and non-environmental costs per flight for optimal and FCFS schedules.

optimized versus FCFS schedules, as seen in Figure 7. The increase in non-environmental costs in a FCFS schedule is evident at higher schedule densities. This may be relevant from an airline's perspective, as optimization not only reduces its overall relevant costs but the savings increase almost exponentially at higher operation rates. On the other hand, the increase in both environmental and non-environmental costs of optimization based schedules is mostly linear as operation rates rise.

3.4.2 Cost of Environmental Runway Scheduling to Airlines

Under current regulations environmental costs are mostly relevant as a cost to the society, rather than the airlines. If runway operations are scheduled based on optimization models that explicitly include environmental costs, it is likely that schedules will not be optimal from an operational cost perspective. Thus, an important question is the additional operational costs incurred by airlines under such environmental scheduling policies.

We compare the environmental and non-environmental costs of optimal schedules under different cost structures. In Figure 8, we show that the additional costs are minimal, and decrease significantly for heavier traffic volumes. More specifically, for baseline emission cost rates these additional costs vary between \$0.05 and \$0.25 per flight. Thus, based on the peak period operations considered in Table 3.4.1 the cumulative costs for airlines add up to be between \$78,000 and \$391,000 annually, if environmental factors are considered in the optimization of runway schedules.

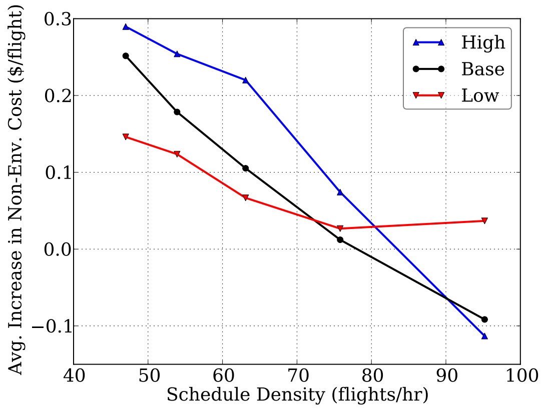


Figure 8: Cost of including the environmental impact in optimization.

3.4.3 Structure of Optimal Schedules

One measure in runway scheduling relates to the structure of the schedules. Given the optimal schedules generated with and without inclusion of the environmental, the issue is whether these schedules are actually implementable. We consider the number of position shifts in the optimal schedules when compared with the FCFS schedule and analyze the magnitude of these deviations to ensure that the schedules are implementable.

The first observation is that the general structure of the optimal schedule based on maximum position shifts is similar over different levels of environmental cost estimates. Hence, the environmental components do not lead to significantly different emphasis areas in the optimization. The average maximum position shifts over all optimization instances are shown in Figure 9. As expected, due to the differences in cost functions, the re-sequencing of the arriving aircraft is done in a more conservative way than the departures. Figure 9 shows that the maximum deviation in the arrival sequences is mostly less than six, even at the highest traffic volumes, while the maximum position shifts in the departures are typically higher at an average level of 10 over all schedule densities simulated.

The separation requirements used in the optimization models are defined according to the weight classes of aircraft. Hence, different fleet mixes may result in different patterns in the optimal schedules. While this is the case, a study of the fleet mixes during peak periods shows that the ratios of different weight classes are mostly similar at the major airports.

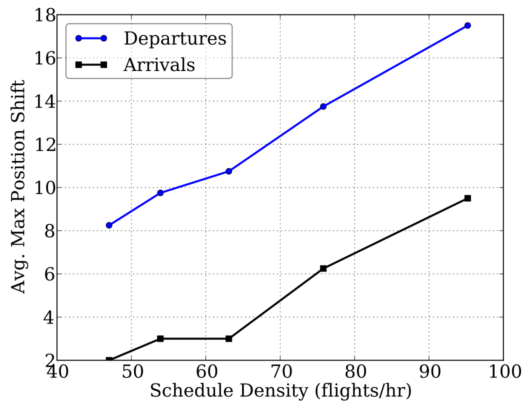


Figure 9: Maximum position shifts in the optimized schedule with respect to the FCFS schedule.

According to the ASPM data, the distribution at these airports is typically around 5% heavy, 7% Boeing 757, 80% large and 8%small class. These values are consistent with the distribution used in the simulations.

CHAPTER IV

TWO-STAGE STOCHASTIC RUNWAY SCHEDULING

In this chapter we develop and analyze a two-stage stochastic program for airport runway scheduling. To the best of our knowledge, this is the first formal treatment of the stochastic runway scheduling problem (SRSP). We begin the chapter by describing the two-stage decision process in Section 4.1 and provide our modeling framework. The close connection between the SRSP and certain machine scheduling problems is emphasized, and a restricted version of the full SRSP model is presented in Section 4.2 using machine scheduling notation. Characteristics attributed to the runway scheduling problem introduces additional complexities into the SRSP, and we discuss ways to handle these complexities in Section 4.3. The SRSP require the solution of a large scale stochastic program, and we develop a solution methodology based on scenario decomposition in which we incorporate efficient lower and upper bounding techniques. The solution methodology is presented in Section 4.4, and the setup for the computational analysis is discussed in Section 4.5. Taking advantage of problems with structures similar to our problem, two distinct formulations for the SRSP are developed and evaluated in Section 4.6, where we also analyze the practicality of the proposed solution methodologies.

4.1 Problem Description

In this section, we first describe the two-stage decision process that captures the stochasticity in scheduling runway operations. We then formally describe a generic modeling framework which we use to formulate a simpler version of the runway scheduling problem, as some limiting assumptions are required for SRSP to fit into this framework.

4.1.1 Two-Stage Scheduling Process

It is emphasized in Ernst, Krishnamoorthy, and Storer (1999) that the arrival scheduling problem at airports can be decomposed into two sequential problems. In this setup, only the sequence of operations is determined in the first stage, while the timing of each operation is decided in the second planning stage. The authors point out that given a feasible sequence, optimized arrival times can easily be determined through a linear program.

Focusing on departure operations, Anagnostakis and Clarke (2003) take the two-stage idea one step further by observing that the runway throughput only depends on the aircraft weight class sequence, and not on the characteristics of individual aircraft. Their two-stage scheduling approach is therefore slightly different, where the first stage determines an aircraft weight class sequence, and the second stage assigns individual aircraft to positions in the weight class sequence. In Anagnostakis (2004), the author elaborates on the advantages of this two-stage approach, although the problems in each stage are deterministic and are solved independently. Primarily, it is computationally efficient to decompose the departure scheduling problem to allow real time implementation. In addition, the two-stage planning process is appropriate for stochastic runway scheduling since the information required in the first planning stage, namely the pool of aircraft weight classes, is more robust than the scheduled time at runway for individual aircraft, which is the information needed in the second planning stage. To exemplify, an air traffic controller with a thirty minute look-ahead window is likely to predict the aircraft weight class mix better than the arrival and departure times for individual aircraft.

In this research we build upon the two-stage decision concept suggested by Anagnostakis and Clarke (2003). A similar decision disaggregation method is considered in Solveling, Solak, Clarke, and Johnson (2011a), where the authors use a stochastic programming model based on Benders' decomposition to find a good weight class sequence to use in the first planning stage. However, for the Benders' decomposition method to work, some simplifications and assumptions are made that do not capture all relevant information, e.g. the aircraft-to-position assignment in the second stage is based on static position times, and not on dynamically determined position times in the first stage. To overcome this draw-back, we suggest a two stage stochastic program that captures all the relevant information when determining an aircraft weight class sequence in the first planning step, which can be referred to as an exact formulation of SRSP.

4.1.2 A Generic Modeling Framework for a Simplified Version of SRSP

We now describe a generic framework which involves some simplifications over the exact definition of SRSP. We refer to this version of the problem as the restricted SRSP (SRSP-R). To emphasis the connection to machine scheduling, we formulate the restricted SRSP using machine scheduling notation and indicate the mapping to the airport runway scheduling problem. Additional complexities inherent to airport runway scheduling are presented in the subsequent section and the notation is adjusted accordingly.

Let $\mathcal{I}$ be the set of jobs (we use $\mathcal{A}$ when referring to aircraft) that can be scheduled in a given planning horizon. Furthermore, let $\mathcal{K}$ be the set of job categories (aircraft weight classes) for the jobs in $\mathcal{I}$. The positions in the sequence are represented by the set $\mathcal{P} = \{1, \ldots, n\}$, where $n = |\mathcal{I}|$. For ease of notation, let $\mathcal{I}_k \subseteq \mathcal{I}$ ($\mathcal{A}_k \subseteq \mathcal{A}$) be the set of jobs that are of category $k \in \mathcal{K}$ with $n_k = |\mathcal{I}_k|$ and let $h_i \in \mathcal{K}$ correspond to the job category of job i. During the planning stage, each job in $\mathcal{I}$ has a release time r_i. In airport runway scheduling, the release times correspond to the estimated time of arrival at the runway for arrivals (time at runway threshold) and the estimated runway departure time (beginning of takeoff roll) for departures. When the direction is irrelevant, we simply refer to this as the *runway time*.

Sequence-dependent setup times are given by parameters s_{k_1,k_2}, where $k_1 \in \mathcal{K}$ is the job category of the leading job and $k_2 \in \mathcal{K}$ is the job category of the trailing job. In the runway scheduling framework, we refer to the sequence-dependent setup times as separation requirements. When a parameter s is indexed by elements $i \in \mathcal{I}$, we implicitly mean the corresponding job category of job $i \in \mathcal{I}$. In this application we use four aircraft weight classes which represent the types of aircraft operating at most major airports: Heavy (H), Boeing 757 (7), Large (L), and Small (S). Because we consider both arrivals and departures, we let the elements of $\mathcal{K}$ include the type of operation as well, e.g. $AH \in \mathcal{K}$ denotes the arrival of an aircraft in the "heavy" weight class, whereas $DL \in \mathcal{K}$ denotes the departure operation for a "large" aircraft. The separation requirements for these weight classes when operating on a single runway are specified by Federal Aviation Administration (2010a), and are shown in Tables 12 and 13. At airports with close parallel runways there are additional

Table 12: Same runway separation requirements for arrival operations.

		Trailing			
		H	7	L	S
Leading	H	4	5	5	6
	7	4	4	4	5
	L	3	3	3	4
	S	3	3	3	3

(a) Separation requirements, in nautical miles, at the runway threshold.

		Trailing			
		H	7	L	S
Leading	H	96	138	138	240
	7	96	108	108	198
	L	60	72	72	162
	S	60	72	72	102

(b) Separation requirements, in seconds, at the runway threshold.

Table 13: Same runway separation requirements, in seconds, for departure operations.

		Trailing			
		H	7	L	S
Leading	H	90	90	120	120
	7	90	90	120	120
	L	60	60	60	60
	S	60	60	60	60

separation requirements defined between operations on separate runways, and these are shown in Table 14.

Now consider the following two-fold objective. The first goal is to find a sequence that minimizes the sum of setup times, i.e. the makespan of completing the n jobs. On the other hand, the second goal is to process the jobs as early as possible after their release times, which translates into determining a sequence of job categories such that the sum of all the delay imposed on the jobs is minimized. Letting $\mathcal{X}$ be the set of all job category sequences,

Table 14: Separation requirements, in seconds, for operations on close parallel runways. The leading operation is performed on one runway and the trailing operation is on the parallel runway.

		Trailing			
		H	7	L	S
Leading	H	68	68	68	80
	7	68	68	68	80
	L	62	62	62	80
	S	48	55	55	80

(a) Leading arrival followed by departure.

		Trailing			
		H	7	L	S
Leading	H	54	58	58	80
	7	54	58	58	80
	L	54	58	58	80
	S	54	58	58	80

(b) Leading departure followed by arrival.

we would like to determine the value $\min_{\mathbf{x}\in\mathcal{X}} c(\mathbf{x})$, where $c(\mathbf{x}) = \lambda c^1(\mathbf{x}) + (1-\lambda)c^2(\mathbf{x})$. Here c^1 captures the cost of the length of the sequence and c^2 captures the cost of job delay. The parameter $\lambda \in [0,1]$ sets the relative weight of the objectives.

Each element $\mathbf{x} \in \mathcal{X}$ can be described as $\mathbf{x} = (k_{(1)}, \ldots, k_{(n)})$, where $k_{(p)} \in \mathcal{K}$ is the job category in position p. We also let $k_{(0)}$ denote the job category that precedes $\mathbf{x}$. In addition, if we let t_i denote the start time for job i, we can define the job delay as $\Delta_i = t_i - r_r$. With this notation we can express the overall objective function as:

$$c(x) = \lambda c^1(x) + (1-\lambda)c^2(x) = \lambda \sum_{p=0}^{n-1} s_{k_{(p)},k_{(p+1)}} + (1-\lambda)\sum_{i\in\mathcal{I}} \Delta_i \tag{12}$$

To model the uncertainty in the problem framework, we assume that the release times $r_i,\ i \in \mathcal{I}$, are not known with certainty. Furthermore, for tractability and to enable a stochastic programming approach, we assume that the probability distribution describing $r_i\, i \in \mathcal{I}$ is discrete with finite support. Let $\boldsymbol{\xi}$ denote the vector containing stochastic input to the problem, where $\boldsymbol{\xi}$ can be realized into a finite number of scenarios. Let Ω be the set of all scenarios, where each scenario $\omega \in \Omega$ has probability ϱ_ω and a corresponding vector $\boldsymbol{\xi}_\omega = (r_1^\omega, \ldots, r_n^\omega)$ with r_i^ω denoting the release time of job $i \in \mathcal{I}$ in scenario ω.

The objective of the stochastic optimization problem is to find a sequence $\mathbf{x} \in \mathcal{X}$ that minimizes the expected value $\mathbb{E}_\xi[c(\mathbf{x}, \boldsymbol{\xi})]$, where the function $c(\mathbf{x})$ is extended to include the dependency on the probabilistic vector $\boldsymbol{\xi}$. More specifically, we have $c(\mathbf{x}, \boldsymbol{\xi}) = \lambda c^1(\mathbf{x}) + (1-\lambda)c^2(\mathbf{x}, \boldsymbol{\xi})$, and the overall objective is:

$$\nu = \min_{\mathbf{x}\in\mathcal{X}} \mathbb{E}_\xi[c(\mathbf{x}, \boldsymbol{\xi})] \tag{13}$$

Given the described structure of the scheduling problem, the task of finding ν can be modeled as a two-stage stochastic program. In the first stage a sequence of job categories is identified, and in the second stage specific jobs are assigned to positions in the sequence. Naturally, a job in the second stage can only be assigned to positions of the corresponding job category.

4.2 Two Alternative Formulations for SRSP-R and SRSP

In this section, we develop two mathematical formulations that are initially used to model the restricted version of the stochastic runway scheduling problem, i.e. SRSP-R, and are then extended to capture the additional complexities in SRSP.

4.2.1 Network Formulation of SRSP-R

Given the general modeling framework, a network flow model can be constructed to capture the two-stage decision process described in Section 4.1. We first describe the corresponding structures for the two stages separately, and then summarize the overall network flow-based stochastic programming formulation for SRSP-R.

The first stage problem in SRSP-R consists of finding a sequence of job categories that minimizes the makespan of completing all the jobs. This structure can be modeled as a TSP as follows. Consider a graph where each job is represented as a node with arcs connecting each pair of nodes, and let the arc costs define setup times between jobs. Note that this construction creates a node for each job rather than each category of job. It is easy to show that the optimal TSP tour on this network would correspond to a sequence of job categories that minimizes the makespan. Once the actual ready times become known, the second stage problem involves assigning individual jobs to positions in the sequence, which requires explicit representation of the position information based on the solution of the first stage problem. However, a drawback of the traditional TSP formulation described above is that the position of a node on a TSP tour cannot be determined without examining the complete sequence. Thus, it is not possible to link the first stage and second stage problems when a standard TSP formulation is used to represent the first stage problem.

Given the need for the position information in the second stage, we model the first stage problem in SRSP-R as a position dependent TSP, which is also known as time dependent TSP. The time dependent TSP has been used for machine scheduling problems on a single machine with sequence-dependent setup times (Bigras, Gamache, and Savard 2008; Picard and Queyranne 1978). In contrast to the traditional TSP, the cost of traversing an arc (or the setup time in the context of machine scheduling) depends on the vertices (jobs)

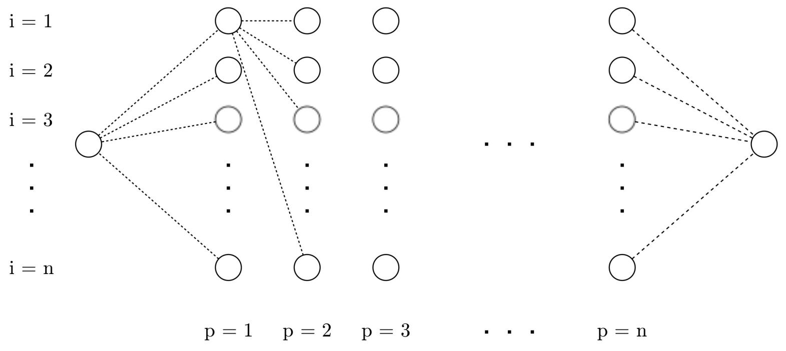


Figure 10: Network representation for the flow based formulation

it connects *and* the position in the sequence. Thus, when using such formulation, the position of a specific node is explicitly captured, which makes it possible to extract the job category for that position. To this end, we define the graph $G = (\mathcal{V}, \mathcal{E})$ with the set of vertices $\mathcal{V} = \{v_{i,p} : i = 1, \ldots, n; p = 1, \ldots, n\} \cup \{v_0, v_{n+1}\}$, where a vertex for each job i and position p is included, and where v_0 and v_{n+1} represent dummy source and sink nodes, respectively. The set of arcs $\mathcal{E}$ is defined as $\mathcal{E} = \{(v_{i,p}, v_{j,p+1}) : i = 1, \ldots, n; j = 1, \ldots, n; p = 1, \ldots, n-1\} \cup \{(v_0, v_{i,1}) : i = 1, \ldots, n\} \cup \{(v_{i,n}, v_{n+1}) : i = 1, \ldots, n\}$. A visual representation of the graph G is shown in Figure 10. The setup times $s_{i,j}$ are defined between categories of jobs, i.e. $s_{i,j} = s_{h_i,h_j}$, where h_i indicates the category of job i.

To formulate the problem, we also define the following variables:

$x_{p,k}$:	1 if job category k is assigned to position p, 0 otherwise
ℓ_p :	time of position p
$z_{i,j}^{p,\omega}$:	1 if there is flow on arc $(v_{i,p}, v_{j,p+1})$ in scenario ω, 0 otherwise
$y_{i,p}^{\omega}$:	1 if job i is assigned to position p in scenario ω, 0 otherwise
σ_p^{ω} :	start time of the job assigned to position p in scenario ω

Given this notation, a network flow-based formulation for SRSP-R can be described as follows.

$$\min \quad \mathbb{E}_{\xi}\left[\lambda c^1(\mathbf{x}) + (1-\lambda) c^2(\mathbf{x}, \boldsymbol{\xi})\right] = \lambda \ell_n + (1-\lambda) \sum_{\omega \in \Omega} \varrho^{\omega} \left(\sum_{p=1}^{n} \sigma_p^{\omega} - C^{\omega} \right) \tag{14}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} z_j^{0,\omega} = 1 \qquad \forall \omega \in \Omega \tag{15}$$

$$z_j^{0,\omega} = \sum_{i=1}^{n} z_{j,i}^{1,\omega} \qquad j = 1, \ldots, n, \ \forall \omega \in \Omega \tag{16}$$

$$\sum_{i=1}^{n} z_{i,j}^{p,\omega} = \sum_{i=1}^{n} z_{j,i}^{p+1,\omega} \qquad p = 1, \ldots, n-2 \quad j = 1, \ldots, n, \ \forall \omega \in \Omega \tag{17}$$

$$\sum_{i=1}^{n} z_{i,j}^{n-1,\omega} = z_j^{n,\omega} \qquad j = 1, \ldots, n, \ \forall \omega \in \Omega \tag{18}$$

$$z_j^{0,\omega} + \sum_{i=1}^{n} \sum_{p=1}^{n-1} z_{i,j}^{p,\omega} = 1 \qquad j = 1, \ldots, n, \ \forall \omega \in \Omega \tag{19}$$

$$\sum_{p \in P} y_{i,p}^{\omega} = 1 \qquad \forall i \in \mathcal{I}, \ \omega \in \Omega \tag{20}$$

$$\sigma_p^{\omega} \geq \sigma_{p-1}^{\omega} + \sum_{i=1}^{n} \sum_{j=1}^{n} s_{i,j} z_{i,j}^{p-1,\omega} \qquad p = 2, \ldots, n, \ \forall \omega \in \Omega \tag{21}$$

$$\sigma_1^{\omega} \geq \sum_{i=1}^{n} s_{0,i} z_i^{0,\omega} \qquad \forall \omega \in \Omega \tag{22}$$

$$\sigma_p^{\omega} \geq r_i^{\omega} \sum_{j=1}^{n} z_{i,j}^{p,\omega} \qquad \forall i \in \mathcal{I}, \ p \in \mathcal{P}, \ \omega \in \Omega \tag{23}$$

$$y_{i,p}^{\omega} = \sum_{j=1}^{n} z_{i,j}^{p,\omega} \qquad \forall i \in \mathcal{I}, \ p \in \mathcal{P}, \ \omega \in \Omega \tag{24}$$

$$y_{i,p+1}^{\omega} = \sum_{j=1}^{n} z_{j,i}^{p,\omega} \qquad \forall i \in \mathcal{I}, \ p \in \mathcal{P}, \ \omega \in \Omega \tag{25}$$

$$x_{p,k} = \sum_{i \in \mathcal{I}: h_i = k} \sum_{j=1}^{n} z_{i,j}^{p,\omega} \qquad p = 1 \ldots, n, \ \forall k \in \mathcal{K}, \ \omega \in \Omega \tag{26}$$

$$x_{n,k} = \sum_{i \in \mathcal{I}: h_i = k} z_i^{n,\omega} \qquad \forall k \in \mathcal{K}, \ \omega \in \Omega \tag{27}$$

$$\ell_p \geq \ell_{p-1} + \sum_{i=1}^{n} \sum_{j=1}^{n} s_{i,j} z_{i,j}^{p-1,\omega} \qquad p = 2, \ldots, n, \ \forall \omega \in \Omega \tag{28}$$

$$\ell_1 \geq s_{0,j} z_j^{0,\omega} \qquad j = 1, \ldots, n, \ \forall \omega \in \Omega \tag{29}$$

$$x, z, \ell, \sigma \geq 0; y \in \{0, 1\} \tag{30}$$

where $z_j^{0,\omega}$ and $z_j^{n,\omega}$ are the flow variables for the arcs leading out of the dummy source and sink nodes, and $C^{\omega} = \sum_{i \in \mathcal{I}} r_i^{\omega}$ is a constant. The objective function (14) corresponds to a weighted sum of the first stage objective and the expectation of objective function value in the second stage. In Proposition 1 we show that this objective function is equivalent to (12).

In the formulation above, constraints (15)-(25) correspond to the second stage decisions. More specifically, constraint (15) forces flow out of the source node, while constraints (16) to (18) ensure conservation of flow at each node in the network for each scenario. Using only these four sets of constraints, a flow from the source node to the sink node can be described. However, we also need to make sure that each job is traversed exactly once. This is done in constraint (19), which destroys the totally unimodular structure of the constraint matrix. The equality (20) is a second stage assignment constraint. Constraints (21) and (22) allow for capturing the time of each position, and constraint (23) ensures that no job can be scheduled before its earliest time r_i^ω. The flow variables are linked to the assignment variables through constraints (24) and (25).

The remaining constraints (26)-(29) define the first stage variables in terms of the second stage decisions for each scenario. In other words, they represent the nonanticipativity requirements in the stochastic programming problem. Note that with constraint (19) in place there is a one-to-one correspondence between the flow variables and a job category sequence $\mathbf{x}$. Hence, constraints (26) and (27) allow the determination of the corresponding sequence for a given flow. The time of position p, i.e. ℓ_p is defined by constraints (28) and (29).

Finally, the constraints in (30) are the nonnegativity and integrality constraints. Since the y-variables are defined to be binary, we can drop the integrality constraints for the flow variables $z_{i,j}^{p,\omega}$, as it can be shown that $z_{i,j}^{p,\omega} = y_{i,p}^\omega y_{j,p+1}^\omega$ for scenario ω. Note that it is possible to eliminate the assignment variables $y_{i,p}^\omega$ altogether, as the job-to-position assignment is captured in the flow variables. In that case, however, the flow variables must be declared as binary, and constraints (20) must be expressed in terms of the flow variables. Of the two options, we use the former as Heilporn, Cordeau, and Laporte (2010a) report that the formulation with fewer, i.e. $\mathcal{O}(n^2)$, binary variables has better computational performance for similarly structured problems, which we verify to hold for our problem as well.

We now show that the formulation used in objective function (14) corresponds to the function defined by equation (12):

Proposition 1. *The function given by* (14), $\lambda\ell_n + (1-\lambda)\sum_{\omega\in\Omega}\varrho^\omega\left(\sum_{p=1}^n \sigma_p^\omega - C^\omega\right)$, *is equivalent to the objective function defined by* (12) *and* (13), *adjusted to consider a finite set of scenarios corresponding to stochastic parameter realizations.*

Proof. First we show that $\lambda\ell_n = \lambda\sum_{p=1}^{n-1} s_{k_{(p)},k_{(p+1)}}$. If $\lambda = 0$, the result is trivial. When $\lambda > 0$, constraints (28) and (29) hold at equality as we are minimizing ℓ_n and no other constraints interfere with the ℓ-variables. Furthermore, due to the coupling constraints (26) and (27), the source-to-sink path defined by the z-variables follows the same sequence of job categories over all scenarios. Since setup times are defined based on job categories, we have $\sum_{i=1}^n\sum_{j=1}^n s_{i,j}z_{i,j}^{p-1,\omega} = \sum_{i=1}^n\sum_{j=1}^n s_{i,j}z_{i,j}^{p-1,\omega'}$, $\forall\omega,\omega'\in\Omega$. Thus,

$$\begin{aligned}\ell_n =&\ell_{n-1} + \sum_{i=1}^n\sum_{j=1}^n s_{i,j}z_{i,j}^{p-1} = \cdots = \ell_1 + \sum_{p=1}^{n-1}\sum_{i=1}^n\sum_{j=1}^n s_{i,j}z_{i,j}^p = \sum_{p=0}^{n-1}\sum_{i=1}^n\sum_{j=1}^n s_{i,j}z_{i,j}^p = \\ =&\sum_{p=0}^{n-1} s_{k_{(p)},k_{(p+1)}}\end{aligned} \tag{31}$$

where $\mathbf{x} = (k_{(1)},\ldots,k_{(n)})$ is the resulting job category sequence.

For the second component, we have for each ω that,

$$\sum_{i\in\mathcal{I}}\Delta_i^\omega = \sum_{i=1}^n t_i^\omega - r_i^\omega = \sum_{i=1}^n\sum_{p=1}^n \sigma_p^\omega y_{i,p}^\omega - C^\omega = \sum_{p=1}^n\sigma_p^\omega\sum_{i=1}^n y_{i,p}^\omega - C^\omega = \sum_{p=1}^n\sigma_p^\omega - C^\omega \tag{32}$$

In (32) we use $\sum_{i=1}^n y_{i,p}^\omega = \sum_{i=1}^n\sum_{j=1}^n z_{i,j}^{p,\omega} = 1$ for all $\omega\in\Omega$. □

4.2.2 Slot Formulation of SRSP-R

Instead of formulating the problem based on the network structure depicted in Figure 10 and the associated flow variables z, we define another model solely based on the assignment variables $x_{p,k}$ for the first stage and $y_{i,p}^\omega$ for the second stage, where both the x and y variables are declared as binary. We refer to this second formulation as the *slot formulation.* We later investigate and compare the relative efficiencies of the two formulations.

In contrast to the flow based formulation where a feasible flow gives the weight class to slot assignments, we explicitly model the assignment decisions in the slot based representation. The resulting formulation is more compact and can be described as follows:

$$\min \quad (14)$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{p \in \mathcal{P}} x_{p,k} = n_k && \forall k \in \mathcal{K} && (33)\\
& \sum_{k \in \mathcal{K}} x_{p,k} = 1 && \forall p \in \mathcal{P} && (34)\\
& \ell_{p+1} - \ell_p \geq s_{k_1,k_2}(x_{p,k_1} + x_{p+1,k_2} - 1) && \forall p \in \mathcal{P},\ \forall k_1, k_2 \in \mathcal{K} && (35)\\
& \ell_1 \geq s_{k_0,k_1} x_{1,k_1} && \forall k_1 \in \mathcal{K} && (36)\\
& \sum_{i \in \mathcal{I}_k} y^{\omega}_{i,p} = x_{p,k} && \forall p \in \mathcal{P},\ k \in \mathcal{K},\ \omega \in \Omega && (37)\\
& \sigma^{\omega}_{p+1} - \sigma^{\omega}_{p} \geq \ell_{p+1} - \ell_p && \forall p \in \mathcal{P},\ \omega \in \Omega && (38)\\
& \sigma^{\omega}_{1} \geq \ell_1 && \omega \in \Omega && (39)\\
& \sigma^{\omega}_{p} \geq r^{\omega}_i y^{\omega}_{i,p} && \forall i \in \mathcal{I},\ p \in \mathcal{P},\ \omega \in \Omega && (40)\\
& (20) && && \\
& \ell, \sigma \geq 0; x, y \in \{0,1\} && && (41)
\end{aligned}
$$

In this formulation, constraint (33) requires the correct number of job categories to be allocated to the positions in the sequence, while (34) ensures that only one category is assigned to each position. The setup times can be modeled according to constraint sets (35) and (36). If a job category k_1 is assigned to position p and a job category k_2 is assigned to position $p+1$, the expression within parenthesis will take value one, effectively setting the setup time between position ℓ_{p+1} and position ℓ_p to be at least s_{k_1,k_2} units.

The possible job assignments are given by the first stage variable $\mathbf{x}$, and the second stage assignment variables are linked to the first stage through constraint set (37). Setup times between jobs are enforced in constraint sets (38) and (39), taking advantage of the already calculated setup-time between positions and the fact that the ℓ variables give a "tight" sequence in an optimal solution. Constraints (37), (38), and (39) form the nonanticipativity constraints for the problem. Constraint (40) ensures that no job can be scheduled before its earliest time r^{ω}_i. Finally, we require each job to be assigned to exactly one position in the sequence in the second stage, which is enforced by constraint (20) as in the network formulation.

4.2.3 Distinctions of SRSP

As noted above, the restricted framework can be applied to scheduling problems where the release times are stochastic and the objective is a trade-off between system performance times and entity wait-times. However, the stochastic runway scheduling problem contains additional components that are not captured in this generic representation. These complexities also highlight the distinctions between classical scheduling problems and SRSP, and can be described as follows:

Early arrivals. The estimated time at the runway for aircraft are assumed to be stochastic and defined by a known probability distribution. The uncertainty in estimated runway times is caused by factors such as aircraft equipage level, pilot behavior and weather for arrivals, and pushback delay and surface congestion for departures. In the restricted model it is assumed that the release time in each scenario, i.e. r_i^ω, coincides with the earliest time a job can be scheduled, implying that a job cannot be scheduled to a time earlier than its release time. While this is a valid assumption for runway departure scheduling, in practice it is possible that an arriving aircraft is asked to increase speed to land early. We explicitly model this in SRSP, which adds to the computational complexity of the problem as the number of feasible assignments increase.

Violated triangle inequalities. When arrival runway scheduling is performed in isolation, i.e. when only arrival operations are considered, the separation requirements do not violate the triangle inequality. For given weight classes i, j, and k, the triangle inequality is:

$$s_{i,k} \leq s_{i,j} + s_{j,k} \tag{42}$$

The implication of this fact is that all separation requirements can be met by considering separation between consecutive operations only. The same holds for pure departure scheduling. However, in mixed operations on the same runway or on close parallel runways (even if operated as dedicated arrival and departure runways) the triangle inequality does not always hold. This issue needs to be addressed in the formulation of SRSP.

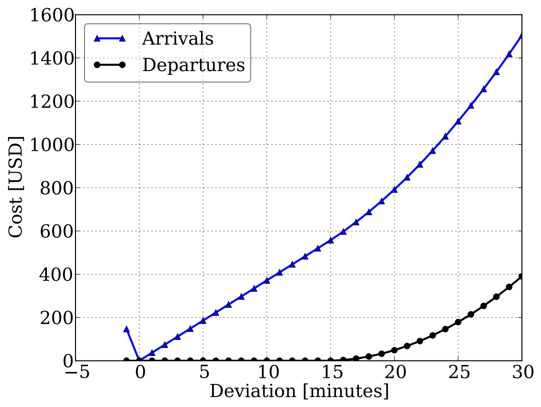


Figure 11: Sample arrival and departure delay cost functions for Boeing 737, representing the Large aircraft weight class.

Nonlinear cost functions. A complicating factor for SRSP is that the cost of delays for aircraft that have to wait before using the runway is typically not linear. In Chapter 3, global cost functions that capture all relevant costs for runway scheduling are developed for different aircraft weight classes. These cost functions account for the impact of delays on airline fuel, crew and passenger costs, as well as on environmental costs due to increased emissions. Sample departure and arrival delay cost functions for Boeing 737 representing the Large aircraft weight class are shown in Figure 11. We note that the remaining flight time determines the maximum time an arrival aircraft can be early. In the example shown we consider arrivals with 30 minutes remaining flying time before reaching the airport. Note that there is no cost for departure delays shorter than 15 minutes as commonly assumed in the literature. For further details on how the cost functions are developed, the reader is referred to Solveling, Solak, Clarke, and Johnson (2011b).

Capturing runway times for cost calculations. In sequencing models that consider an ordering of operations and a cost measure between consecutive operations, such as machine scheduling problems or the time dependent TSP, it is generally easy to obtain the total cost from the beginning of the sequence up to a position p. For example, in machine scheduling problems with sequence-dependent setup times we can add all processing times, setup times and idle times of all jobs processed in positions 1 to $p-1$ to obtain the start time for job

p. However, in SRSP distinct cost functions based on individual aircraft (job) delay are used as part of the objective function. Hence, it is necessary to capture the start time for each job. From a modeling perspective, getting the start time of a particular job is not that straight forward, as the position to which the particular job is assigned must be identified. This results in non-linear models and increased complexity.

4.2.4 Extension of the Two Formulations to SRSP

In this section, we expand the flow and slot formulations of SRSP-R to model the actual airport runway scheduling problem, i.e. SRSP. In the representation of the SRSP we let aircraft take the role of jobs and aircraft weight classes replace job categories. We consider each complexity in SRSP separately and discuss how to integrate them into the two formulations.

Modeling early arrivals. In order to model the possibility of early arrivals for scheduling, we define o_a as the largest time by which an aircraft a can be early. We assume that departure aircraft cannot be scheduled early (i.e. $o_a = 0$, when $a \in Dep$ is a departure), while for arrivals the value of o_a is determined by the remaining flight time to the runway. Hence, we define $\hat{r}_a^\omega$ as the earliest time aircraft a can be scheduled and set $\hat{r}_a^\omega = r_a^\omega - o_f$ and replace r_a^ω with $\hat{r}_a^\omega$ in constraints (23) and (40).

Modeling violated triangle inequalities. Violated triangle inequalities defined in Equation (42) constitute a characteristic that clearly distinguishes the general runway scheduling problem from the single machine scheduling problem with sequence-dependent setup times. When the triangle inequality is satisfied between all pairs of operations, it is sufficient to consider separation between consecutive operations, which reduces the complexity of the problem. However, the issue becomes relevant when combined arrival and departure operations are considered.

There exist some proposed approaches in dealing with this complexity. By formulating the deterministic runway scheduling problem through a different form, the issue of violated triangle inequalities can be eliminated. Beasley, Krishnamoorthy, Sharaiha, and Abramson

(2000) suggest defining the time of operation for aircraft a directly and including all pairwise separation constraints. Due to the asymmetric nature of the separation requirements, this formulation needs additional binary decision variables indicating if aircraft i precedes aircraft j. A disadvantage with this formulation, which excludes it from being considered in this thesis, is that we only get the time of operation and not the position in the sequence. A hybrid approach to overcome violated triangle inequalities is also developed in Sherali, Ghoniem, Baik, and Trani (2012), where the authors use consecutive position separations and include pairwise separation constraints for those aircraft that may potentially violate the triangle inequality.

In our formulation of the SRSP we do not explicitly consider violated triangle inequalities. This is because of the implicit handling of this issue in any potential practical implementation of the stochastic runway scheduling algorithm. In practice, the aircraft weight class sequence obtained from the solution of SRSP can be used by air traffic controllers to sequence arriving and departing aircraft. As uncertainty is realized, a simpler deterministic model, similar to our second stage model, is solved to set the final aircraft sequence and enforce all separation constraints. Violated triangle inequalities would be handled in this simpler model, either through explicit inclusion as in Sherali, Ghoniem, Baik, and Trani (2012), or through post processing where aircraft are delayed in case of such a violation in the optimal solution.

Modeling nonlinear cost functions. For the nonlinear cost functions, we note that the functions are convex, and use standard piecewise linearization techniques to model the cost components. To that end, let $\mathcal{M}_a$ denote the set of linear segments defining the cost function for aircraft $a \in \mathcal{A}$. The slope and intercept of each element $m \in \mathcal{M}_a$ are given by γ_m^a and ζ_m^a, respectively. Similarly, the cost function for runway utilization is defined by the slope and intersect γ_m and ζ_m with linear segments m as elements of $\mathcal{M}$. Clearly, these piecewise linear representations result in increases in problem size for SRSP as described below.

To model the cost functions, we define the continuous variables $\Delta_{a,\omega}^{+}$ and $\Delta_{a,\omega}^{-}$, for the positive (delay) and negative (early) deviations in each scenario respectively. The cost of deviation for aircraft a is captured in the continuous variables $\rho^+_{a,\omega}$ and $\rho^-_{a,\omega}$. Similarly, we define the continuous variable ρ to capture the cost of runway utilization in the first stage. The objective function (14) in SRSP is then defined as:

$$\min \quad \mathbb{E}_\xi \left[\lambda c^1(\mathbf{x}) + (1-\lambda)c^2(\mathbf{x}, \boldsymbol{\xi})\right] = \rho + \sum_{\omega\in\Omega} \varrho^\omega \left(\sum_{a\in\mathcal{A}} \rho^+_{a,\omega} + \rho^-_{a,\omega}\right) \tag{43}$$

Note that the relative weights between the two objectives, which was earlier parameterized by λ, is now eliminated. Instead, the definition of the cost functions gives the absolute weight for each component.

The first stage runway utilization cost ρ is determined by the time of the latest position ℓ_n through the following constraints:

$$\rho \geq \ell_n \gamma_m + \zeta_m \qquad \forall m \in \mathcal{M} \tag{44}$$

Similarly, in the second stage, the flight deviation costs are expressed in terms of the deviation variables Δ as depicted in constraints (45) and (46) below:

$$\rho^+_{a,\omega} \geq \Delta^+_{a,\omega}\gamma^a_m + \zeta^a_m \qquad a \in \mathcal{A},\ \omega \in \Omega,\ m \in \mathcal{M}^a : \gamma^a_m \geq 0 \tag{45}$$

$$\rho^-_{a,\omega} \geq (-\Delta^-_{a,\omega})\gamma^a_m + \zeta^a_m \qquad a \in \mathcal{A},\ \omega \in \Omega,\ m \in \mathcal{M}^a : \gamma^a_m < 0 \tag{46}$$

The positive and negative deviations are calculated through constraint (47) as follows:

$$\Delta^+_{a,\omega} - \Delta^-_{a,\omega} = t^\omega_a - r^\omega_a \qquad a \in \mathcal{A} \tag{47}$$

Modeling the capturing of runway times for cost calculations. On capturing the runway times, we note that with the non-linear cost functions in place we cannot use Equation (32) directly to capture the sum of flight times, i.e. $\sum_{a=1}^n t^\omega_a$, from the sum of position times, i.e. $\sum_{p=1}^n \sigma^\omega_p$. Thus, we need to explicitly model the relationship between the position times σ^ω_p and the flight times t^ω_a in SRSP. This can be done using constraints based on the standard big-M type constructs as follows:

$$t^\omega_a \geq \sigma^\omega_p - M_{a,p}(1 - y^\omega_{a,p}) \qquad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \omega \in \Omega \tag{48}$$

$$t_a^{\omega} \leq \sigma_p^{\omega} + M_{a,p}(1 - y_{a,p}^{\omega}) \qquad \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \omega \in \Omega \tag{49}$$

However, these big-M constraints add significant computational complexity to the network model preventing the exploitation of any flow structure. In Section 4.3, we develop improved formulations for these constraints by defining some valid inequalities.

With these extensions, both the network and slot formulations defined for the restricted problem can be used to represent the true stochastic runway scheduling problem. However, these complexities require that the efficiencies of the models are improved for tractability. In the next section we describe some improvement procedures to help in that regard.

4.3 Valid Inequalities for SRSP

We improve the modeling of SRSP by tightening the formulations through more efficient representations of the big-M constructs used in the model. More specifically, we note that constraints (48) and (49) involve the constants $M_{a,p}$ to model the link between slot times and flight times. As the strength of the LP-relaxation depends on the value of $M_{a,p}$ we are interested in either minimizing $M_{a,p}$ (and yet keeping it sufficiently large) or reformulating the constraint and making it as strong as possible. To this end, we derive some tight valid inequalities to be applied to each scenario problem in the two formulations we develop. However, for the clarity of presentation, we omit the index ω when referring to the stochastic parameters and the second stage variables in this section.

First, we assume that the aircraft are ordered by their estimated time at the runway, i.e. $r_{a_1} \leq r_{a_2} \leq \cdots \leq r_{a_n}$. Recall that r_a is the realized runway time for aircraft a and that $\hat{r}_a \leq r_a$. Furthermore, the positions are ordered by time, so that $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_n$, which is modeled through the separation constraint (21). We also have $\sum_{p=1}^{n} y_{a,p} = 1$ for all $a \in \mathcal{A}$.

To derive the valid inequalities, we define $\Psi(p, a)$ as the longest possible sequence with respect to separation requirements, to position $p \geq 2$ when the weight class of aircraft a is not a part of the sequence. Similarly, we define $\Phi(\kappa, a)$ to be the earliest time possible for position κ, based on the shortest sequence to κ given that aircraft a is assigned to that position. Now consider the following reformulations of constraints (48) and (49), where it can be observed that if aircraft a is assigned to position p, i.e. $y_{a,p} = 1$, then $t_a = \sigma_p$.

$$t_a \geq \sigma_p - \sum_{\kappa=1}^{p-1} u_{p,a,\kappa} y_{a,\kappa} + \sum_{\kappa=p+1}^{n} \mu_{p,a,\kappa} y_{a,\kappa} \qquad \forall p \in \mathcal{P}, a \in \mathcal{A} \tag{50}$$

$$t_a \leq \sigma_p - \sum_{\kappa=1}^{p-1} \bar{\mu}_{p,a,k} y_{a,\kappa} + \sum_{\kappa=p+1}^{n} \bar{u}_{p,a,\kappa} y_{a,\kappa} \qquad \forall p \in \mathcal{P}, a \in \mathcal{A} \tag{51}$$

Using the notation above, we can state the following result:

Proposition 2. *For $u_{p,a,\kappa} = \Psi(p - \kappa, a) + r_n$ and $\mu_{p,a,\kappa} = \Phi(\kappa - p + 1, a)$, constraint* (50) *is a valid inequality for SRSP.*

Proof. To simplify the proof we make two non-restrictive assumptions:

1. The latest scheduled time, r_n, is larger than the maximum separation requirement, i.e. $r_n \geq s_{k_1,k_2}$ for all $k_1, k_2 \in \mathcal{K} \times \mathcal{K}$.
2. If an aircraft, a, is scheduled after the estimated runway time, in which case $\Delta_a^+ > 0$, a smaller delay is always favorable. If the convex cost function, denoted by c_a, is non-zero, this is natural as $c_a(\Delta_{a,2}^+) > c_a(\Delta_{a,1}^+)$ when $\Delta_{a,2}^+ > \Delta_{a,1}^+$. In the trivial case where the function is zero, we assume that $\Delta_{a,1}^+$ is better than $\Delta_{a,2}^+$.

Given these general assumptions, we consider three cases based on the assignment of aircraft a to position p:

Case-I: If aircraft a is assigned to position p, then $y_{a,\kappa} = 0$ for $\kappa = 1, \ldots, p-1, p+1, \ldots, n$. This implies that $t_a \geq \sigma_p$ as desired.

Case-II: Assume that aircraft a is assigned to position $\kappa > p$. From equation (50) and Case-I, we get $t_a \geq \sigma_\kappa$. To see that (50) is valid, it is sufficient to show that $\sigma_p + \mu_{p,a,\kappa} \leq \sigma_\kappa$. This holds since the separation requirements are maintained through constraints (21) and $\mu_{p,a,\kappa}$ defines the shortest possible sequence.

Case-III: Assume that aircraft a is assigned to position $\kappa < p$. Similar to the above, we get $t_a \geq \sigma_\kappa$. To show that (50) is valid, we need to show that $\sigma_p - u_{p,a,\kappa} \leq \sigma_\kappa$. After substitution and rearrangement, we state the required relationship as $\sigma_\kappa + \Psi(p-\kappa, a) + r_n \geq \sigma_p$.

Assume this is not the case, i.e. $\varsigma = \sigma_\kappa + \Psi(p-\kappa, a) + r_n < \sigma_p$ for some optimal σ and t. Let $\sigma'_p = \sigma_p - \epsilon$ where $\epsilon > 0$ is small enough such that $\varsigma \le \sigma'_p$. σ'_p is feasible as it is scheduled after the time of the last scheduled aircraft plus the length of the longest possible sequence of length $p-\kappa$. Note that we do not need to consider the longest sequence of length $p-\kappa+1$ due to assumption (1) above. Let $t_{a(p)} = \sigma_p$ and $t'_{a(p)} = \sigma'_p$, where $a(p)$ denotes the aircraft assigned to position p. As both $t_{a(p)} > r_n$ and $t'_{a(p)} > r_n$, $\Delta^+_{a(p)} > 0$ in both cases. Moreover, $t_{a(p)} > t'_{a(p)}$ and by assumption (2), this contradicts the fact that the σ and t are optimal.

Hence, constraint (50) is valid. □

In an analogous way to the discussion above, we define $\bar{\Psi}(p,a)$ as the longest possible sequence to position p when the weight class of aircraft a must be a part of the sequence and $\bar{\Phi}(\kappa, a)$ as the earliest time possible for position κ given that aircraft a is assigned to the first position. Then, the following result can be concluded:

Proposition 3. *For $\bar{u}_{p,a,\kappa} = \bar{\Psi}(\kappa - p, a) + r_n$ and $\bar{\mu}_{p,a,\kappa} = \bar{\Phi}(p-\kappa+1, a)$, constraint* (51) *is a valid inequality for SRSP.*

Proof. We again consider three cases:

Case-I: If aircraft a is assigned to position p, then $y_{a,\kappa} = 0$ for $\kappa = 1, \dots, p-1, p+1, \dots, n$. This implies that $t_a \le \sigma_p$ as desired.

Case-II: Assume that aircraft a is assigned to position $\kappa < p$. From equation (51) and Case-I, we get $t_a \le \sigma_\kappa$. To see that (51) is valid, it is sufficient to show that $\sigma_\kappa \le \sigma_p - \bar{\mu}_{p,a,\kappa}$. This holds since the separation requirements are maintained through constraints (21) and $\bar{\mu}_{p,a,\kappa}$ defines the shortest possible sequence.

Case-III: Assume that aircraft a is assigned to position $\kappa > p$. Similar to the above, we get $t_a \le \sigma_\kappa$. To show that (51) is valid, we need to show that $\sigma_p + \bar{u}_{p,a,\kappa} \ge \sigma_\kappa$. After substitution and rearrangement, we state the required relationship as $\sigma_p + \Psi(\kappa - p, a) + r_n \ge \sigma_\kappa$. This can be shown to hold by reversing the roles of p and κ in Case-III in the proof of Proposition 2.

Hence, constraint (51) is valid. □

Given the validity of constraints (50) and (51), an important issue deals with the tightness of these inequalities. We show below that these constraints cannot be made any tighter, which has implications for the complexity and efficiency of SRSP and SRSP-R.

Proposition 4. *If $\mu_{p,a,\kappa} = \Phi(\kappa - p + 1, a)$ and $u_{p,a,\kappa} = \Psi(p - \kappa, a) + r_n$, then constraint (50) is tight, i.e. $\mu_{p,a,\kappa}$ cannot be made larger and/or $u_{p,a,\kappa}$ cannot be made smaller without violating the validity of the inequality.*

Proof. We analyze the bounds on $\mu_{p,a,\kappa}$ and $u_{p,a,\kappa}$ separately through two instances.

For $\mu_{p,a,\kappa}$, consider an instance with two aircraft, a_1 and a_2 scheduled in that order, of the same weight class. Let the separation between aircraft of this weight class be s^*, clearly the shortest sequence is of length s^*. The separation constraints together with (50) and (51) for $(p = 1, a_1)$ and $(p = 2, a_2)$ give $\sigma_1 = 0 = t_{a_1}$ and $\sigma_2 = s^*$. Letting $\mu_{1,a_2,2} = \Phi(2, a_2) = s^*$ we have the constraint $t_{a_2} \geq \sigma_1 + \mu_{1,a_2,2} y_{a_2,2} = 0 + \mu_{1,a_2,2} = s^*$. Increasing μ by any amount $\epsilon > 0$ cuts off the feasible point $t_{s_2} = s^*$.

For $u_{p,a,\kappa}$, consider the set of aircraft a_1, a_2, and a_3 of different weight classes k_1,k_2, and k_3, respectively. Let $r_{a_1} = 0$, $r_{a_2} = 0$, and $r_{a_3} = 10$. Assume that the weight class sequence in the first stage takes values $x_{1,k_2} = x_{2,k_3} = x_{3,k_1} = 1$ and that s_{k_2,k_3} and s_{k_3,k_1} are the two largest separation requirements, i.e. this is the longest sequence. As the weight classes are distinct, there is only one way to assign aircraft. We have $y_{a_2,1} = y_{a_3,2} = y_{a_1,3} = 1$. Assuming that $s_{k_2,k_3} \leq 10$, the separation constraint (21) together with the earliest time constraints, and constraints (50) and (51) for $(p = 1, a_2)$, $(p = 2, a_3)$ and $(p = 3, a_1)$ give $\sigma_1 = t_{a_2} = 0, \sigma_2 = t_{a_3} = 10$ and $\sigma_3 = t_{a_1} = 10 + s_{k_3,k_1}$.

For aircraft a_2 and position $p = 3$, (50) gives $t_{a_2} \geq \sigma_3 - \sum_{\kappa=1}^{2} u_{3,a_2,\kappa} y_{a_2,\kappa} = \sigma_3 - u_{3,a_2,1} = \sigma_3 - \Psi(2, a_2) - r_{a_3} = (10 + s_{k_3,k_1}) - (s_{k_3,k_1} + 10) = 0$. Decreasing u by any amount $\epsilon > 0$ cuts off the feasible point $t_{a_2} = 0$. □

Proposition 5. *If $\bar{\mu}_{p,a,\kappa} = \bar{\Phi}(p - \kappa + 1, a)$ and $\bar{u}_{p,a,\kappa} = \bar{\Psi}(\kappa - p, a) + r_n$, then constraint (51) is tight, i.e. $\bar{\mu}_{p,a,\kappa}$ cannot be made larger and/or $\bar{u}_{p,a,\kappa}$ cannot be made smaller without violating the validity of the inequality.*

Proof. For $\bar{\mu}_{p,a,\kappa}$, consider an instance with two aircraft, a_1 and a_2 scheduled in that order, of the same weight class. Let the separation between aircraft of this weight class be s^*. Clearly, the shortest sequence is of length s^*. The separation constraints together with (50) and (51) for $(p = 1, a_1)$ and $(p = 2, a_2)$ give $\sigma_1 = 0 = t_{a_1}$ and $\sigma_2 = s^*$. Letting $\bar{\mu}_{2,a_1,1} = \Phi(2, a_1) = s^*$ we have the constraint $t_{a_1} \leq \sigma_2 - \bar{\mu}_{2,a_1,1} y_{a_1,1} = s^* - s^*$. Increasing $\bar{\mu}$ by any amount $\epsilon > 0$ cuts off the feasible point $t_{a_1} = 0$.

For $\bar{u}_{p,a,\kappa}$, consider the set of aircraft a_1, a_2, and a_3 of different weight classes k_1,k_2, and k_3, respectively. Let $r_{a_1} = 0$, $r_{a_2} = 0$, and $r_{a_3} = 10$. Assume the weight class sequence in the first stage takes values $x_{1,k_2} = x_{2,k_3} = x_{3,k_1} = 1$ and that s_{k_2,k_3} and s_{k_3,k_1} are the two largest separation requirements, i.e. this is the longest sequence. As the weight classes are distinct, there is only one way to assign aircraft. We have $y_{a_2,1} = y_{a_3,2} = y_{a_1,3} = 1$. Assuming that $s_{k_2,k_3} \leq 10$, the separation constraints (21) together with the earliest time constraints, and constraints (50) and (51) for $(p = 1, a_2)$, $(p = 2, a_3)$ and $(p = 3, a_1)$ give $\sigma_1 = t_{a_2} = 0, \sigma_2 = t_{a_3} = 10$ and $\sigma_3 = t_{a_1} = 10 + s_{k_3,k_1}$.

For aircraft a_1 and position $p = 1$, (51) gives $t_{a_1} \leq \sigma_1 + \sum_{\kappa=2}^{3} \bar{u}_{1,a_1,\kappa} y_{a_1,\kappa} = \sigma_1 + \bar{u}_{1,a_1,3} = \sigma_1 + \bar{\Psi}(2, a_1) + r_{a_3} = 0 + (s_{k_3,k_1} + 10) = s_{k_3,k_1} + 10$. Decreasing $\bar{u}$ by any amount $\epsilon > 0$ cuts off the feasible point $t_{a_1} = 10 + s_{k_3,k_1}$. □

4.4 Solution Methodology for SRSP-R and SRSP

Noting that the deterministic version of the runway scheduling problem is NP-complete by itself, it is clear that the solutions of SRSP-R and SRSP, which involve several scenario problems, would require significant computational effort. We further discuss and quantitatively demonstrate this issue in Section 4.6. For the solution methodology, we propose a scenario decomposition procedure coupled within a sampling based approximation method to obtain good solutions in practically reasonable amounts of time. We especially emphasize practical implementability due to the dynamic nature of runway scheduling and the resulting need for fast solutions. Our computational results described in Section 4.6 show the potential effectiveness of the proposed solution procedures in achieving this.

4.4.1 Scenario Decomposition

Our decomposition approach is based on a Lagrangian decomposition framework with special improvement procedures. Other decomposition approaches that can potentially be considered for implementation have several limitations that prevent them from being applicable. Of these, the classical Benders' framework is unable to capture the second stage decision process as discussed in Section 4.1.1, while generalized Benders' methods (Geoffrion 1972) are inapplicable due to the problem failing to satisfy a required property for such implementations. On the other hand, combinatorial Benders' approaches (Codato and Fischetti 2006a) are not effective due to the continuous objective function structure in SRSP. The general procedure we utilize is based on scenario decomposition where we relax the nonanticipativity constraints for the first stage decision variables.

As part of the decomposition procedure, we slightly revise the described stochastic programming formulations for SRSP by defining the first stage variables separately for each scenario ω, i.e. as $x^{\omega}_{p,k}$ and ℓ^{ω}_{p}, and then by adding explicit nonanticipativity constraints that would require the values of these variables to be equal for each scenario.

As a simplifying characteristic for SRSP, we note below that it suffices to enforce nonanticipativity only over the assignment variables $x^{\omega}_{p,k}$ for both flow and slot formulations.

Proposition 6. *For any pair of scenarios ω and ω', if $x^{\omega}_{p,k} = x^{\omega'}_{p,k}$, $\forall p \in \mathcal{P}$, then $\ell^{\omega}_{p} = \ell^{\omega'}_{p}$, $\forall p \in \mathcal{P}$ for both the slot and flow formulations.*

Proof. We assume that there is a positive cost associated with ℓ^{ω}_{n}, $\forall \omega \in \Omega$, where the cost does not depend on scenarios as otherwise the value of ℓ^{ω}_{p}, $\forall p \in \mathcal{P}$, $\omega \in \Omega$ is not relevant. Since we consider a minimization problem, ℓ^{ω}_{n} is minimized. Thus, constraints (28) and (29) in the flow formulation and constraints (35) and (36) in the slot based formulation hold at equality in all scenarios. As the weight class sequence is the same over all scenarios and the separation requirements only depend on the weight classes, the values of ℓ^{ω}_{p}, $\forall \omega \in \Omega$ are equal $\forall p \in \mathcal{P}$. $\square$

This property allows for reduction in the number of Lagrangian multipliers to be optimized in the solution of the Lagrangian dual, allowing for improved convergence in the subgradient procedure. Hence, assuming that all first stage variables are indexed by ω, we include the following constraint in the formulation of SRSP to enforce nonanticipativity:

$$x^{\omega}_{p,k} = \sum_{\omega' \in \Omega} \varrho^{\omega'} x^{\omega'}_{p,k} \qquad \forall \omega \in \Omega,\ p \in \mathcal{P},\ k \in \mathcal{K} \tag{52}$$

Note that the structure used in the formulation of constraint (52) accounts for the scenario probabilities, and prevents the ill-conditioning in the Lagrangian dual, as described by Louveaux and Schultz (2003). Although there are other formulations for the non-anticipativity constraints, initial experiments indicate that the formulation in Equation (52) gives the best computational performance for this particular solution methodology. The duality gaps, that can exist in scenario decomposition methods for integer stochastic programming, are typically very close to zero in the implementations for the runway scheduling problem, suggesting the effectiveness of the proposed procedures on this problem.

We let the Lagrangian dual variables be denoted by $\alpha^{\omega}_{p,k}$ and relax constraint (52) to form the following two Lagrangians for the updated objective functions of SRSP-R and SRSP, respectively.

$$L_R(\mathbf{x}, \boldsymbol{\alpha}) \equiv \sum_{\omega \in \Omega} \varrho^{\omega} \left(\lambda \ell^{\omega}_{n} + (1-\lambda) \left(\sum_{p=1}^{n} \sigma^{\omega}_{p} - C^{\omega} \right) \right) + \sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} \alpha^{\omega}_{p,k} \left(\sum_{\omega' \in \Omega} \varrho^{\omega'} x^{\omega'}_{p,k} - x^{\omega}_{p,k} \right) \tag{53}$$

$$L(\mathbf{x}, \boldsymbol{\alpha}) \equiv \sum_{\omega \in \Omega} \varrho^{\omega} \left(\rho^{\omega} + \left(\sum_{a \in \mathcal{A}} \rho^{+}_{a,\omega} + \rho^{-}_{a,\omega} \right) \right) + \sum_{p \in \mathcal{P}} \sum_{k \in \mathcal{K}} \alpha^{\omega}_{p,k} \left(\sum_{\omega' \in \Omega} \varrho^{\omega'} x^{\omega'}_{p,k} - x^{\omega}_{p,k} \right) \tag{54}$$

For given dual vectors $\boldsymbol{\alpha}$, both $L_R(\mathbf{x}, \boldsymbol{\alpha})$ and $L(\mathbf{x}, \boldsymbol{\alpha})$ can be decomposed by scenarios, which allows for the solution of the corresponding Lagrangian dual problem in each case by independently solving individual scenario problems, where the Lagrangian dual problem can be expressed as:

$$\max_{\alpha} \{ \mathcal{Z}(\mathbf{x}, \boldsymbol{\alpha}) = \min \{ \sum_{\omega \in \Omega} L^{\omega}(\mathbf{x}^{\omega}, \boldsymbol{\alpha}^{\omega}) \} \} \tag{55}$$

This follows from the fact that the Lagrangian dual problem is a non-smooth concave maximization problem, and can be solved by subgradient optimization methods (Hiriart-Urruty and Lemarechal 1993). At each iteration of these methods, the solution of $\mathcal{Z}(\mathbf{x}, \boldsymbol{\alpha})$, which is separable by scenarios, is required to obtain a subgradient. The dual multipliers are updated at each iteration based on the components of the subgradient vector and a step size rule until a termination criterion is reached.

The specific implementation that we use for this procedure involves several special steps and is summarized as follows. Note that $\boldsymbol{\alpha}$ refers to the vector of Lagrange multipliers, where each component is $\alpha^{\omega}_{p,k}$. Moreover, although we refer to SRSP in some steps of the algorithm, it is implied that the procedure also applies to SRSP-R.

Step 0: *Initialization.* Set iteration counter $q = 1$, no-improvement counter $i = 0$, initial lower bound $\underline{L}^0 = -\infty$, best lower bound $\underline{L} = \underline{L}^0$. Set initial dual multipliers $\boldsymbol{\alpha}^q$, termination thresholds $\epsilon^{\text{abs}}, \epsilon^{\text{rel}}$, limit $\bar{\imath}$ for no improvement, and the step size parameters ϕ and θ. Calculate the parameters used in the valid inequalities (50) and (51). Calculate the initial upper bound $\bar{L}^0$ as follows:

- For each scenario ω, identify the FCFS sequence $FCFS^{\omega}$.
- For each $FCFS^{\omega}$, fix $x \equiv FCFS^{\omega}$ and solve SRSP to obtain candidate upper bound $\bar{L}^0_{\omega}$.
- Set $\bar{L}^0 = \min_{\omega}\{\bar{L}^0_{\omega}\}$. Set the best upper bound $\bar{L} = \bar{L}^0$.

Step 1: *Lower Bound.* Solve $\mathcal{Z}(\mathbf{x}, \boldsymbol{\alpha}^q)$ to obtain candidate lower bound $\underline{L}^q$ and sequence x^q_{ω}. If $\underline{L}^q > \underline{L}$, set $\underline{L} = \underline{L}^q$.

Step 2: *Upper Bound.* Try to improve $\bar{L}$ as follows:

- Collect the sequences obtained in Step 1 and list them by number of occurrences. Let X^q_{UB} be the set of sequences with most occurrences.
- For each sequence $\mathbf{s} \in X^q_{\text{UB}}$, fix $\mathbf{x}$ according to $\mathbf{s}$ and solve SRSP to obtain candidate upper bound $\bar{L}^q_s$.
- Set $\bar{L}^q = \min_{\mathbf{s} \in X^q_{\text{UB}}}\{\bar{L}^q_s\}$. If $\bar{L}^q < \bar{L}$, set $\bar{L} = \bar{L}^q$.

Step 3: *Step Size.* Let $\Lambda^q_L = \bar{L} - \underline{L}$. Update the step size parameter θ as follows:

Table 15: Parameters used in the implementation of the scenario decomposition procedure.

	SRSP-R	SRSP
$\boldsymbol{\alpha}^q$	1	1
ϵ^{abs}	1×10^{-3}	1×10^{-3}
ϵ^{rel}	1×10^{-3}	1×10^{-3}
$\bar{i}$	5	5
ϕ	0.1	1
θ	0.25	0.25
$\pi_0, \pi_1, \pi_2, \pi_3$	0.6,0.2,0.1,0,1	0.6,0.2,0.1,0,1

- If $\Lambda_L^q > \Lambda_L^{q-1}$, then set $i \leftarrow i+1$, and
- If $i = \bar{i}$, then set $\theta \leftarrow \theta/2$, $i \leftarrow 0$.

Step 4: *Step Direction.* Set bundled subgradient $\hat{\mathbf{g}}^q$ as $\hat{\mathbf{g}}^q = \pi_0\mathbf{g}^q + \pi_1\mathbf{g}^{q-1} + \pi_2\mathbf{g}^{q-2} + \pi_3\mathbf{g}^{q-3}$, where $\mathbf{g}^k$ is the subgradient in iteration k.

Step 5: *Multiplier Updates.* Set multipliers $\boldsymbol{\alpha}^{q+1} = \boldsymbol{\alpha}^q + \max(\frac{\phi}{q}, \frac{\theta(\bar{L}-L^q)}{||\hat{\mathbf{g}}^q||})\hat{\mathbf{g}}^q$.

Step 6: *Termination.* Terminate if

- $\Lambda_L^q < \epsilon^{\text{abs}}$, or
- $\Lambda_L^q/\bar{L} < \epsilon^{\text{rel}}$

Step 7: *Iteration.* Update $q = q+1$, go to Step 1.

The implementation described above involves effective upper and lower bounding procedures, as well as a modified subgradient optimization structure where the step direction is determined based on a weighted sum of subgradients from previous iterations. As in many subgradient optimization implementations, the ideal parameters settings for the algorithm are based on problem characteristics and are determined experimentally. The values used for the parameters of the algorithm in solving SRSP-R and SRSP are presented in Table 4.4.1 separately.

4.4.2 The Sample Average Approximation Implementation

While the scenario decomposition procedure is effective in getting high quality solutions, the large number of scenarios in a typical instantiation of SRSP prevents it from being directly applied to solve the exact problem. Hence, we make use of the sample average approximation (SAA) method (also known as the sample path method), a Monte Carlo simulation technique that approximates a stochastic program by a set of smaller problems based on a random sample from the set of possible scenarios (Linderoth, Shapiro, and Wright 2006; Shapiro 2003). Noting that $\omega^1, \ldots, \omega^N$ is an independent and identically distributed (i.i.d.) random sample of N realizations of the random vector ω, the SAA problem for either formulation of SRSP can be defined as:

$$\min_{\mathbf{x} \in \mathcal{X}} \{\hat{c}_N(\mathbf{x}) = \frac{1}{N} \sum_{l=1}^{N} c(\mathbf{x}, \omega^l)\} \tag{56}$$

where $c(\mathbf{x}, \omega^l)$ is the objective function for realization ω^l. If ν^* and $\hat{\nu}_N$ represent the optimal values of the "true" and SAA problems respectively Kleywegt, Shapiro, and De-Mello (2002) show that $\hat{\nu}_N$ converges to ν^* at an exponential rate as sample size N is increased. However, given that the computational complexity of the SAA problem increases exponentially with the value of N, it is typically more efficient to select a smaller sample size N, and solve several SAA problems with i.i.d. samples.

We solve M SAA problems with N samples in each, and use $\hat{\nu}_N^m$ and $\hat{x}_N^m$, $m = 1, \ldots, M$, to refer to the optimal objective value and solution of the mth replication, respectively. Once a feasible solution $\hat{x}_N^m \in \mathcal{X}$ is obtained by solving the SAA problem, the objective value $c(\hat{x}_N^m)$ needs to be determined. The value of a given solution can be approximated by the estimator

$$\hat{c}_{N'}(\hat{x}_N^m) = \frac{1}{N'} \sum_{l=1}^{N'} c(\hat{x}_N^m, \omega^l) \tag{57}$$

where N' is typically larger than N, as the computational effort required to estimate the objective value for a given solution is generally less than that required to solve the SAA problem. The quality of a solution $\hat{x}_N^m$ is then computed through the optimality gap estimator $c(\hat{x}_N^m) - \nu^*$, where $c(\hat{x}_N^m)$ can be calculated exactly or estimated by (57), and ν^* is approximated by

$$\bar{\nu}_N^M = \frac{1}{M} \sum_{m=1}^{M} \hat{\nu}_N^m \tag{58}$$

Given that the sample sizes would have a significant impact on the accuracy of this optimality gap estimator, it is essential that the variances of the estimators are taken into account. Using a $(1-\varpi)\%$ confidence level similar to Kleywegt, Shapiro, and De-Mello (2002), an upper bound on the optimality gap estimator can be defined as

$$c(\hat{x}_N^m) - \nu^* + z_\varpi \sqrt{\frac{\Sigma_{N'}^2}{N'} + \frac{\Sigma_M^2}{M}} \tag{59}$$

where $\Sigma_{N'}^2$ and Σ_M^2 are the sample variances for the corresponding estimators.

The sampling procedure can be terminated once the optimality gap estimate is sufficiently small or after performing all M replications, and the best solution among the SAA solutions can be selected using an appropriate criterion.

In the computations performed, values of $N = 8$ and $N = 12$ are investigated. While these sample sizes can be considered small, this is a limitation caused by the complex combinatorial structure of the problem, as solving even a single scenario problem may be difficult for certain instances. When the first stage variables are fixed, the problem decomposes into separate second-stage problems that can be solved independently from each other. In our implementation we use $N' = 200$ scenarios when evaluating a first stage sequence. We use $M = 100$ as an upper bound on the number of replications.

Effective implementation of the above sampling procedure requires that the SAA problems can be solved efficiently and the candidate solutions are evaluated accurately. To this end, the improvements described in Section 4.3 have been analyzed in terms of their impact on achieving higher computational efficiency. These computational experiments are described in Section 4.5 below.

4.5 Computational Framework and Experimental Setup

In the previous sections of this chapter we have described the stochastic version of a practical scheduling problem for air traffic flow management, namely the runway scheduling problem. We noted the restricted and general versions of this problem, and derived alternative formulations for the two problem types. Several improvement procedures for the formulations were also developed. Given this framework, there are some natural questions that should be explored, such as: (1) Is the flow or slot formulation better for SRSP-R and SRSP, and considering the solution procedure to be used, does the best formulation

Table 16: Overall fleet mix at the 35 OEP airports in 2009. The fleet mix is obtained from 12 months of Aviation System Performance Metrics (ASPM) data.

Weight Class	Percentage of Operations
Heavy	6.3%
Boeing 757	7.1%
Large	82.3%
Small	4.3%

Table 17: Description of instance groups used for model evaluation.

Instance Group Name	Sampled Arrival Rate [flights/hr]	Sampled Departure Rate [flights/hr]	Resulting Fleet Mix			
			H	7	L	S
Group 1	20	20	0.07	0.16	0.72	0.05
Group 2	30	30	0.12	0.12	0.72	0.03
Group 3	40	40	0.07	0.17	0.71	0.05
Group 4	20	0	0.1	0.1	0.79	0.01
Group 5	30	0	0.08	0.08	0.79	0.04
Group 6	45	0	0.1	0.07	0.79	0.04

differ for the deterministic and stochastic instances?, (2) What is the impact of some improvement procedures on computational performance of SRSP? (3) How difficult is SRSP over SRSP-R?, and (4) How implementable is SRSP in practice? We perform numerous computational tests and try to obtain answers to these questions.

For evaluation of the various formulations, we generate 30 random instances that reflect realistic runway schedules. The instances are generated from six base cases where each case is based on a specific arrival and departure rate. Using the given rates we generate five random schedules for each base case. The schedules are generated assuming a Poisson arrival process corresponding to exponential inter-arrival times. This assumption has been commonly used and justified in the literature, for example in Balakrishan and Chandran (2010).

In each base case we use the fleet mix shown in Table 4.5 when assigning aircraft weight classes to the sampled flights. The fleet mix is the overall fleet mix at the 35 Operational Evolution Partnership (OEP) airports for calendar year 2009, and is thus reflective of actual operational conditions (Federal Aviation Administration 2009). A summary of the sampled arrival and departure rates together with the resulting fleet mixes for the six base cases are shown in Table 4.5.

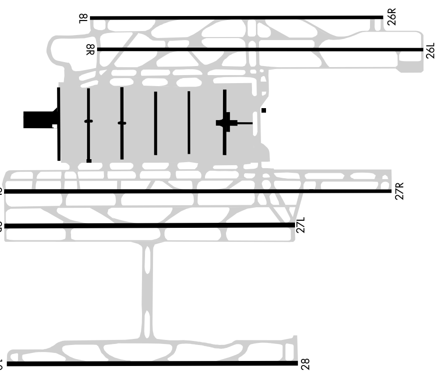


Figure 12: Hartsfield-Jackson Atlanta International Airport operates two-pairs of close parallel runways where 26L and 27R are used for departures, while 26R and 27L are used for arrivals.

We assume that operations take place on two close parallel runways with a dedicated runway for arrivals and a runway for departures. This configuration is widely used at major airports (Doyle and McGee 1998). Currently, nine of the ten busiest airports in the U.S. have at least one pair of close parallel runways and with the completion of the modernization plan at Chicago O'Hare Airport, all of the ten busiest airports will have close parallel runways. Figure 12 shows such a configuration with two pairs of close parallel runways (26L-26R and 27L-27R) at Hartsfield-Jackson Atlanta International Airport. Computational experiments also involve several instances where only arriving aircraft are considered. For those instances, single runway operations are assumed. However, as noted previously, when operating under instrumental flight rules, the arrival and departure runways cannot be independently scheduled. In that case, separation requirements defined in Section 4.1.2 for operations on parallel runways are used.

In the stochastic runway scheduling problem we treat the estimated time of arrival at the runway as probabilistic. As is customary in stochastic programming, we discretize the probability distribution of estimated runway arrival times into discrete probability distributions for use in the scenario generation process. In this implementation we use three different outcomes: a flight can either be early, on-time or late. We use the expected value of the estimated runway time distribution as the on-time outcome, and the expected value plus/minus one standard deviation as the early and late outcomes, respectively. For arrival distributions, we analyze historical Enhanced Traffic Management System (ETMS) data obtained from the Federal Aviation Administration (FAA) to estimate the deviations from estimated arrival times. For departures we analyze historical push-back delays and use results found in the literature for taxi delay, which are also described in Solveling, Solak, Clarke, and Johnson (2011a).

For performance evaluations, we have implemented the various model formulations in C++ using Concert/CPLEX 12.2. The computations were performed on a quad-core computing cluster whose head node is a 2.66 GHz Xeon X5355 processor.

4.6 Computational Study and Analysis

We perform our computational analysis by considering different configurations and performing comparisons among these configurations. Our analysis has four major dimensions.

The first dimension involves identifying the best formulation structures for SRSP-R and SRSP. This dimension has two components consisting of the identification of the best formulation structure for (1) deterministic instances solved through branch-and-bound methods, and (2) stochastic SAA problem instances with limited number of scenarios which are solved through decomposition. The complete SAA algorithm implementation was only performed for SRSP using the best formulation structures.

The second dimension in our analysis deals with the impacts of potential improvement procedures for each formulation. As part of the third dimension, we consider the best formulations for SRSP-R and SRSP, and then compare the two problem types from an efficiency and complexity perspective. Finally, we analyze the mathematical models from a practical implementation perspective. To this end, several practically tractable implementation alternatives are considered and discussed.

However, we first demonstrate the computational challenges of solving the deterministic equivalent of a given stochastic programming instance of SRSP. In Figure 13 we show a comparison of run times when the problem is solved using scenario decomposition versus when the deterministic equivalent formulation is used directly. In this figure, the solid lines represent average and median run times when scenario decomposition is used and the

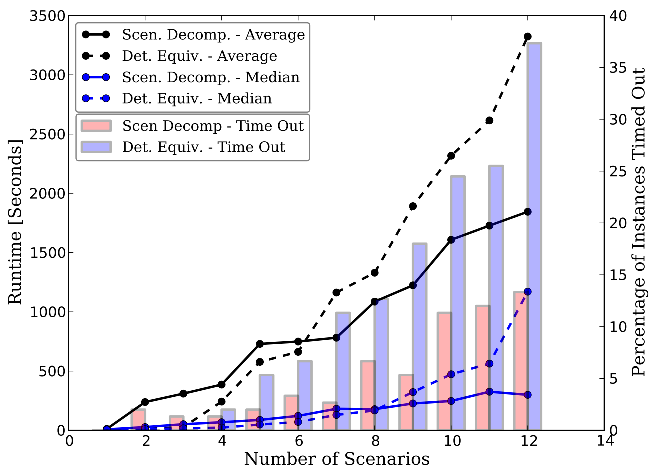


Figure 13: Comparison of run times when the problem is solved using scenario decomposition versus when the deterministic equivalent formulation is used directly. The higher values correspond to the deterministic equivalent solution.

dashed lines show average and median run times when the deterministic equivalent problem is solved. The higher values correspond to the deterministic equivalent solution. The bars show the percentage of instances that time out in 1 hour, where 300 instances are used for each configuration. The exponential increase in the run times for the deterministic equivalent model is evident in these plots, while the scenario decomposition approach is clearly more robust to increases in problem size, demonstrating the value and need for a special solution algorithm.

4.6.1 Analysis-I: Efficiency of Flow vs. Slot Formulations for SRSP-R and SRSP

We compare the efficiency of the two types of formulations separately for different problem structures. More specifically, we consider the computational performances of the two formulations for deterministic instances, stochastic instances with limited number of scenarios and stochastic instances solved using the SAA method. The last configuration is analyzed only for SRSP.

4.6.1.1 Deterministic Problem

SRSP-R. We show in Table 4.6.1.1 the computational performances of the flow and slot formulations for deterministic instances of SRSP-R, where a run time limit of 1 hour is used. The results are for a single scenario and show the average and median CPU time, as well as the average integrality gap for the LP-relaxation at the root node of the branch and bound tree. The reason we investigate the comparative statistics involving the average and median CPU times is due to the potential high variance in instance complexities. For example, a single difficult instance can pull the average CPU time up significantly, while its effect on the median CPU time would be not so significant. Hence, the two statistics are considered together, and the difference between the two is used as a proxy measure of the variance of instance complexity in each configuration. In addition, the integrality gap at the root node is also used to distinguish between the two formulations as a measure to diagnose the reasons for the distinctions.

Overall, as seen in Table 19a, it is clear that the flow formulation performs much better for deterministic SRSP-R, with almost ten times the computational efficiency of the slot formulation. This efficiency is likely due to the network structure in the flow formulation, which results in better LP relaxation solutions during the branch and bound implementation as seen through the smaller integrality gaps. We also note that the median CPU times are much lower than the average values for both types of formulations, implying that most instances of the problem are typically solved relatively fast while few instances result in longer computational times.

It is also important to consider the efficiency of the two formulations over different number of flights and different schedule densities. An interesting observation seen in Table 19b is that tighter schedules result in faster computational times in the flow formulation, while the slot formulation displays the exact opposite trend. This holds both for Groups 1-3 where mixed operations are scheduled and for Groups 4-6 where only arrivals are considered. This non-intuitive result is most likely due to the decrease in the integrality gap as the network structure becomes more rigid with the introduction of tighter operation windows.

With regard to the performance of the two formulations over different number of flights,

Table 18: Computational results comparing the slot based formulation to the network flow based formulation of deterministic instances of SRSP-R.

Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
Slot	Flow	Slot	Flow	Slot	Flow
707.3	34.4	47.2	4.2	83.4	23.4

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Slot	Flow	Slot	Flow	Slot	Flow
Group 1	375.1	31.8	27.8	10.6	84.0	35.0
Group 2	891.4	4.4	122.2	3.6	87.3	10.8
Group 3	1495.3	3.9	376.8	2.6	89.3	8.7
Group 4	107.1	86.7	11.6	11.2	78.3	44.8
Group 5	345.0	73.5	39.7	8.3	80.8	28.2
Group 6	1029.9	6.0	151.0	1.8	80.7	12.5

(b) Results aggregated by groups.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Slot	Flow	Slot	Flow	Slot	Flow
8	2.5	0.3	1.8	0.3	79.0	22.3
10	45.6	4.0	13.5	2.9	82.7	23.2
12	692.7	18.9	126.7	10.7	85.2	23.9
14	2088.4	114.3	1639.2	18.1	86.8	24.1

(c) Results aggregated by number of flights.

the exponential increase in computational times is evident in Table 19c, especially for the slot formulation of SRSP-R. The flow formulation is a bit more robust as the rate of increase in run times is lower. Overall, it can be concluded that the deterministic instances of SRSP-R with up to 10 aircraft can be solved in near-real time.

SRSP. Similar results for deterministic instances of SRSP are shown in Table 4.6.1.1, which suggest a somewhat different performance structure than SRSP-R. The first observation is that the significant superiority of the flow formulation does not hold for SRSP. Indeed, the median CPU time for the slot formulation is lower than the flow formulation as seen in Table 20a. The fact that the average CPU time is higher for the slot formulation is due to the relative inefficiency of the model in larger instances, also reflected in Table 20c. For smaller instances, i.e. those with 8 or 10 aircraft, the slot formulation is mostly equivalent to flow formulation. This is despite the difference in the integrality gap at the root node. For larger instances, the median run time for the slot formulation is lower but the average value is higher. Hence, there does not appear to be a distinctly better formulation for deterministic instances of SRSP. Overall, however, it can be concluded that the advantage gained by the network structure in SRSP-R is lost due to the more complicated objective function form in SRSP.

The loss of the network properties is also supported by the observation in Table 20b that tighter schedules result in worsened computational performance in the flow formulation for SRSP. There is also a clear distinction between the performances of the two formulations when schedules consist of arrivals only, as opposed to mixed flight operations. In the former case, the slot formulation is superior, while flow formulation is observed to be better in the latter case. A general conclusion is that the slot formulation is expected to perform better for SRSP instances with only arrivals or with smaller number of aircraft considered. More complex instances of SRSP are better handled by the flow formulation.

Overall, we conclude that for deterministic instances, the flow formulation is significantly better for SRSP-R for all types of instances. For SRSP, either formulation may be preferred, depending on schedule density and the mix of operations.

4.6.1.2 Stochastic Problem with Limited Number of Scenarios

These limited scenario problems correspond to instances where the deterministic equivalent of the stochastic programming problem is solved using scenario decomposition. This is only possible if the decomposition procedure can be solved in a reasonable amount of time. The SAA problems described in Section 4.4 fall into this category as multiple solutions of such problems are required as part of the iterative SAA method implementation.

SRSP-R. In Table 4.6.1.2, we compare the performance of flow and slot formulations for the SAA instances of SRSP-R. For each group, schedule density, and number of scenarios, we report the average and median CPU times, the average number of iterations, in what iteration the best upper bound was found, and the average optimality gap at termination. In each stochastic instance the number of flights considered was 8 or 10, while the number

Table 19: Computational results comparing the slot based formulation to the network flow based formulation of deterministic instances of SRSP.

Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
Slot	Flow	Slot	Flow	Slot	Flow
139.4	27.4	7.7	10.0	82.4	56.0

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Slot	Flow	Slot	Flow	Slot	Flow
Group 1	28.6	19.4	10.7	11.3	54.5	53.2
Group 2	129.2	28.3	31.6	16.5	99.5	71.7
Group 3	637.0	39.3	28.1	13.5	96.8	26.6
Group 4	4.2	7.3	3.1	4.0	79.3	79.3
Group 5	8.5	40.0	5.2	9.5	84.4	71.0
Group 6	28.8	30.3	3.8	7.8	79.9	34.1

(b) Results aggregated by groups.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Slot	Flow	Slot	Flow	Slot	Flow
8	1.6	1.6	1.3	1.5	80.0	49.3
10	8.2	7.0	4.4	5.5	82.6	55.5
12	64.7	18.1	11.1	16.4	84.1	57.2
14	483.1	83.0	33.3	47.4	82.9	62.0

(c) Results aggregated by number of flights.

of scenarios was 8 or 12. A run time limit of 2 hours has been used in the implementations.

As expected, we observe in Table 21a that the behavior of the two SRSP-R formulations in the decomposition based stochastic solutions is somewhat similar to the deterministic instances. First, we note that the flow formulation performs better in general, both in terms of average and median CPU times. However, the distinction is not as large since it takes more iterations for the flow formulation to converge. Hence, the slot formulation solutions provide a better path in the subgradient optimization, but they cost more computationally. The same pattern can be seen in Tables 21b-21d across different instance configurations. The flow formulation is more robust as the instances get more difficult, and the overall performance is thus better. We also note that the tighter schedules result in poorer performance in the flow formulation for the stochastic instances. This may be due to the larger number of iterations performed for tighter schedules.

An interesting observation is that the best upper bound, i.e. the best solution, is identified quite early in the iterations for both implementations. This is especially the case for schedules consisting of arrivals only, where the best solution is identified in the first few iterations. This experimental conclusion suggests that a practical heuristic based on early termination of the decomposition based solution procedure may be quite effective. Such a heuristic is discussed and analyzed in Section 4.6.4. Another observation is that while an increase in computational time is evident in both formulations when more scenarios are included, this increase is at a lower rate for the slot formulation. This suggests the following for the efficiency of the two formulations as more complex problems are considered. When compared to the impact of considering more scenarios in the instances, the impact of higher scheduled densities is larger on the efficiency of slot formulation of SRSP-R, while the opposite is true for the flow formulation of SRSP-R.

SRSP. Same types of results for SRSP are shown in Table 4.6.1.2. Overall, the slot formulation is observed to be superior with respect to most measures, except for schedules that are tighter, while the flow formulation has improved performance over high schedule densities. This observation is consistent with the results for the deterministic case for

Table 20: Computational results comparing the slot based formulation to the network flow based formulation of SRSP-R using scenario decomposition.

Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
2509	737	741	99	11.8	21.2	3.4	3.7	0.5	0.1

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
Group 1	1855	900	786	226	17.5	21.7	4.8	4.4	0.1	0.1
Group 2	4205	1541	2075	153	15.0	30.6	5.4	8.2	1.2	0.1
Group 3	4362	1157	5637	212	16.8	52.8	8.5	7.7	1.3	0.1
Group 4	215	146	123	41	4.7	4.7	0.0	0.0	0.0	0.0
Group 5	1451	321	390	108	8.5	8.5	1.3	1.3	0.1	0.1
Group 6	2968	360	452	28	8.2	8.9	0.7	0.7	0.1	0.0

(b) Results aggregated by groups.

Number of Scenarios	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
8	2282	605	673	72	11.4	22.2	4.0	3.6	0.4	0.1
12	2737	870	830	146	12.1	20.2	2.8	3.8	0.5	0.1

(c) Results aggregated by number of scenarios.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
8	561	58	344	34	11.8	11.9	2.9	2.5	0.0	0.0
10	4458	1417	4018	360	11.8	30.5	3.9	4.9	0.9	0.1

(d) Results aggregated by number of flights.

Table 21: Computational results comparing the slot based formulation to the network flow based formulation of SRSP using scenario decomposition.

Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
541	786	98	135	18.4	24.8	2.1	1.8	0.2	0.2

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
Group 1	474	1773	149	198	45.5	81.5	4.0	2.8	0.1	0.3
Group 2	1361	1436	238	322	19.2	20.7	3.5	3.3	0.8	0.8
Group 3	1170	856	396	277	14.7	15.4	3.7	3.3	0.1	0.1
Group 4	20	57	13	18	8.2	8.2	0.0	0.0	0.0	0.0
Group 5	120	342	60	163	15.0	15.0	0.9	0.9	0.1	0.1
Group 6	100	254	36	77	8.0	8.0	0.5	0.5	0.0	0.0

(b) Results aggregated by groups.

Number of Scenarios	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
8	449	703	61	109	20.5	32.0	2.2	1.9	0.1	0.1
12	633	870	125	185	16.4	17.5	2.0	1.7	0.3	0.3

(c) Results aggregated by number of scenarios.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Best UB Found in Iteration		Optimality Gap at Termination [%]	
	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow	Slot	Flow
8	92	98	37	70	12.4	12.4	1.6	1.3	0.1	0.1
10	989	1474	188	488	24.5	37.2	2.6	2.3	0.3	0.4

(d) Results aggregated by number of flights.

SRSP. Indeed, it is more evident here that the slot formulation would work better for most stochastic instances of SRSP due to the fact that it takes more iterations for the flow formulation to converge.

It is still noticeable that the best solutions are identified early in the procedure, while the remaining run time is used mostly to improve the lower bound for the problem. In all cases, the convergence is achieved and the procedure is terminated with a very low optimality gap, which is typically around 0.2%. This suggests that the quality of the solutions obtained are verifiably high, despite the duality gap that typically has a negative impact on the quality of the solutions obtained using Lagrangian decomposition for such integer problems.

Overall, we conclude that the flow formulation should be preferred for SRSP-R to solve small scale stochastic problems. Similar to the deterministic case, SRSP can be solved more efficiently using the slot formulation, except for cases that involve combined operations and high schedule densities.

4.6.1.3 Stochastic Problem with Large Number of Scenarios

These problems correspond to those where a sampling based procedure, e.g. the SAA method, is used due to the large number of scenarios considered in the problem definition.

Given that the actual problem of interest is SRSP, and the relative differences between the flow and slot formulations of SRSP-R have been identified above, we perform numerical tests of the SAA implementation for SRSP only. The results of this analysis are discussed in Section 4.6.4 along with the performance comparisons made with respect to truncated run times.

4.6.2 Analysis-II: Impact of Improvement Procedures for SRSP

Given the relative complexity of SRSP over SRSP-R, it is necessary to consider the impact of computational improvement procedures for SRSP. In this section we implement several potential improvement procedures for SRSP, and determine their effectiveness in order to conclude a "best" formulation type for the model. More specifically, we consider both the flow and slot formulations and analyze the impact of the improvement procedures on each model.

4.6.2.1 Impact of Valid Inequalities

The impact of using the valid inequalities described in Section 4.3 for SRSP can be observed in Tables 4.6.2.1 and 4.6.2.1 for the slot and flow formulations, respectively. In these results, we compare the use of the valid inequalities (50) and (51) with the use of the original big-M formulation depicted in (48) and (49). The results are for a single scenario and show the average and median CPU times, as well as the average integrality gaps for the LP-relaxations. In the tables "V.I." denotes the model using valid inequalities and "Big M" denotes the standard formulation. The comparative analysis implies that the valid inequalities are quite efficient in improving computational performance. As seen in Tables

Table 22: Computational results showing the impact of using valid inequalities (50) and (51) for the slot based formulation of SRSP.

Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
Big M	V.I.	Big M	V.I.	Big M	V.I.
1177.1	506.1	12.1	8.4	94.8	82.4

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Big M	V.I.	Big M	V.I.	Big M	V.I.
Group 1	2900.0	47.0	21.2	12.0	80.0	54.5
Group 2	512.3	171.1	40.9	32.1	100.0	99.5
Group 3	3368.5	2777.3	39.6	23.4	99.7	96.8
Group 4	13.6	5.2	6.5	3.8	100.0	79.3
Group 5	84.0	11.4	8.3	6.0	100.0	84.4
Group 6	184.4	24.4	6.6	5.1	88.8	79.9

(b) Results aggregated by groups.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Big M	V.I.	Big M	V.I.	Big M	V.I.
8	1.7	1.9	1.6	1.3	95.1	80.0
10	14.3	8.4	7.2	5.4	95.5	82.6
12	335.0	88.6	48.5	14.7	95.7	84.1
14	4357.4	1925.3	540.8	38.9	92.7	82.9

(c) Results aggregated by number of flights.

23a and 24a, the average improvement in CPU time is seen to be around 30% for the slot formulation, while it is around 83% for the flow formulation.

When performance over different types of instances is considered, for the slot formulation, which typically appears to work better for SRSP, we observe more significant improvement in median CPU times as higher numbers of flights are included in the instances. This suggests that the overall impact of valid inequalities is significant, but the impact is less in magnitude for more complex instances. Moreover, the addition of valid inequalities reduce the integrality gap as expected, and this reduction is higher for lower schedule densities.

For the flow formulation, the impact of the valid inequalities is more significant, as observed in Table 24a. While the slot formulation for SRSP may performs better in some cases, there are categories of instances where the flow formulations might be preferred, such as those involving mixed operations and higher schedule densities. Hence, the significance of valid inequalities for such instances has practical and algorithmic implications. It is

Table 23: Computational results showing the impact of using valid inequalities (50) and (51) for the network flow based formulation of SRSP.

Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
Big M	V.I.	Big M	V.I.	Big M	V.I.
479.2	27.4	55.9	10.0	76.3	56.0

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Big M	V.I.	Big M	V.I.	Big M	V.I.
Group 1	331.8	19.4	53.8	11.3	80.0	53.2
Group 2	312.8	28.3	52.9	16.5	78.4	71.7
Group 3	357.4	39.3	111.6	13.5	43.3	26.6
Group 4	257.4	7.3	38.0	4.0	100.0	79.3
Group 5	777.9	40.0	75.6	9.5	93.5	71.0
Group 6	838.3	30.3	57.7	7.8	62.8	34.1

(b) Results aggregated by groups.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	Big M	V.I.	Big M	V.I.	Big M	V.I.
8	3.3	1.6	2.7	1.5	71.6	49.3
10	45.9	7.0	23.2	5.5	76.4	55.5
12	399.9	18.1	205.0	16.4	76.3	57.2
14	1467.9	83.0	1173.1	47.4	81.0	62.0

(c) Results aggregated by number of flights.

Table 24: Computational results showing the impact of the heuristic upper bound calculations.

Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Time spent in UB Heuristic [s]	
SP	UB	SP	UB	SP	UB	SP	UB
541	1557	98	58	18.4	58.1	30.6	0.0

(a) Results for all instances using the slot based model.

Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Time spent in UB Heuristic [s]	
SP	UB	SP	UB	SP	UB	SP	UB
786	1757	135	105	24.8	59.3	94.5	0.0

(b) Results for all instances using the network flow based model.

observed in Tables 24b and 24c that the valid inequalities improve the performance more significantly as higher number of flights are considered. On the other hand, the impact on performance is somewhat consistent over the range of different schedule densities, similar to the overall model which is robust to such changes in instance complexity.

4.6.2.2 Impact of Upper Bounds

In order to investigate the impact of the heuristic upper bound calculations in the scenario decomposition algorithm, we solve the stochastic problem with a limited number of scenarios using a fixed, optimal upper bound. We set the upper bound to the optimal value (obtained in earlier runs) in the initialization phase of the algorithm and do not spend any time trying to improve the upper bound. We compare the run time for this algorithm to the complete algorithm where upper bound improvement is included as described in Step 2 of the scenario decomposition implementation in Section 4.4.1. Furthermore, we record the time spent in the upper bound heuristic during the algorithmic runs. The results can be seen in Table 4.6.2.2. In Table 4.6.2.2, the column denoted "SP" shows the results for the standard decomposition implementation and the column denoted "UB" shows the result using the same algorithm but holding the upper bound fixed at the optimal value. The table also includes the average and median CPU times, the number of iterations and the time spent by the upper bound heuristic. The time spent by the heuristic is included in the reported average and median run times.

For both flow and slot formulations, the run time for the scenario decomposition procedure is increased when a fixed upper bound, i.e. the optimal objective function value, is used in determining step sizes. In other words, despite the additional computational time spent in calculating an upper bound, the use of an upper bound heuristic indeed allows for faster convergence through fewer iterations. This is likely due to the role that the upper bounds play in the determination of the step sizes in the subgradient optimization algorithm. In the case of a fixed upper bound, the dynamics of the search procedure are not reflected in the iterations, thus resulting in slower convergence. The impact is similar for both types of formulations. Moreover, it can be observed that the total time spent in calculating the upper bounds is not that significant when compared to the overall run times.

4.6.2.3 Impact of Prioritization

In order to potentially increase the efficiency of the two formulations of SRSP, we evaluate the impact of branching priorities for the two sets of binary variables x and y. Recall that the x-variables represent the first stage decisions and the y-variables represent the second stage variables. To this end, we considered four cases for analysis purposes as follows. Case I: The x-variables are modeled as continuous variables, Case II: The x-variables are modeled as binary variables according to (41), Case III: The x-variables are modeled as binary and are prioritized, and Case IV: The x-variables are modeled as binary and the y-variables are prioritized.

For the slot formulation, we find that the impact of prioritization differs in deterministic and stochastic implementations. As seen in Table 4.6.2.3, prioritization does not have a positive impact on computational performance in the deterministic case. On the other hand, when stochastic instances are considered, the prioritization of the x-variables result in slightly improved performance in run times. This interesting observation is likely due to the increase in scale and complexity for stochastic instances. As the instances get more complex, prioritization becomes more effective in the branch and bound procedure.

Similarly, we evaluate the impact of prioritization in the network based SRSP. In the initial model presented in Section 4.2.1, the first stage x-variables are modeled as continuous

Table 25: Computational results for showing the impact of prioritization for the slot based formulation of SRSP over four different cases.

Average CPU Time [s]				Median CPU Time [s]			
I	II	III	IV	I	II	III	IV
133.3	132.4	149.3	266.8	7.8	7.8	8.3	8.1

(a) Deterministic problem

Average CPU Time [s]				Median CPU Time [s]			
I	II	III	IV	I	II	III	IV
795	646	553	1070	107	95	71	122

(b) Stochastic problem with limited number of scenarios

Table 26: Computational results for showing the impact of prioritization for the network flow based formulation of SRSP over four different cases.

Average CPU Time [s]				Median CPU Time [s]			
I	II	III	IV	I	II	III	IV
26.8	28.8	113.4	31.7	8.3	8.4	10.0	8.0

(a) Deterministic problem

Average CPU Time [s]				Median CPU Time [s]			
I	II	III	IV	I	II	III	IV
787	776	1335	810	154	125	235	141

(b) Stochastic problem with limited number of scenarios

variables. These variables will be either 0 or 1 in any solution due to their relationship to the z-variables and y-variables. We model the x-variables as binary which gives the solver an opportunity to branch on these variables, in addition to the y-variables. Furthermore, we test the impact of prioritization by considering the four cases described above for the slot formulation. As part of this analysis, we find for the network formulation of SRSP that modeling the first stage x-variables as continuous is slightly more efficient than modeling them as binary, and moreover prioritization is not that effective, except when the y variables are prioritized. The impact of the different prioritization strategies for the network formulation is shown in Table 4.6.2.3. Unlike the slot formulation, prioritization plays a similar role over the deterministic and stochastic instances when the flow formulation is used.

Table 27: Computational results comparing the deterministic model of SRSP-R to the deterministic model of SRSP for the slot based formulation.

Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
707.3	139.4	47.2	7.7	83.4	82.4

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
Group 1	375.1	28.6	27.8	10.7	84.0	54.5
Group 2	891.4	129.2	122.2	31.6	87.3	99.5
Group 3	1495.3	637.0	376.8	28.1	89.3	96.8
Group 4	107.1	4.2	11.6	3.1	78.3	79.3
Group 5	345.0	8.5	39.7	5.2	80.8	84.4
Group 6	1029.9	28.8	151.0	3.8	80.7	79.9

(b) Results aggregated by groups.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
8	2.5	1.6	1.8	1.3	79.0	80.0
10	45.6	8.2	13.5	4.4	82.7	82.6
12	692.7	64.7	126.7	11.1	85.2	84.1
14	2088.4	483.1	1639.2	33.3	86.8	82.9

(c) Results aggregated by number of flights.

4.6.3 Analysis-III: Comparison of SRSP vs. SRSP-R

We use the best configurations described above to determine the relative efficiency of the SRSP-R and SRSP models. More specifically, we demonstrate that the representation of the objective function in SRSP results in significant additional computational complexity for most problem instances. Overall, if cost of delay is not important and only the total delay is to be minimized, then SRSP-R is sufficient and easier to solve. However, SAA convergence issues are still a problem for stochastic instances of SRSP-R.

In Table 4.6.3, we compare the deterministic versions of SRSP-R and SRSP based on the performance of the slot formulation. We note that deterministic instances of SRSP are solved much more efficiently than SRSP-R if the slot formulation is used. This is mainly due to the generally poor performance of the slot formulation for SRSP-R, and does not imply anything on the relative overall complexities of the two problems. The integrality gaps on the other hand are mostly at similar levels. For the stochastic slot formulations, SRSP

Table 28: Computational results comparing the SRSP-R model to the SRSP model for the slot based formulation.

Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Optimality Gap at Termination [%]	
SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
2509	541	741	98	11.8	18.4	0.5	0.2

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Opt. Gap at Termination [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
1	1855	474	786	149	17.5	45.5	0.1	0.1
2	4205	1361	2075	238	15.0	19.2	1.2	0.8
3	4362	1170	5637	396	16.8	14.7	1.3	0.1
4	215	20	123	13	4.7	8.2	0.0	0.0
5	1451	120	390	60	8.5	15.0	0.1	0.1
6	2968	100	452	36	8.2	8.0	0.1	0.0

(b) Results aggregated by groups.

Number of Scenarios	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Opt. Gap at Termination [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
8	2282	449	673	61	11.4	20.5	0.4	0.1
12	2737	633	830	125	12.1	16.4	0.5	0.3

(c) Results aggregated by number of scenarios.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Opt. Gap at Termination [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
8	561	92	344	37	11.8	12.4	0.0	0.1
10	4458	989	4018	188	11.8	24.5	0.9	0.3

(d) Results aggregated by number of flights.

again performs better despite typically taking more iterations to complete. This is a result of the efficiency achieved in solving the scenario problems in the scenario decomposition procedure. The corresponding run times are shown in Table 4.6.3.

In Table 4.6.3, we compare SRSP-R and SRSP based on the performance of the flow formulation. We note that deterministic instances of SRSP-R are solved more efficiently than that of SRSP when the flow formulation is used. In addition, the reduction in the integrality gap of SRSP-R when compared to SRSP is evident in the results. When stochastic flow formulations are considered, the situation involves some additional observations. SRSP is harder to solve as seen in the results in Table 4.6.3. On the other hand, the best upper bound is found in earlier iterations in SRSP which makes it more amenable to heuristic based approaches.

Table 29: Computational results comparing the deterministic model of SRSP-R to the deterministic model of SRSP for the flow based formulation.

Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
19.1	27.4	4.2	10.0	23.4	56.0

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
Group 1	31.8	19.4	10.6	11.3	35.0	53.2
Group 2	4.4	28.3	3.6	16.5	10.8	71.7
Group 3	3.9	39.3	2.6	13.5	8.7	26.6
Group 4	39.3	7.3	11.2	4.0	44.8	79.3
Group 5	29.4	40.0	8.3	9.5	28.2	71.0
Group 6	6.0	30.3	1.8	7.8	12.5	34.1

(b) Results aggregated by groups.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Integrality Gap [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
8	0.3	1.6	0.3	1.5	22.3	49.3
10	4.0	7.0	2.9	5.5	23.2	55.5
12	18.9	18.1	10.7	16.4	23.9	57.2
14	53.3	83.0	18.1	47.4	24.1	62.0

(c) Results aggregated by number of flights.

Table 30: Computational results comparing the SRSP-R model to the SRSP model for the network flow based formulation.

Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Optimality Gap at Termination [%]	
SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
737	786	99	135	21.2	24.8	0.1	0.2

(a) Results for all instances

Group Name	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Opt. Gap at Termination [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
1	900	1773	226	198	21.7	81.5	0.1	0.3
2	1541	1436	153	322	30.6	20.7	0.1	0.8
3	1157	856	212	277	52.8	15.4	0.1	0.1
4	146	57	41	18	4.7	8.2	0.0	0.0
5	321	342	108	163	8.5	15.0	0.1	0.1
6	360	254	28	77	8.9	8.0	0.0	0.0

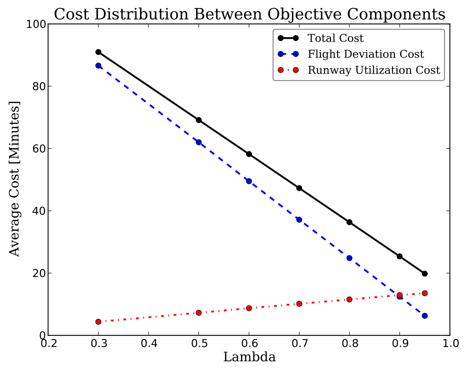
(b) Result aggregated by groups.

Number of Scenarios	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Opt. Gap at Termination [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
8	605	703	72	109	22.2	32.0	0.1	0.1
12	870	870	146	185	20.2	17.5	0.1	0.3

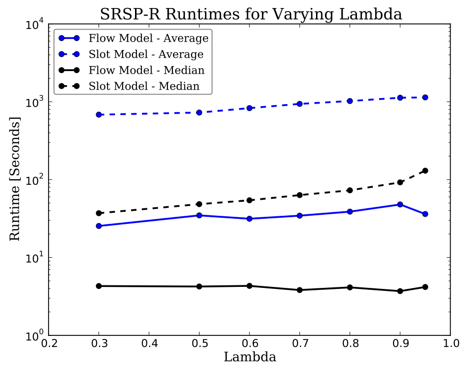
(c) Results aggregated by number of scenarios.

Number of Flights	Average CPU Time [s]		Median CPU Time [s]		Number of Iterations		Opt. Gap at Termination [%]	
	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP	SRSP-R	SRSP
8	58	98	34	70	11.9	12.4	0.0	0.1
10	1417	1474	360	488	30.5	37.2	0.1	0.4

(d) Results aggregated by number of flights.



(a) Average contribution to objective function for optimal solutions of SRSP-R for different values of λ.



(b) Average runtime for SRSP-R for different values of λ. Higher curve for each model corresponds to the averages.

Figure 14: Impact of the objective weights for SRSP-R.

It is important to consider the impact of the objective weights used in the implementations of the SRSP-R, especially with respect to the run times. The objective function used in SRSP is based on the actual monetary cost for flight deviations and the cost of runway utilization. For optimal solutions of SRSP, the flight delay cost contributes to about 37% of the total cost and the share for runway utilization is around 63%. However, as part of the objective function in SRSP-R we measure the deviation in minutes, which makes it difficult to put appropriate weights on runway utilization and flight delay in the objective. In the case with equal weights, i.e. for $\lambda = 0.5$, the resulting cost distribution is 90% for flight deviation and the remaining 10% percent is for runway utilization. This is expected, as one minute of flight delay has the same weight of runway delay. As can be seen in Figure 14a, to achieve the same cost distribution for SRSP-R as SRSP we need $\lambda > 0.9$. On the other hand, we can see from the analysis in Figure 14b that the objective values in SRSP-R are insensitive to the value of λ, likely due to the degeneracy that may exist in the problem structure. Similarly, it can be observed that the run times are also insensitive to the value of λ, and specifically the ratio of run times for flow and slot times remain the same over different values of λ. This implies that the selection of the value of λ does not play a role in our comparative analyses of different problem configurations.

4.6.4 Analysis-IV: Practical Performance

In this section, we consider the practical implementability of stochastic runway scheduling in real life operations. This requires that a "good" solution is obtained for a given stochastic instance in reasonable amount of time. Hence, we limit the run times for the SAA algorithm and compare the best solutions obtained in truncated runs with those obtained from the full runs. More specifically, we perform four truncated implementations corresponding to run time limits of 5, 10, 15 and 20 minutes.

First, in Tables 4.6.4 and 4.6.4 we compare the performance of the complete runs, implemented using a two-hour time limit, with the 20-minute implementation. The implementations are based on the slot based model which appears to be more effective for SRSP. In Table 4.6.4 we show the average number of replications (M), the average number of distinct weight class sequences that are an optimal solution in at least one replication, and the average relative adjusted optimality gap. It is interesting to note that the relative adjusted optimality gaps for the two implementations are very close, implying that high quality solutions can still be obtained with truncated run times. We also note that on average the same number of potentially optimal solutions are identified in each implementation. This supports our observations that the structure of the problem allows for identification of good solutions, even optimal solutions, in the early iterations of the algorithm. Finally, it is also notable that the relative optimality gap is reduced in both cases as the schedule densities go up. This is due to the increase in the absolute values of the objective functions as more aircraft are considered, resulting in larger values for the total cost. Hence, a lower percent gap is calculated despite the increase in the absolute gaps. This might also be an indication that the solutions obtained in less dense schedules might also be close to optimal, but the lower bounds are not tight enough to reflect this in the gap estimates.

In Table 4.6.4, we compare the estimated values of the best solutions obtained in the full and 20 minute implementations. For each instance we consider the best sequence and note the number of iterations in which the overall best sequence is the winner, the average number of samples that are performed for this sequence, the mean, the standard deviation and the upper limit of the 95% confidence interval of the mean. The closeness of the quality of the

Table 31: Computational results comparing the characteristics of the full two hour algorithm to a truncated version that solves the problem in 20 minutes. The slot based model is used in the implementations.

Average Number of Iterations		Average Number of Potential Sequences		Relative Adjusted Optimality Gap [%]	
Full	20 min	Full	20 min	Full	20 min
51.9	23.0	4.5	3.5	5.65	6.14

(a) Results for all instances

Group Name	Average Number of Iterations		Average Number of Potential Sequences		Relative Adjusted Optimality Gap [%]	
	Full	20 min	Full	20 min	Full	20 min
Group 1	54.1	24.1	8.3	5.9	11.49	12.62
Group 2	24.9	12.6	4.4	3.4	6.55	7.33
Group 3	15.4	8.9	4.3	3.8	3.06	3.17
Group 4	89.6	45.0	2.1	1.6	3.72	4.17
Group 5	68.1	25.2	5.9	4.4	7.24	7.40
Group 6	59.4	22.3	2.3	2.0	1.86	2.12

(b) Results aggregated by groups.

Number of Scenarios	Average Number of Iterations		Average Number of Potential Sequences		Relative Adjusted Optimality Gap [%]	
	Full	20 min	Full	20 min	Full	20 min
8	57.0	26.6	5.0	3.7	6.22	6.70
12	46.8	19.4	4.0	3.3	5.09	5.57

(c) Results aggregated by number of scenarios.

Number of Flights	Average Number of Iterations		Average Number of Potential Sequences		Relative Adjusted Optimality Gap [%]	
	Full	20 min	Full	20 min	Full	20 min
8	72.0	29.9	4.2	3.4	4.83	5.13
10	31.8	16.0	4.8	3.6	6.48	7.14

(d) Results aggregated by number of flights.

solutions is evident, especially based on the mean values. The only difference is in the upper bounds for the 95% confidence interval on the objective function value estimates, where the full implementation results in lower values as expected. However, even this difference is not significant. The robustness in the quality of the solutions obtained through the truncated implementation is observable over different configurations of schedules, number of flights and number of scenarios. Overall, marginal impact of additional run times is very minimal, implying that the algorithm can potentially be implemented for practical use through truncated run times.

Given the efficiency and effectiveness of truncated runs based on a 20 minute limit, a

Table 32: Computational results comparing the best sequence found in the full two hour algorithm compared to the best sequence found in the truncated version that solves the problem in 20 minutes.

Average Number of Samples		Average Mean		Average Std. Deviation		Average Upper Confidence Interval on Mean	
Full	20 min	Full	20 min	Full	20 min	Full	20 min
7250	3267	1238.7	1239.4	189.0	186.9	1245.3	1248.0

(a) Results for all instances

Group Name	Average Number of Samples		Average Mean		Average Std. Deviation		Average Upper Confidence Interval on Mean	
	Full	20 min	Full	20 min	Full	20 min	Full	20 min
Group 1	4820	2320	236.2	235.5	87.7	89.8	240.7	241.2
Group 2	3200	1670	787.7	791.3	179.3	180.9	795.9	802.2
Group 3	1590	780	2499.6	2502.4	216.7	217.4	2512.3	2516.5
Group 4	16510	8450	255.3	255.2	119.5	119.7	256.8	257.7
Group 5	6470	2310	832.9	830.8	257.8	246.1	839.5	842.3
Group 6	10910	4070	2820.4	2821.3	216.2	215.8	2826.3	2828.4

(b) Results aggregated by groups.

Number of Scenarios	Average Number of Samples		Average Mean		Average Std. Deviation		Average Upper Confidence Interval on Mean	
	Full	20 min	Full	20 min	Full	20 min	Full	20 min
8	7690	3713	1238.8	1239.4	188.2	187.1	1244.8	1247.9
12	6810	2820	1238.6	1239.4	189.7	186.6	1245.7	1248.1

(c) Results aggregated by number of scenarios.

Number of Flights	Average Number of Samples		Average Mean		Average Std. Deviation		Average Upper Confidence Interval on Mean	
	Full	20 min	Full	20 min	Full	20 min	Full	20 min
8	10130	4370	1137.0	1136.7	165.8	165.6	1140.7	1142.3
10	4370	2163	1340.4	1342.1	209.6	206.0	1349.8	1353.8

(d) Results aggregated by number of flights.

Table 33: Computational results comparing the impact of runtime for the truncated SAA algorithm implementations.

Average Mean				Average Std. Deviation				Average Upper Confidence Interval on Mean			
I	II	III	IV	I	II	III	IV	I	II	III	IV
1239.4	1238.7	1239.4	1240.7	186.9	184.6	187.6	196.5	1248.0	1248.2	1249.5	1253.4

Table 34: Computational results showing the percentage of instances for which the best sequence, as identified by the 2 hour run, is found. The increase, relative to the 2 hour run, in the upper confidence interval limit is also shown for cases where the best sequence is the same and when it is not.

	I	II	III	IV
Instances Where Best Sequence is Found	82.5%	85.8%	82.5%	73.3%
Increase in Optimal Obj. Value Est. When Best Sequence is Found	1.3	2.2	2.6	5.0
Increase in Optimal Obj. Value Est. When Best Sequence is Not Found	9.6	7.1	11.7	16.6

natural question involves the efficiency of even shorter run times. This is especially of practical relevance given the potentially faster decision making process in scheduling runway operations, specifically when the schedule densities are high. In Table 4.6.4, we report the performances of truncated implementations based on four different run times, where the notation is defined as I: 20 minute runtime, II: 15 minute runtime, III: 10 minute runtime, and IV: 5 minute runtime. For each case we look at the best sequence and note the mean objective function value, the standard deviation and the upper limit of the 95% confidence interval of the mean. The results are very promising in terms of the practical implementability of the procedures described in this chapter. The difference between the 20 minute and 5 minute runs is very small, and the level of robustness is again the same across different configurations. As we are looking at the best sequence that is found for each runtime scenario, we note that if the same sequence is found in versions with different run times, the only difference to the result presented in Table 4.6.4 is the number of samples we can afford to use to get an estimate of the cost. In Table 4.6.4, we show the percentage of instances in which the best sequence, as identified by the full 2 hour run, is found. In the same table we have also separated the cases to evaluate the impact of reduced number of samples in truncated runs, as given by the increase in the obtained objective function value estimate with respect to the estimate from the full run. All indications suggest that the problem formulations and solution methodologies proposed result in effective implementations which mostly produce high quality solutions in very short run times.

CHAPTER V

SINGLE-STAGE STOCHASTIC SEQUENCING

In the previous chapter we presented a stochastic integer program for a certain class of machine scheduling problem and the airport runway scheduling problem. The two-stage decision process was motivated by the observation that the makespan of a sequence depend on the job categories rather than characteristics of individual jobs (aircraft) when demand exceeds capacity. As is the case for general stochastic integer programs, we assumed a discrete probability distribution modeling the uncertainty.

A different approach to stochastic integer programming is taken in the stochastic branch and bound algorithm, which is a Monte Carlo based approached to find a good (ideally optimal) solution while taking uncertainty into account. The idea of the algorithm is to divide the solution space into smaller subsets and estimate stochastic bounds on the objective function value in each subset. The subsets are organized into a branching tree. In contrast to deterministic branch and bound models, branches of the tree cannot be deterministically pruned since the true bounds are not known. Another significant difference is that there is no requirement on the probability distribution, other than that samples can be drawn from it. The stochastic branch and bound algorithm is well suited for sequencing problems due to the natural partitioning of the solution space based on positions in the sequence. Since there is only one level of decisions, namely the sequence, the method can be seen as a single stage stochastic model.

Due to the many applications in manufacturing, in particular machine scheduling, it is common to frame sequencing and scheduling problems in terms of machine scheduling notation. We usually consider n jobs which are to be scheduled on m machines. The most common configuration is to consider a single machine.

When a single machine is considered, we refer to the *sequence* as the order in which jobs are processed, and denote this by $\mathbf{x}$. The *schedule* refers to the timing of each job. For our purpose we are interested in the start time of each job, denoted by $\mathbf{t} = (t_1, \ldots, t_n)$, but it is also common to consider the completion time of the jobs, $\mathbf{C}$. We assume that the schedule $\mathbf{t}$ can easily be found given a sequence $\mathbf{x}$, either through a direct calculation or by solving a simple auxiliary problem. In the other direction, a schedule implies a sequence.

To simplify the categorization of different types of machine scheduling problems, Graham, Lawler, Lenstra, and Kan (1979) established the three-field notation $\alpha|\beta|\gamma$ to describe the scheduling problem at hand. The first parameter, α, describes the machine environment, the second parameter, β describes the job characteristics, and the γ parameter indicates what objective is minimized. As we are dealing with the single machine problem we always use $\alpha = 1$. A common job attribute is the release time r_i. It is also common to specify a due date d_i, a time by which a job needs to be completed without incurring a penalty. Some common objectives are to minimize the last completion time, $C^{\max}$, or to minimize the sum of tardiness, $\sum \Delta_i$, where $\Delta_i = \max\{0, C_i - d_i\}$.

In this chapter we consider a machine scheduling problem on a single machine where the processing time for each job, p_i, is zero, i.e. $p_i = 0$, $i \in \{1, \ldots, n\}$. The due date of each job coincide with the release time, $r_i = d_i$, implying that we ideally would like to schedule the job as soon as it is released. Furthermore, we have sequence-dependent setup times that enforce that there is sufficient amount of time between the completion time of job i, and the start time of job j. We denote this time as $s_{i,j}$. Two different objectives are used. Maximizing the utilization of the machine is equivalent to minimizing the completion time of the last job, $C^{\max}$. The delay for each job, or the tardiness, is computed as $\Delta_i = \max\{0, C_i - d_i\} = \max\{0, t_i + p_i - r_i\} = t_i - r_i$, hence the second objective is to minimize $\sum_{i=1}^{N} \Delta_i$. Using the three-field notation, the studied problems are of the form $1|r_i = d_i, s_{i,j}|C^{max}$ and $1|r_i = d_i, s_{i,j}| \sum_i \Delta_i$.

Note that the special case where $r_i = 0$, $\forall i \in \{1, \ldots, n\}$, $s_{i,j} = s_{j,i}$ and the $C^{\max}$ objective is used is an instance of the symmetric traveling salesman problem. Special cases where the $\sum_{i=1}^{N} \Delta_i$ objective is used can be reduced to the traveling repairman problem (Heilporn, Cordeau, and Laporte 2010b). These special cases show that both versions of the machine scheduling problem under consideration are NP-hard (Blum, Chalasani, Coppersmith, Pulleyblank, Raghavan, and Sudan 1994).

The machine scheduling problem presented above is a generalization of the airport runway scheduling problem. Clearly, sequencing jobs on a machine correspond to sequencing aircraft on one or multiple dependent runways. The separation requirements between aircraft can be seen as sequence-dependent setup times. Furthermore, the earliest time an aircraft is available for scheduling on the runway corresponds to the release time.

In this chapter we propose several enhancements to the stochastic branch and bound algorithm. We investigate the computational impact of the proposed enhancements and apply the methodology to applications in airport runway scheduling.

The chapter is structured as follows: In Section 5.1 the stochastic branch and bound algorithm tailored to sequencing problems is presented and the proposed enhancements are discussed in Section 5.2. The remainder of the chapter is divided into two parts. In the first part, we present the implementation details for the general machine scheduling problem in Section 5.3. The impacts of the proposed enhancements are evaluated in Section 5.4. In the second part we shift focus to the runway scheduling problem. Due to additional complexities in the runway scheduling problem, a new method to estimate upper and lower bounds is presented in Section 5.5. The computational study for the runway scheduling problem can be found in Section 5.6.

5.1 Stochastic Branch and Bound

Let $\mathcal{I} = \{1, \ldots, n\}$ be the set of jobs that are considered. We want to find a sequence $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is the set of all possible job sequences, such that the cost of the sequence, $f(\mathbf{x})$, is minimized. As we are considering probabilistic release times we denote these by R_i, $i \in \mathcal{I}$, indicating that the release times are random variables. We let $\mathbf{R} = (R_1, \ldots, R_n)$. All random variables have distributions $P_1, \ldots, P_n$, with the requirement that samples can be drawn from these distributions. To explicitly include the dependency of uncertainty, the objective is rewritten as $f(\mathbf{x}, \mathbf{r}^\omega) = f(\mathbf{x}, \omega)$, where $\omega \in \Omega$ and $(\Omega, \Sigma, \mathbb{P})$ is the probability space. The vector $\mathbf{r}^\omega = (r_1^\omega, \ldots, r_n^\omega)$ gives the realization of the random variables $\mathbf{R}$ in scenario ω. The start times of the jobs are given in the vector $\mathbf{t} = (t_1, \ldots, t_n)$.

Formally, we have

$$\begin{aligned} f(\mathbf{x}, \omega) = \min \quad & g(\mathbf{t}, \mathbf{r}^\omega) \\ \text{s.t.} \quad & t_j \geq t_i + s_{i,j} \quad \text{if } i \text{ is before } j \text{ in } \mathbf{x},\ \forall i, j \in \mathcal{I} \times \mathcal{I},\ \ i \neq j \\ & t_i \geq r_i^\omega \qquad\quad\ \forall i \in \mathcal{I} \\ & \mathbf{t} \in \mathbb{R}_+ \end{aligned} \tag{60}$$

The function $g(\mathbf{t}, \mathbf{r}^\omega)$ denotes the cost of the schedule. The two different objective versions are:

$$g(\mathbf{t}, \mathbf{r}^\omega) = \max_{i \in \mathcal{I}} t_i \tag{61}$$

$$g(\mathbf{t}, \mathbf{r}^\omega) = \sum_{i \in \mathcal{I}} \Delta_i = \sum_{i \in \mathcal{I}} (t_i - r_i^\omega) \tag{62}$$

Given the uncertainty in the release times, we would like to find a sequence such that the expected cost is minimized. We define the overall problem as

$$\min_{\mathbf{x} \in \mathcal{X}} F(\mathbf{x}) \text{ where } F(\mathbf{x}) = \mathbb{E}(f(\mathbf{x}, \omega)) = \int_\Omega f(\mathbf{x}, \omega)\, \mathbb{P}(d\omega) \tag{63}$$

The set $\mathcal{X}$ denotes all possible complete sequences, with no further restrictions. A complete sequence $\mathbf{x}$ is defined as $\mathbf{x} = [x(1), \ldots, x(n)]$ where $x(i)$ is the job assigned to position i. We also introduce the notion of incomplete sequences. Let $\mathcal{X}_{x(1),\ldots,x(k)}$, where $k \leq n$, be the set of sequences where positions $1, \ldots, k$ have been fixed to use job $x(i)$ in position $i = 1, \ldots, k$. As an example, consider a set of jobs $\mathcal{I} = \{1, 2, 3, 4\}$. The set $\mathcal{X}$ consists of all 4! permutations of the possible sequences. One such element of $\mathcal{X}$ is $\mathbf{x} = (1, 3, 4, 2)$. The set $\mathcal{X}_{2,1}$ consist of two sequences, namely $\mathcal{X}_{2,1} = \{(2, 1, 3, 4), (2, 1, 4, 3)\}$.

5.1.1 Algorithm

The main idea of the stochastic branch and bound algorithm is to:

- Partition subsets of sequences into smaller subsets.
- Estimate lower and upper bounds on the objective in each subset.

- Remove subsets that are not promising.

Before the algorithm is presented, some additional notation is needed. Let $\mathcal{X}^s, s = \{1, 2, \dots\}$, be the subsets that currently divide $\mathcal{X}$, so that $\mathcal{X} = \cup_s \mathcal{X}^s$. At iteration q of the algorithm, $\mathcal{X}$ is divided into partition $\mathcal{P}^q$, where the partition is made up of subsets $\mathcal{X}^s \in \mathcal{P}^q$, $s \in \{1, \dots, w_q\}$.

For a subset $\mathcal{X}^s$, let $F^*(\mathcal{X}^s) = \min_{\mathbf{x} \in \mathcal{X}^s} F(\mathbf{x})$ be the optimal objective in that subset. It is assumed that there exists bounds on the optimal objective in each subset, i.e. $L(\mathcal{X}^s) \leq F^*(\mathcal{X}^s) \leq U(\mathcal{X}^s)$. In addition, if $\mathcal{X}^s$ is singleton, then $L(\mathcal{X}^s) = F^*(\mathcal{X}^s) = U(\mathcal{X}^s)$. It is also assumed that for each subset $\mathcal{X}^s$, there exist a sequence of random estimates, $\xi^l(\mathcal{X}^s), l = 0, 1, \dots$ and $\eta^m(\mathcal{X}^s), m = 0, 1, \dots$ of $L(\mathcal{X}^s)$ and $U(\mathcal{X}^s)$ respectively, such that:

$$\xi^l(\mathcal{X}^s) \xrightarrow[l \to \infty]{a.s.} L(\mathcal{X}^s)$$
$$\eta^m(\mathcal{X}^s) \xrightarrow[m \to \infty]{a.s.} U(\mathcal{X}^s)$$

In the presentation of the algorithm we let $\xi^{l(q)}(\mathcal{X}^s)$ denote the estimate of the lower bound in iteration q. Thus, the quantity $l(q)\,[\mathcal{X}^s]$ denotes the index in the overall sequence $\xi^l(\mathcal{X}^s), l = 0, 1, \dots$ to which the estimate in iteration q is inserted. Analogous notation is used for the upper bound estimates $\eta^{m(q)}(\mathcal{X}^s)$ and the sequence of upper bounds $\eta^m(\mathcal{X}^s), m = 0, 1, \dots$.

The Stochastic Branch and Bound algorithm is stated in Algorithm 1.

5.1.1.1 Lower Bound Estimation

To obtain a lower bound $L(\mathcal{X}^s)$ on $F^*(\mathcal{X}^s)$ we interchange the expectation and minimization operators as suggested in Norkin, Pflug, and Ruszczyński (1998).

$$F^*(\mathcal{X}^s) = \min_{\mathbf{x} \in \mathcal{X}^s} \mathbb{E}(f(\mathbf{x}, \omega)) \geq \mathbb{E}(\min_{\mathbf{x} \in \mathcal{X}^s} f(\mathbf{x}, \omega)) = L(\mathcal{X}^s) \tag{64}$$

$\mathbb{E}(\min_{\mathbf{x} \in \mathcal{X}^s} f(\mathbf{x}, \omega))$ can be estimated by sampling $N = N\,[l]\,(\mathcal{X}^s)$ realizations $\omega_1, \dots, \omega_N$. Now,

$$\xi^l(\mathcal{X}^s) = \frac{1}{N} \sum_{i=1}^{N} \min_{\mathbf{x} \in \mathcal{X}^s} f(\mathbf{x}, \omega_i) \tag{65}$$

Algorithm 1 Stochastic Branch and Bound Algorithm for Sequencing Problems

```
$\mathcal{P}^0 = \mathcal{X}$
Set $q = 0$
while Stopping criterion is not met do
  Select the subset with lowest bound, $\mathcal{Y}^q = \arg\min\{\xi_q(\mathcal{X}^s)|\mathcal{X}^s \in \mathcal{P}^q\}$
  Select an approximate solution $\mathbf{x}^q \in \mathcal{X}^q \in \arg\min\{\eta_q(\mathcal{X}^s)|\mathcal{X}^s \in \mathcal{P}^q\}$
  if $\mathcal{Y}^q$ is singleton then
    Set $\mathcal{P}^{q+1} = \mathcal{P}^q$
  else
    Construct a partition of the lowest bound subset $\mathcal{Y}^q = \mathcal{X}_{x(1),x(2),\ldots,x(k)}$:
      Let $\mathcal{P}_q^{'}(\mathcal{Y}^q) = \{\mathcal{X}_{x(1),x(2),\ldots,x(k),j}|j \in A, j \neq x(1),\ldots,x(k)\}$
    Construct the new full partition:
      $\mathcal{P}^{q+1} = (\mathcal{P}^q \setminus \mathcal{Y}^q) \cup \mathcal{P}_q^{'}$
  end if
  for all subsets $\mathcal{X}^s \in \mathcal{P}^{q+1}$ do
    Estimate stochastic lower bound, $\xi_q(\mathcal{X}^s) = \xi^{l(q)}(\mathcal{X}^s)$
    Estimate stochastic upper bound, $\eta_q(\mathcal{X}^s) = \eta^{m(q)}(\mathcal{X}^s)$
  end for
  Set $q = q + 1$
end while
```

where $N\,[l]\,(\mathcal{X}^s)$ is the number of samples used for estimate l in subset $\mathcal{X}^s$, is an unbiased estimator of $L(\mathcal{X}^s)$.

5.1.1.2 Upper Bound Estimation

As is the case in all minimization problems, any feasible solution to the problem forms an upper bound on the objective. Let $\mathbf{x}^* \in \mathcal{X}^s$ be a feasible point in the subset $\mathcal{X}^s$, so $U(\mathcal{X}^s) = F(\mathbf{x}^*)$. Similar to the lower bounds, the upper bounds can be estimated by sampling $M = M\,[m]\,(\mathcal{X}^s)$ realizations $\omega_1, \ldots, \omega_M$. Now,

$$\eta^m(\mathcal{X}^s) = \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}^*, \omega_i) \tag{66}$$

is an unbiased estimator of $U(\mathcal{X}^s)$. $M\,[m]\,(\mathcal{X}^s)$ denotes the number of samples used in estimate m for subset $\mathcal{X}^s$.

A natural candidate for an upper bound in a general sequencing problem is the solution to the deterministic expected value problem $\min_{\mathbf{x}\in\mathcal{X}^s} f(\mathbf{x}, \mathbb{E}(\mathbf{R}))$. For the single machine scheduling problem with sequence-dependent setup times and release times, we suggest a simpler, and faster, approach. From subset $\mathcal{X}^s = \mathcal{X}_{x(1),\ldots,x(k)}$, fix the beginning of the

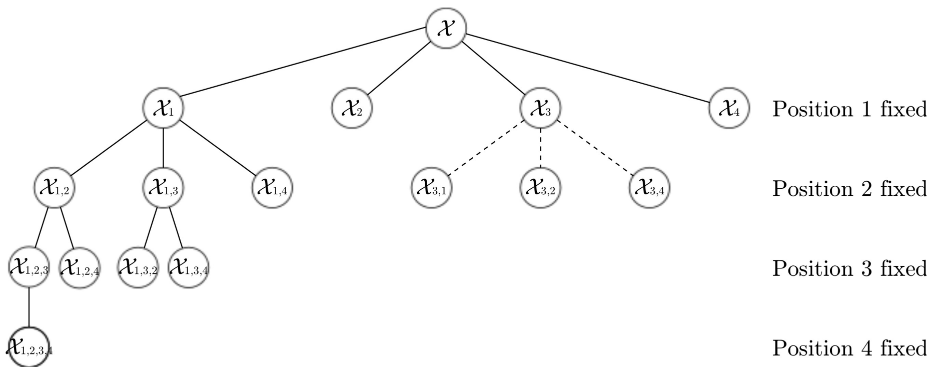


Figure 15: Example of an incomplete branching tree for 4 jobs.

sequence according to $x(1), \ldots, x(k)$. The sequence of the remaining $n-k$ jobs is obtained by taking the expected value, $\mathbb{E}(\mathbf{R}_i)$, for each job i that is not fixed and arrange these jobs in increasing order of expected release time.

5.1.1.3 Convergence

For the algorithm to converge to an optimal solution we need the bound estimates to converge (Norkin, Ermoliev, and Ruszczyński 1998). This can be achieved by successively increasing the number of bound estimates in subsets that are no longer partitioned, i.e. $\lim_{q\to\infty} l(q)\,[\mathcal{X}'] = \lim_{q\to\infty} m(q)\,[\mathcal{X}'] = \infty$, where $\mathcal{X}'$ is a subset that is an element of $\mathcal{P}^q$ for infinitely many q. Gutjahr, Hellmayr, and Pflug (1999) discuss how bound estimates can be reused in consecutive iterations by subsets that are not partitioned. In this application we use this idea and generate an initial set of observations for newly created subsets and add more observations to subsets that remain unchanged from the previous iteration.

5.1.2 Partitioning

We are looking to identify the best sequence of operations among all possible sequences $\mathcal{X}$. The set $\mathcal{X}$ is partitioned into subsets $\mathcal{X}^s$ by considering partial sequences. Consider a subset $\mathcal{X}^s = \mathcal{X}_{x(1),\ldots,x(k)}$ defined by the partial $(k < n)$ sequence $\mathbf{x} = [x(1), \ldots, x(k)]$. The subset can be partitioned into $n-k$ new subsets $\mathcal{X}^{s'} = \mathcal{X}_{x(1),\ldots,x(k),x(k+1)}$ where $x(k+1) \in \{1, \ldots, n\} \setminus \{x(1), \ldots, x(k)\}$.

An example is provided in Figure 15 where the subsets $\mathcal{X}^s \in \mathcal{P}^q$, $s \in \{1, \ldots, w_q\}$ are represented in a tree structure. With the tree structure in place it is natural to refer to the subsets as *nodes.* Partitioning subset $\mathcal{X}_3$ leads to 3 new subsets (nodes), $\mathcal{X}_{3,1}, \mathcal{X}_{3,2}$, and $\mathcal{X}_{3,4}$.

5.2 Algorithm Enhancements

In contrast to many deterministic branch and bound schemes, the node selection rule for branching is defined as a part of the algorithm. The algorithm always selects the node with the least lower bound estimate. This way the part of the branching tree which shows promising subsets is partitioned and explored in greater detail. Another significant difference from the deterministic branch and bound algorithm is that a node in the stochastic version cannot be pruned solely based on the condition that the lower bound is worse (greater) than the best upper bound. The bounds calculated in the algorithm are only estimates of the true lower and upper bounds, and pruning based on these may lead to the elimination of the optimal solution.

Branches in the tree can however be deterministically pruned if it is possible to get exact upper and lower bounds. This is the case when the probability distributions $P_1, \ldots, P_n$ are bounded and the variance of the resulting random variables $\xi^l(\mathcal{X}^s)$ and $\eta^m(\mathcal{X}^s)$ can be bounded. One such deletion rule is presented in Norkin, Pflug, and Ruszczyński (1998).

In contrast to a pruning rule, we present a strategy to dynamically change the computational emphasis, i.e. the number of samples used, for various parts of the branching tree. By focusing the bound estimation on subsets that are likely to contain good or optimal solutions, we can significantly improve the performance of the algorithm.

5.2.1 Dynamic Sample Size Update and Statistical Testing

In many cases it is difficult to find tight upper and lower bounds on the stochastic variables $\xi^l(\mathcal{X}^s)$ and $\eta^m(\mathcal{X}^s)$ to allow for deterministic pruning in the stochastic branch and bound algorithm. As an alternative, we propose an approach that decreases the number of samples used to evaluate lower bounds in non-promising nodes. The same approach is used to determine the number of samples for upper bound estimation.

In the following discussion we focus on the lower bounds, but we treat the upper bounds in the exact same way, thus keeping the sample size for lower and upper bounds equal.

Letting $\varphi(\mathcal{X}^s)$ (when it is clear from the context which subset is considered we simply use φ) be the iteration in which a given subset X^s is created, we set the initial sample size $N\left[l(\varphi)\right](X^s) = M\left[m(\varphi)\right](X^s) = N^0 \geq 1$ and estimate lower and upper bounds $\xi^{l(\varphi)}(\mathcal{X}^s)$ and $\eta^{m(\varphi)}(\mathcal{X}^s)$ according to Equation (65) and Equation (66). For the remaining active subsets $\{\mathcal{X}^t \in \mathcal{P}^{\varphi} \setminus X^s\}$, we update the lower bound estimate by adding $N\left[l(\varphi)\right](X^t)$ samples to the estimate obtained in previous iterations and calculate the new estimate according to Equation (67). In the subsequent iteration q, the sample size used in X^s is adjusted to a new value $N^1 > 0$ (typically smaller than N^0). Note that it is possible that $0 < N^1 < 1$. In that case we force N^1 to be of the form $1/\delta$ and let $N\left[l(q)\right](\mathcal{X}^s) = 1$ every δ iteration and 0 in the intermediate iterations. Letting $N'\left[l\right](\mathcal{X}^s)$ be the total number of samples included in estimate l, we have:

$$\xi^{l(q)}(\mathcal{X}^s) = \frac{N'\left[l(q-1)\right](\mathcal{X}^s)\xi^{l(q-1)}(\mathcal{X}^s) + \sum_{i=1}^{N[l(q)](\mathcal{X}^s)} \min_{\mathbf{x}\in\mathcal{X}^s} f(\mathbf{x},\omega_i)}{N'\left[l(q-1)\right](\mathcal{X}^s) + N\left[l(q)\right](\mathcal{X}^s)} \tag{67}$$

During the course of the algorithm, $\xi^{l(q)}(\mathcal{X}^s)$ is updated in every iteration (except when $N^1 < 1$, in which case it is updated every δ iteration) for all active subsets. The computational effort required to update the bounds is proportional to the number of samples used. For subsets that are unlikely to contain the optimal (or a good) solution, we successively decrease the number of samples as long as it is possible to do so. We ensure that $N\left[l(q)\right](\mathcal{X}^s) > N^1_{min} > 0$.

At every q^E iteration, all active nodes are compared to the node with the best upper bound. More specifically, given iteration q in which the sample sizes are reviewed, we have the subset with least upper bound, $\bar{\mathcal{X}}^q \in \arg\min\{\eta^q(\mathcal{X}^s)|\mathcal{X}^s \in \mathcal{P}^q\}$. For all other active subsets $\{\mathcal{X}^t \in \mathcal{P}^q \setminus \bar{\mathcal{X}}^q\}$ we compare the lower bound $\xi^q(X^t)$ to $\eta^q(\bar{\mathcal{X}}^q)$ using a statistical hypothesis test to assert if we can reject the null hypothesis $H_0 : \eta^q(\bar{\mathcal{X}}^q) < \xi^q(X^t)$. We use the two-sample t-test for equal mean (NIST/SEMATECH 2012). Depending on the outcome of the test, we update the sample size as

$$N\left[l(q)\right](\mathcal{X}^s) = \begin{cases} \max\left(N\left[l(q-1)\right](\mathcal{X}^s) - N^-, 1\right) & \text{if we fail to reject } H_0 \\ & \text{and if } N\left[l(q-1)\right](\mathcal{X}^s) > 1 \\ \max\left(1/(\delta + N^-), N^1_{\min}\right) & \text{if we fail to reject } H_0 \text{ and if} \\ & N\left[l(q-1)\right](\mathcal{X}^s) = 1/\delta \leq 1 \\ N\left[l(q-1)\right](\mathcal{X}^s) & \text{otherwise} \end{cases} \tag{68}$$

where N^- is the decrease in sample size and N^1_{min} is the minimum sample size. The two-sampled t-test for equal mean is used with 5% significance level.

Proposition 7. *The stochastic branch and bound algorithm converges using the sample size update rule specified in Equation* (68).

Proof. Lemma 1 in Norkin, Ermoliev, and Ruszczyński (1998) states that the convergence criteria are satisfied if the total number of observations for estimate $l(q)$ is of the form

$$N'[l(q)](\mathcal{X}^s, \omega) = N'[l(q-1)](\mathcal{X}^s, \omega) + I_q(\mathcal{X}^s, \omega)$$

with $N'[l(q-1)](\mathcal{X}^s, \omega) = 0$ if $\mathcal{X}^s$ is newly created, *and* if

$$\sum_{q=\varphi(\mathcal{X}^s)}^{\infty} \mathbb{P}\{I_q(\mathcal{X}^s, \omega) > 0 | \mathcal{X}^s, N'[l(q-1)](\mathcal{X}^s, \omega)\} = \infty \tag{69}$$

for every $\mathcal{X}^s$ in the current partition. $\varphi(\mathcal{X}^s)$ is the iteration in which $\mathcal{X}^s$ was created. An identical condition exists for upper bound sample sizes.

Using Equation (68), we have $I_q(\mathcal{X}^s, \omega) = N\left[l(q)\right](\mathcal{X}^s)$, so that $\mathbb{P}\{I_q(\mathcal{X}^s, \omega) > 0 | \mathcal{X}^s, N'[l(q-1)](\mathcal{X}^s, \omega)\} \geq N^1_{min} > 0$ and Equation (69) is satisfied. An identical argument can be used for the upper bounds. □

5.2.2 Termination Criteria

The stochastic branch and bound algorithm does not have a clearly defined termination criteria. Termination criteria used in other applications of the algorithm include iteration count (Gutjahr, Hellmayr, and Pflug 1999), stopping when the first singleton node is found

(Norkin, Ermoliev, and Ruszczyński 1998), and total runtime (Gutjahr, Strauss, and Wagner 2000). We introduce an additional criteria that terminates the algorithm when the node with the least lower bound appear to be recurrent, i.e. when the same node repeatedly gets selected as the node with least lower bound estimate. Note that only singleton nodes can be selected multiple times, as non-singleton nodes are partitioned if they contain the least lower bound estimate. We let the parameter q^T denote the maximum number of consecutive iterations where the node with the least lower bound estimate remains the same.

5.2.3 Heuristic Solution Methods

As is the case for all branch and bound algorithms, the overall problem is simplified by considering a subset of the full solution space. Nevertheless, to obtain exact lower bounds for the expected cost in a given node, an NP-hard problem needs to be solved for each lower bound estimate. Convergence of the algorithm to an optimal solution requires a very large number of lower bound estimates. Even if the algorithm is terminated with an approximate solution, a large number of estimates are required for qualitative solutions. To that end, we use heuristic methods to estimate lower bounds. Note that by using a heuristic solution in the lower bound estimate we may miss the true lower bound, and therefore overestimate the lower bound. Overestimation of the lower bound may lead to a subset not being selected as the least lowest bound subset, and therefore not partitioned. On the other hand, by using heuristic solution methods, many more samples can be used in an estimate, thus improving the quality of the estimate. It has been reported (Gutjahr, Strauss, and Wagner 2000) that heuristic lower bounds in certain cases outperform the traditional version of the stochastic branch and bound algorithm where exact lower bounds are used. Furthermore, the underlying structure of the deterministic problem to be solved can call for the use of heuristic solution methods. This is the case for $1|r_i = d_i, s_{i,j}|C^{max}$ and $1|r_i = d_i, s_{i,j}| \sum_i \Delta_i$. The heuristic solution methods are presented in Section 5.3.

5.3 Implementation Details

5.3.1 Upper Bound Estimation

To estimate an upper bound on the objective $\min_{\mathbf{x}\in\mathcal{X}^s} F(\mathbf{x})$ in subset $\mathcal{X}^s = \mathcal{X}_{x(1),\ldots,x(k)}$ we use Equation (66) to get an unbiased estimator of $U(\mathcal{X}^s)$. We let $\mathbf{x}^* \in \mathcal{X}^s$ be a feasible point, with the intention that $U(\mathcal{X}^s) = F(\mathbf{x}^*)$ gives a good, i.e. a small, upper bound to $F(\mathcal{X}^s)$. In Section 5.1.1.2, we described how $\mathbf{x}^* = [x(1),\ldots,x(k),x(k+1),\ldots,x(n)]$ is obtained by fixing the first k positions based on the current subset and order the remaining jobs in increasing order of expected release time, $\mathbb{E}(\mathbf{R}_i)$, $\forall i \in \mathcal{I}\setminus\{x(1),\ldots,x(k)\}$, such that $\mathbb{E}(\mathbf{R}_{x(k+1)}) \leq \cdots \leq \mathbb{E}(\mathbf{R}_{x(n)})$.

Given a complete sequence $\mathbf{x}^*$ we can calculate the start times $\mathbf{t} = (t_1,\ldots,t_n)$ by the simple recursion described in Procedure 1. The objective function value is calculated as $f(\mathbf{x}^*,\omega) = g(\mathbf{t},\mathbf{r}^\omega)$. The function g is defined in Equation (61) or Equation (62), depending what version of the problem we are solving.

Procedure 1. *Given a sequence of jobs, $\mathbf{x} = [x(1),\ldots,x(\ell)]$, of length ℓ and with attributes r_i^ω denoting the realized release time, the following procedure is used to calculate the start time of each job.*

```
Given initial job $x(0)$ scheduled at time $t_{x(0)}$
for $i = 1 \to \ell$ do
    $t_{x(i)} = \max(r^\omega_{x(i)}, t_{x(i-1)} + s_{x(i-1),x(i)})$
end for
```

5.3.2 Lower Bound Estimation

To estimate a lower bound $L(\mathcal{X}^s)$ on the true optimal objective value $F^*(\mathcal{X}^s)$ we use Equation (65) to calculate an unbiased estimator of $L(\mathcal{X}^s)$. In doing so we need to solve the problem

$$\min_{\mathbf{x}\in\mathcal{X}^s} f(\mathbf{x},\omega) \tag{70}$$

Letting $\mathcal{X}^s = \mathcal{X}_{x(1),\ldots,x(k)}$, the first k components of the sequence are fixed, and the corresponding start times can be calculated using Procedure 1. For the remaining jobs we need to find a sequence such that $f(\mathbf{x}, \omega)$, defined in Equation (60), is minimized.

The problem defined in Equation (60) can be modeled as a mixed integer linear program. Note that the constraint $t_j \geq t_i + s_{i,j}$ is only valid if job i is sequenced before job j. Therefore, a mixed integer program requires a "big-M" construction, effectively disabling the constraint if job j is scheduled before job i. These kinds of formulations typically show a poor linear relaxation (Codato and Fischetti 2006b) and are, therefore, notoriously hard to solve using commercial solvers for mixed integer programming.

To solve the problem $\min_{\mathbf{x} \in \mathcal{X}^s} f(\mathbf{x}, \omega)$ we use a dynamic programming algorithm where each job can only be shifted by a pre-specified number of positions from a nominal job sequence. The nominal sequence is constructed by considering release times for each job and order the jobs in increasing order of earliest release time. The algorithm, which is known as a Constrained Position Shift (CPS) heuristic, has been extensively studied for aircraft scheduling problems and used to obtain good quality solutions in a short amount of time (Balakrishan and Chandran 2010; Dear and Sherif 1991; Psaraftis 1980; Trivizas 1998). We use the formulations in Balakrishan and Chandran (2010), where the C^{max} objective version can be solved in time that is polynomial in the number of jobs n. Note that the algorithm is exponential with respect to τ, the parameter describing the maximum allowable number of position shift, but τ is typically small (1,2, or 3).

The problem $1|r_i = d_i, s_{i,j}| \sum_i \Delta_i$ is more difficult to solve using traditional dynamic programming, as it requires that timing is brought into the state space. By discretizing the release times and setup times to an appropriate level of detail, a dynamic programming model can be used to find the sequence generating minimum system delay under a constrained position shift policy. This version of the CPS heuristic is now pseudo-polynomial, as the size of the problem depends on the values of the release times and the sequence-dependent setup time.

In aircraft sequencing applications, the CPS methodology is motivated by the fact that it is in many cases non-trivial for aircraft to reposition themselves if they are physically lined up in the nominal sequence. Furthermore, by limiting the number of position shifts, fairness is maintained between the entities in the system, as no entity can be moved too far from its nominal position.

5.4 Computational Study

The purpose of the computational study is to investigate the impacts of the algorithm enhancements presented in Section 5.2. More specifically we address the following questions:

- What impact does the sample size (initial sample size and sample size of additional observations) have on solution quality and runtime?
- What is a good strategy for dynamically updating the sample size? Should we be aggressive and drastically reduce the sample size for non-promising subsets, or is it better to gradually decrease the sample size for non-promising subsets? To measure the solution quality we compare the obtained solutions to the best available estimate of the optimal solution.
- What is a good termination criteria to use? How will different termination criteria affect the quality of the solution, e.g. how much optimality (with respect to the best known estimate) do we lose by terminating the algorithm after a predefined runtime or when the first full sequence is obtained?

In the remainder of this section we first report on the computational setup used to evaluate the stochastic branch and bound algorithm applied to the machine scheduling problems $1|r_i = d_i, s_{i,j}|C^{max}$ and $1|r_i = d_i, s_{i,j}| \sum_i \Delta_i$, including the computational enhancements. We begin the computational study by forming a reference case to which the changes can be compared.

5.4.1 Experiment Setup

To perform the computational analysis we implement the stochastic branch and bound algorithm described in Section 5.1 together with the models used to obtain upper and lower bounds. We also develop a schedule generator that is able to generate instances with given characteristics. We feed the desired number of jobs, the average inter-release time between

Table 35: Sequence-dependent setup times for five categories of jobs, C1,...,C5. The rows indicate the leading job and the columns indicate trailing job.

		Trailing				
		C1	C2	C3	C4	C5
Leading	C1	110	140	140	160	250
	C2	75	105	110	110	200
	C3	75	105	105	105	160
	C4	70	80	80	80	105
	C5	50	55	60	60	60

jobs, and the distribution of job categories to the schedule generator and it produces a schedule where each job is given a release time and a category.

Inspired by inter-aircraft separation requirements in aircraft sequencing, we assign a job category to each job. Sequence-dependent setup times are now defined between job categories rather than individual jobs, and we are able to specify a smaller matrix of sequence-dependent setup times. Letting h_i be the job category of job i, we have $s_{i,j} = s_{h_i,h_j}$, $\forall i, j \in \mathcal{I} \times \mathcal{I}$. The specific values can be seen in Table 35. We consider five different job categories, where we in the schedule generation process assume equal probability between categories. To simplify the discussion, we assume that the time unit is seconds.

If we assume an even distribution of job categories, the average sequence-dependent setup time is 106 seconds. In a tight sequence, with no gaps between consecutive jobs, we would be able to schedule 34.0 jobs in an hour using the average setup time. In the schedule generation process, we use three levels of job release rates, 30, 40, and 50 jobs per hour, respectively. Furthermore, we initially use nine different schedule lengths, which gives 27 different combinations of rates and schedule lengths. Using 15 instances for each combination, we get a total of 405 jobs for each experiment.

In the stochastic setting, the nominal release time (generated by the schedule generator) can be viewed as the earliest possible release time for a job. Recall that release times are considered to be stochastic in this application. For each job i we let P_i' be the distribution of delay from the nominal release time r_i^0. The distribution of release times for job i is therefore $P_i = r_i^0 + P_i'$. For the computational experiments we specify P_i' as a triangular distribution with lower limit, mode and upper limit as described in Table 36.

Table 36: Triangular delay distribution, in seconds, from earliest possible release time r^0.

Job Category	Lower limit	Mode	Upper limit
C1,C2,C3	0	60	360
C4,C5	0	60	720

Table 37: Parameters used for the reference experiment.

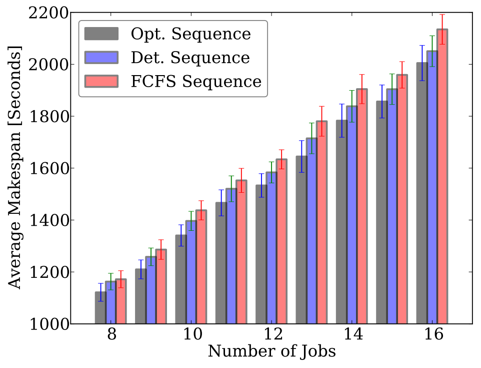
Parameter	Value
$T^{\max}$, time limit	30 minutes
q^T, iterations without improvement	50
q^E, iterations between sample evaluation	10
N^0, sample size for newly created subsets	50
N^1, sample size for existing subsets	10
$N^1_{\min}$, minimum sample size for existing subsets	1
N^-, sample size decrease	2

The algorithm is implemented in C++, and the computations are performed on a quad-core computing cluster whose head node is a 2.66 GHz Xeon X5355 processor.

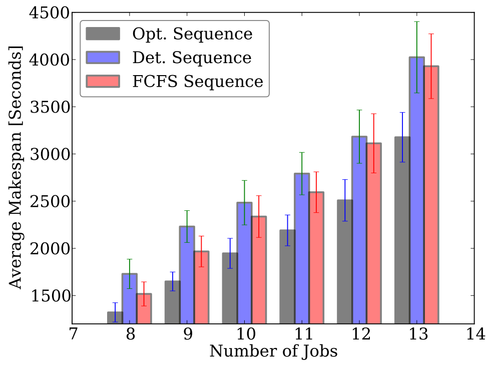
5.4.2 Reference Experiment

For comparison purposes we start by presenting the reference experiment, the experiment to which the forthcoming experiments are compared. For the reference experiment we use the algorithm parameters specified in Table 37.

The result run can be seen in Figure 16 and Figure 17. For each instance we obtain three sequences. (1) The sequence producing the minimum (estimated) expected objective using the stochastic branch and bound algorithm, (2) the optimal deterministic sequence, using r_i^0 as earliest time, and (3) the first-come, first-served sequence, also with respect to r_i^0. Each one of these sequences are sampled 1,000 times to get a good estimate of their expected objective. As there are multiple instances for each schedule length, we show the average expected objective value in Figure 16. Note that due to the additional complexity in the lower bound model, we are able to handle fewer jobs in each schedule when the system delay objective is used. In cases where the algorithm has not found a complete sequence, we use the best sequence obtained so far. If a sequence obtained from a deterministic model is used in place of the optimal sequence (obtained by the algorithm) the average makespans increase by 2% to 4%. Similarly, the FCFS sequence increases the makespans by, on average, 5% to 7% compared to the optimal sequence. Analogous values for the system delay objective

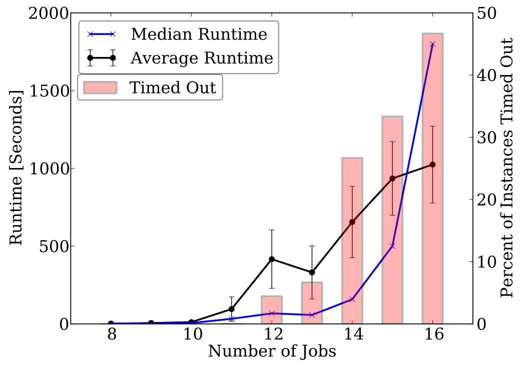


(a) Makespan objective.

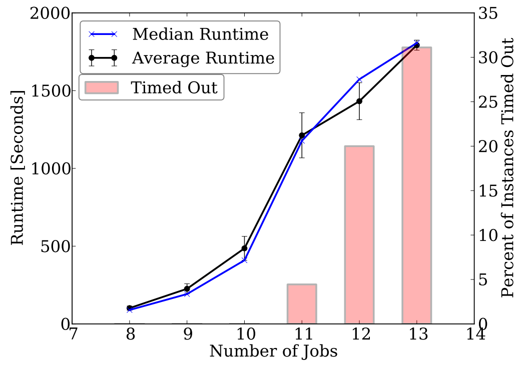


(b) System delay objective.

Figure 16: Average objective for three different scheduling methods: stochastic branch and bound algorithm (*Opt. Sequence*), optimal deterministic sequence (*Det. Sequence*), and a FCFS policy (*FCFS Sequence*). The error bars indicate the 95% confidence interval for the average objective value.



(a) Makespan objective.



(b) System delay objective.

Figure 17: Average (with 95% confidence interval) and median runtime for instances solved by the stochastic branch and bound algorithm. The bars show the percentage of instances in which we are not able to find a solution within 30 minutes of runtime.

are 27% to 31% for a deterministic sequence and 15% to 24% when the optimal sequence is replaced by a FCFS sequence.

The average and median runtimes of the stochastic branch and bound algorithm are shown in Table 17. The 95% confidence interval is included for the average runtime. We also show the percentage of instances where we are not able to find a complete sequence within the allocated run time.

The result presented in Figure 16 validates that the stochastic algorithm is able to generate sequences with significantly better objective values than deterministic models and

FCFS scheduling. Figure 17 illustrates the difference in runtime variability between the two objectives. When the makespan objective is considered, Figure 17a, there is a significant difference between the average runtime and the median runtime. This indicates that most instances with fewer than 14 jobs can be solved fast, and that a small number of instances contribute significantly to the average runtime. For the system delay objective, Figure 17b, the runtime variability is small, exemplified by the almost identical median and average runtime.

5.4.3 Sample Size Selection

We begin the computational study by finding appropriate sample sizes. In the reference experiment we use $N^0 = 50$ initial samples and $N^1 = 10$ for existing subsets. At termination of the algorithm, the optimal sequences (as generated by the algorithm) have on average been estimated using 1950 samples for the makespan objective and 830 samples for the system delay objective. This appears to be a large number of samples and we are therefore interested to see if the number of samples used to produce the bound estimates can be reduced without sacrificing optimality. Tables 38 and 39 show the average objective function value and the average runtime for different values of N^0 and N^1. The tables also show the average number of samples used to estimate the bounds in the optimal sequence, and the average number of iterations performed.

In Table 38 we can see that the values of the objective function when makespan is considered do not change with reduced sample size. This indicates that the objective is not very sensitive to what sequence is found by the algorithm. Since the separation requirements are defined by job categories, there can potentially be many sequences with the same makespan, resulting in multiple optimal solutions. By decreasing the initial sample size N^0 to 25 or 10 samples, and decreasing N^1, a slight reduction in runtime can be achieved. Note that for small N^0, the actual runtime increases. The reason for this is that the actual number of iterations increases with a small initial sample size. Furthermore, it can be seen that the number of samples in an optimal solution depends more on N^1 than on N^0.

For the system delay objective, shown in Table 39, the runtime can be significantly

Table 38: Average runtime change (39a) and average objective value change (39b) for the makespan objective for different values of N^0 and N^1. The table also shows the average number of samples used for the optimal sequence (39c) and the average number of iterations required by the algorithm (39d).

		N^1			
		10	5	3	1
N^0	50	0.0%	-1.8%	-3.2%	-0.7%
	25	3.4%	-5.6%	-1.6%	-6.9%
	10	7.7%	-0.6%	-0.6%	-0.4%
	5	12.9%	3.3%	3.2%	1.5%

(a) Change in runtime

		N^1			
		10	5	3	1
N^0	50	0.0%	0.0%	0.0%	0.1%
	25	0.0%	0.0%	0.0%	0.1%
	10	0.0%	0.0%	0.0%	0.0%
	5	0.0%	0.0%	0.0%	0.1%

(b) Change in average makespan

		N^1			
		10	5	3	1
N^0	50	1953	818	551	250
	25	1606	810	532	218
	10	1707	798	483	219
	5	1700	900	504	233

(c) Average number of samples in optimal sequence

		N^1			
		10	5	3	1
N^0	50	615	592	653	636
	25	540	696	714	689
	10	579	724	731	738
	5	564	724	808	803

(d) Average number of iterations

Table 39: Average runtime change (40a) and average objective value change (40b), for the makespan objective for different values of N^0 and N^1. The table also shows the average number of samples used for the optimal sequence (40c) and the average number of iterations required by the algorithm (40d).

		N^1			
		10	5	3	1
N^0	50	0.0%	-31.2%	-39.7%	-37.1%
	25	-2.4%	-31.6%	-39.7%	-33.3%
	10	-1.3%	-23.4%	-26.3%	-31.8%
	5	1.3%	-23.6%	-17.9%	-17.2%

(a) Change in runtime

		N^1			
		10	5	3	1
N^0	50	0.0%	-0.4%	-0.4%	-0.1%
	25	-0.3%	-0.3%	-0.2%	0.1%
	10	0.1%	-0.3%	-0.1%	0.4%
	5	0.2%	0.1%	-0.1%	0.3%

(b) Change in average system delay

		N^1			
		10	5	3	1
N^0	50	832	490	295	148
	25	800	474	315	134
	10	822	451	278	127
	5	863	475	288	124

(c) Average number of samples in optimal sequence

		N^1			
		10	5	3	1
N^0	50	129	158	157	157
	25	132	161	191	180
	10	143	177	197	215
	5	149	186	205	221

(d) Average number of iterations

Table 40: Change in runtime and average system delay for $N^0 = 10$ and different values of N^1.

	N^1				
	1/2	1/4	1/8	1/16	1/32
Runtime	-13.1%	-34.7%	-48.8%	-59.9%	-69.6%
Objective Increase	0.1%	0.2%	0.2%	0.3%	0.4%

decreased by decreasing the sample size. No clear trend can be seen for the change in the average system delay. Once again, a larger initial sample size results in fewer iterations, whereas a smaller value for N^1 results in fewer number of overall samples and reduced runtime. Tables 40a and 40b suggest that N^1 can be reduced further. In Table 40 we see that in doing so, the objective is strictly worsened.

5.4.4 Dynamic Sample Size

During the course of the algorithm, the sample sizes used for bound estimation are evaluated and updated every q^E iteration. For subsets that are not promising, i.e. they are not likely to contain an optimal solution, we successively decrease the sample size to reduce the computational burden and speed up convergence without sacrificing optimality. We select the experiment in the previous section with $N^0 = 10$ initial samples, $N^1 = 3$ samples for existing samples, and $N^1_{\text{min}} = 1$ as the reference for this analysis. Table 41 and 42 show the computational results for various levels of q^E, N^1_{min}, and sample size decrease N^-.

For the makespan objective, we first observe that had the sample size not been dynamically updated, i.e. $q^E = \infty$, the runtime would have increased by approximately 19%. In Table 42b it can be seen that a more aggressive sample size reduction does not impact the average object function value. This shows that the statistical procedure to reduce the sample size for non-promising subsets is doing what it is supposed to do. This is further supported by Table 42c, where we see that the average number of samples for an optimal sequence remains high even for an aggressive reduction scheme, in particular when $q^E = 10$. A higher sample size evaluation frequency, i.e. when $q^E = 5$, leads to a slight decrease in runtime at the expense of the number of samples used to estimate the bounds in the optimal sequence.

When considering the system delay objective, the runtime can be significantly decreased

Table 41: Impact of dynamically changing the sample size for the makespan objective. In Tables 42a and 42b the change is with respect to the base case with $N^1_{\text{min}} = 1, N^- = 2$ and $q^E = 10$.

		q^E	
N^1_{min}	N^-	10	5
1	2	0.0%	-4.0%
1/10	1	-24.3%	-22.0%
1/10	2	-25.2%	-28.8%
1/10	4	-24.6%	-28.9%
$q^E = \infty$		18.8%	

(a) Change in runtime

		q^E	
N^1_{min}	N^-	10	5
1	2	0.00%	-0.02%
1/10	1	-0.02%	-0.02%
1/10	2	0.00%	-0.03%
1/10	4	-0.01%	-0.01%
$q^E = \infty$		0.03%	

(b) Change in average makespan.

		q^E	
N^1_{min}	N^-	10	5
1	2	466	476
1/10	1	457	393
1/10	2	437	341
1/10	4	392	300
$q^E = \infty$		641	

(c) Average number of samples in the optimal sequence.

		q^E	
N^1_{min}	N^-	10	5
1	2	7.16%	6.42%
1/10	1	5.43%	6.42%
1/10	2	5.68%	5.68%
1/10	4	6.17%	5.43%
$q^E = \infty$		9.63%	

(d) Percentage of instances timed out.

Table 42: Impact of dynamically changing the sample size for the system delay objective. In Tables 42a and 42b the change is with respect to the base case with $N^1_{\text{min}} = 1, N^- = 2$ and $q^E = 10$.

		q^E	
N^1_{min}	N^-	10	5
1	2	0.0%	9.7%
1/10	1	-42.2%	-66.4%
1/10	2	-60.2%	-68.3%
1/10	4	-66.9%	-74.3%
$q^E = \infty$		55.6%	

(a) Change in runtime

		q^E	
N^1_{min}	N^-	10	5
1	2	0.00%	0.06%
1/10	1	0.07%	-0.05%
1/10	2	-0.04%	0.24%
1/10	4	-0.04%	0.19%
$q^E = \infty$		0.39%	

(b) Change in average system delay

		q^E	
N^1_{min}	N^-	10	5
1	2	265	266
1/10	1	295	269
1/10	2	277	281
1/10	4	285	229
$q^E = \infty$		248	

(c) Average number of samples in optimal sequence

		q^E	
N^1_{min}	N^-	10	5
1	2	2.96%	1.85%
1/10	1	1.48%	0.74%
1/10	2	0.74%	0.00%
1/10	4	0.74%	0.37%
$q^E = \infty$		5.56%	

(d) Percentage of instances timed out

Table 43: Change in average objective function value, compared to $T^{\max} = 30$, and the percentage of instances in which no complete sequence was found for different time limits.

Runtime limit, $T^{\max}$ (minutes)	Change in average objective	Percentage of instances timed out
5	0.08%	11.11%
10	0.03%	9.38%
15	0.00%	8.64%
20	0.00%	8.89%
30	0.00%	7.16%

(a) Makespan objective

Runtime limit, $T^{\max}$ (minutes)	Change in average objective	Percentage of instances timed out
5	1.49%	18.15%
10	0.46%	5.56%
15	0.35%	5.19%
20	0.02%	2.59%
30	0.00%	2.96%

(b) System delay objective

without increasing the average system delay. However, changing the sample size too aggressively leads to an increase in average system delay when $q^E = 5$. By dynamically changing the sample size, the percentage of instances in which no complete sequence is found can be reduced from 3% in the base case to less than 1%.

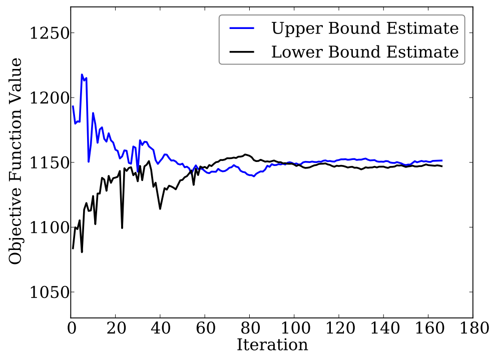
From the computational results presented above we conclude that by dynamically changing the sample size we are able to reduce the runtime by 40% and 80% (compared to the case where $q^E = \infty$) for the makespan objective and system delay objective without increasing the value of the objective function. We also conclude that evaluating the sample sizes too aggressively, e.g. $q^E = 5$ and $N^- = 4$, can lead to increased objective function values.

5.4.5 Termination Criteria

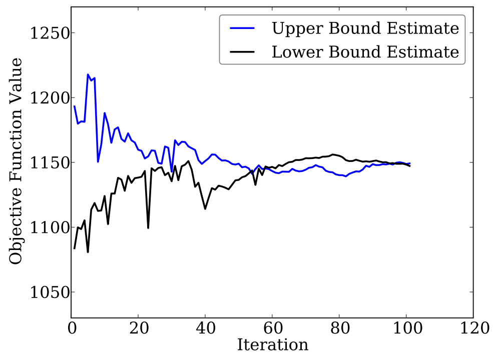
In the final section of the computational study we investigate various termination criteria. For the reference case we use the parameters specified in Table 37, with the exception of N^0 and N^1, where we use $N^0 = 10$ and $N^1 = 3$.

Table 43 shows the change in objective function value and the percentage of instances in which no complete sequence was found, for different runtime limits. Using the makespan objective we once again see that there are multiple optimal sequences. The algorithm fails to find sequences with makespan close to the optimal makespan only when a small time limit is used. For the system delay objective, we can see that a time limit of 20 minutes gives close to identical objective values as 30 minutes.

So far, the algorithm has terminated when the time limit is reached or if we have not seen any improvement for $q^T = 50$ iterations. Figure 18 shows the upper and lower bound for



(a) $q^T = 50$



(b) $q^T = 20$

Figure 18: Upper and lower bound estimates for an instance with 9 jobs.

Table 44: Change in average runtime, average objective function value and the percentage of instances in which no complete sequence was found for two different termination criteria.

Termination Criteria	Change in average runtime	Change in average objective	Percentage of instances timed out
Base	0.0%	0.0%	7.2%
q^T= 20	-10.5%	0.0%	6.9%
Stop at first sequence	-45.5%	0.3%	7.7%

(a) Makespan objective

Termination Criteria	Change in average runtime	Change in average objective	Percentage of instances timed out
Base	0.0%	0.0%	3.0%
q^T= 20	-34.3%	0.3%	1.9%
Stop at first sequence	-68.2%	2.3%	2.6%

(b) System delay objective

an instance with 9 jobs. This example suggest that q^T can be decreased without increasing the average objective function value. In Table 44, the impact of decreasing q^T to 20 is evaluated. We also show the impact of terminating the algorithm after the first complete sequence is found, i.e. the algorithm stops as soon as we find the first solution.

From Table 44, we conclude that q^T can be decreased for a runtime reduction of 10% and 34%, respectively. However, terminating the algorithm when the first sequence is found will lead to an increase in average objective function value.

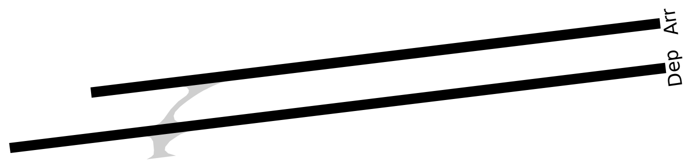


Figure 19: The runway configuration in this study consist of a pair of closely spaced parallel runways with one crossing where arrivals taxiing to the gate cross the departure runway.

5.5 Implementation for Runway Scheduling

In the next two sections we apply the methodology previously developed to an instance of airport runway scheduling. More specifically, we consider an existing runway configuration and include the necessary level of detail to accurately model real world operations. Due to the added complexity from the operational constraints, we propose a different method to solve the problem $\min_{\mathbf{x}\in\mathcal{X}^s} f(\mathbf{x},\omega)$ and to calculate the objective of a fixed sequence.

For a more natural description of the problem we switch the terminology from machine scheduling to airport runway scheduling. Thus, the set of jobs $\mathcal{I}$ is replaced by the set of aircraft, $\mathcal{A}$. The release times $r_a^\omega,\ a \in \mathcal{A}$ refer to the earliest time an aircraft is available for scheduling at the runway and the start time $t_a,\ a \in \mathcal{A}$ are the times by which the aircraft use the runway. Also, the aircraft separation requirements take the role of sequence-dependent setup times.

Closely spaced parallel runways are defined as a set of runways whose centerlines are separated by less than 4,300 ft. Generally, one runway is dedicated for departures and the other runway is used by arriving aircraft. In this study, we consider a runway layout depicted in Figure 19, where arriving aircraft bound for the airport terminal need to cross the departure runway. In addition, we assume that the taxiway between the runways can only accommodate one aircraft at a time. This configuration can be found for eastbound operations at Los Angeles International Airport (LAX) using runways 24L and 24R.

5.5.1 Runway Scheduling Model

Observing that advanced sequencing models are most beneficial when traffic loads are high (Brentnall and Cheng 2009; Solveling, Solak, Clarke, and Johnson 2011a), we design our solution methodology for peak periods with high arrival and departure rates. Therefore, we consider runway utilization as our primary objective and individual aircraft delays as a secondary objective. It is important to note that increased runway throughput in cases where the demand for runway resources is higher than the runway capacity leads to less overall delay. However, fairness cannot be guaranteed because a subset of aircraft may experience increased delays.

In addition to the times by which aircraft use the runway, we need to include decisions for crossing. We denote this by ct_a, $a \in Arr$, where Arr is the set of all arriving aircraft, or in vector form as $\mathbf{ct}$.

Thus, in this instance for airport runway scheduling we define the value of $f(\mathbf{x}, \omega)$ in Equation (71).

$$f(\mathbf{x}, \omega) = \min\{\max_{a \in \mathcal{A}} t_a, \max_{a \in Arr} ct_a\} \tag{71}$$

$$\begin{aligned}
\text{s.t.} \quad & t_j \geq t_i + s_{i,j} && \text{if } i \in \mathcal{A} \text{ is sequenced before } j \in \mathcal{A} \text{ in } \mathbf{x} \\
& && \forall i, j \in \mathcal{A} \times \mathcal{A},\ i \neq j && (72) \\
& ct_a \geq t_a + tt_a && \forall a \in Arr && (73) \\
& ct_i \leq t_j + \beta_j tt_j && \text{if } i \in Arr \text{ is sequenced before } j \in Arr \text{ in } \mathbf{x} && (74) \\
& ct_j \geq t_i + s^{dc}_{i,j} && \text{if } i \in Dep \text{ is sequenced before } j \in Arr && (75) \\
& t_j \geq ct_i + s^{cd}_{i,j} && \text{if } i \in Arr \text{ is sequenced before } j \in Dep && (76) \\
& t_a \geq r^{\omega}_a && \forall a \in \mathcal{A} && (77) \\
& ct_a \geq 0 && \forall a \in Arr && (78)
\end{aligned}$$

The objective function, given in Equation (71), aims to minimize the time of the last operation. Separation requirements between all operations on the runway are defined in Constraint (72). The parameter tt_a denotes the minimum taxi time between touchdown and the point where the departure runway is crossed. Thus, Constraint (73) enforces the minimum time between runway arrivals and crossing of the departure runway. The assumption that only one aircraft can be held between the runways is modeled in Constraint (74). The parameter β_a represents the fraction of taxi time aircraft $a \in Arr$ spend on the runway.

If aircraft $i \in Arr$ is scheduled before aircraft $j \in Arr$, then i needs to begin crossing before aircraft j exits the runway at time $t_j + \beta_j tt_j$. Note that Constraint (74) implies separation between consecutive crossings as $ct_j \geq ct_i + (1 - \beta_j)tt_j$.

Constraint (75) enforce separation between departures followed by crossings, where $s^{dc}_{i,j}$ is the minimum separation between departing aircraft $i \in Dep$ and crossing aircraft $j \in Arr$. Similarly, Constraint (76) enforces separation between crossings followed by departures, $s^{cd}_{i,j}$ is the minimum separation between crossing aircraft $i \in Arr$ and departing aircraft $j \in Dep$. In this application, we assume that separation requirements involving crossing operations are independent of aircraft weight class, and set $s^{dc}_{i,j} = s^{dc}$, $\forall i, j \in \mathcal{A} \times \mathcal{A}$ and $s^{cd}_{i,j} = s^{cd}$, $\forall i, j \in \mathcal{A} \times \mathcal{A}$.

Note that the sequence of operations, $\mathbf{x}$, only define arrival and departure operations on the runway and that crossings are not included in $\mathbf{x}$. Therefore, decisions on where crossings shall be inserted in the departure stream can be seen as a recourse decision and we can, in this setting, think of the model as a two-stage stochastic program.

5.5.2 Solution Procedure for Runway Scheduling Model

Given a sequence of operations, $\mathbf{x}$, defining the usage of the arrival and departure runways and the random outcome ω, we present a method to evaluate the value of $f(\mathbf{x}, \omega)$. In the method we take advantage of the assumption that only one aircraft can be held between the runways. In addition to finding the time of arrival and departure operations $\mathbf{t}$, we also need to find where in the departure stream crossings need to be inserted to minimize the latest time of operation.

We begin by defining a procedure to calculate runway operations for the aircraft in $\mathcal{A}$ when runway crossings are *not* considered.

Procedure 2. *Given a sequence of aircraft $\mathbf{x} = [x(1), \ldots, x(\ell)]$ with attributes r^{ω}_{a} for each aircraft, the following procedures set the runway times. If present, $x^{Arr}0$ and x^{Dep}_0 are initial aircraft scheduled at time t^{Arr}_0 and t^{Dep}_0, respectively.*

```
Set $x^{Arr}_0, t^{Arr}_0, x^{Dep}_0, t^{Dep}_0$ if present
for $i = 1 \to \ell$ do
```

```
    $sepA = t_0^{Arr} + s_{x_0^{Arr},x(i)}$;    $sepD = t_0^{Dep} + s_{x_0^{Dep},x(i)}$
    $t_{x(i)} = \max(r^{\omega}_{x(i)}, sepA, sepD)$
    if $x(i) \in Arr$ then
      $x_0^{Arr} = x_i$, $t_0^{Arr} = t_{x(i)}$
    else
      $x_0^{Dep} = x_i$, $t_0^{Dep} = t_{x(i)}$
    end if
  end for
```

Because we can process the crossings sequentially, we only need to consider sequences of the form $a(1), d(1), \dots, d(n), a(2)$ where $a(1)$ is the arrival aircraft whose crossing time is to be determined, $a(2)$ is the subsequent arrival and the sequence of intermediate departures (which can be empty) is represented by $d(1), \dots, d(n)$.

Given the schedule prior to arrival aircraft $a(1)$ and the time for this aircraft, $t_{a(1)}$, the departures in the sequence together with the subsequent arrival $a(2)$ (if such exists) are scheduled according to Procedure 2. Let $\tilde{t}_{d(i)}$, $i = 1, \dots, n$ and $\tilde{t}_{a(2)}$ be the scheduled times resulting from Procedure 2.

We now need to find the best position in the departure sequence where we can insert the crossing of $a(1)$. Let the schedule slack generated by aircraft $d(i)$ be defined as $\tilde{s}_{d(i)} = \tilde{t}_{d(i)} - \tilde{t}_{d(i-1)} - s_{d(i-1),d(i)}, i = 2, \dots, n$, $\tilde{s}_{d(1)} = 0$, similarly $\tilde{s}_{a(2)} = \tilde{t}_{a(2)} - \tilde{t}_{d(n)} - s_{d(n),a(2)}$. As separation constraints are enforced in the calculation of $\tilde{\mathbf{t}}$ we have that $\tilde{s}_{d(i)} \geq 0$. Using the slack, we can calculate how much delay the remaining schedule can absorb without delaying $a(2)$ (or $d(n)$, if no subsequent arrival exist) to a time later than $\tilde{t}_{a(2)}$ ($\tilde{t}_{d(n)}$). We define this maximum schedule absorption for departure $d(j)$ as $\tilde{p}_{d(j)} = \sum_{k=j+1}^{n} \tilde{s}_{d(j)}$, $j = 1, \dots, n-1, \tilde{p}_{d(n)} = s_{a(2)}$ ($\tilde{p}_{d(n)} = 0$ if $a(2)$ is not present).

Let $\tilde{ct}_{a(1),i}$ be the time that aircraft $a(1)$ crosses the departure runway *if* it is scheduled to cross immediately before departure $d(i)$. Now, $\tilde{ct}_{a(1),i}$ is defined as:

$$\tilde{ct}_{a(1),i} = \begin{cases} \max(t_{a(1)} + tt_{a(1)}, \tilde{t}_{d(i-1)} + s^{dc}) & \text{if } i = 2, \dots, n \\ \max(t_{a(1)} + tt_{a(1)}, t_{d(0)} + s^{dc}) & \text{if } i = 1 \end{cases} \tag{79}$$

where $t_{d(0)}$ is the time of the last departure prior to $a(1)$.

We continue by letting $\hat{t}_{d(i)}$ be the departure time of aircraft $d(i)$ if aircraft $a(1)$ is scheduled to cross immediately before departure $d(i)$. $\hat{t}_{d(i)}$ is defined as $\hat{t}_{d(i)} = \max(\tilde{t}_{d(i)}, \tilde{ct}_{a(1),i} + s^{cd})$. For each departure $d(i), i = 1, \ldots, n$ we let $\tilde{\delta}_{d(i)}$ be the delay from $\tilde{t}_{d(i)}$ if $a(1)$ is scheduled to cross immediately before $d(i)$, $\tilde{\delta}_{d(i)} = \hat{t}_{d(i)} - \tilde{t}_{d(i)}$.

Starting with $d(1)$, we can sequentially try to schedule the crossing before departure $d(i)$. If $\tilde{\delta}_{d(i)} \leq 0$, we can let the arrival cross at time $\tilde{ct}_{a(1),i}$ without delaying the next departure. If $\tilde{\delta}_{d(i)} \leq \tilde{p}_i$ we can schedule the crossing at time $\tilde{ct}_{a(1),i}$ without delaying $a(2)$ ($d(n)$ if $a(2)$ is not present) and worsen the objective. If no departure can be found where any of the previous statements hold, we choose to schedule the crossing arrival before departure $j = \arg\min_{i=1,\ldots,n}(\tilde{\delta}_{d(i)} - \tilde{p}_{d(i)})$, with a total delay of $\tilde{\delta}_{d(j)} - \tilde{p}_{d(j)}$ units for the last departure. We also check if it is beneficial to schedule the crossing after the departure of aircraft $d(n)$.

Once we have decided where to insert the crossing, we need to update the schedule accordingly. If we schedule the crossing prior to departure $d(j)$, we get the crossing time $ct_{a_1} = \tilde{ct}_{a(1),j}$. We set $t_{d(i)} = \tilde{t}_{d(i)}, i = 1, \ldots, j-1$, we set $t_{d(j)} = \hat{t}_{d(j)}$, and we schedule the remaining aircraft, including a_2 if it exists, according to Procedure (2). If $t_{a(2)}$ violate (74), we set $t_{a(2)} = ct_{a(1)} - \beta_{a(2)} tt_{a(2)}$

5.5.3 Lower Bound Estimation

To estimate the lower bound we need to repeatedly solve the minimization problem defined in Equation (70). In Section 5.3, we use a dynamic programming heuristic to approximately solve the problem for the portion of the sequence that is not fixed. Due to the additional complexities introduced by runway crossings, we are not able to use the dynamic programming algorithm, and we, therefore, propose a different solution method.

The heuristic we propose is a hybrid between partial enumeration and the CPS heuristic used in Section 5.3. Rather than finding a solution to Equation (70) directly, we implicitly generate a subset of aircraft sequences and look for the best sequence among these. The process of finding a subset of aircraft sequences is a two-step process where we first generate a subset set of weight class sequences and then extract the aircraft sequences. Using the two-step process we avoid generating aircraft sequences that are dominated by other aircraft sequences and thus give a worse objective.

Given a branching node with the corresponding subset $\mathcal{X}^s = \mathcal{X}_{x(1),\ldots,x(k)}$, we perform the lower bound estimation according to Procedure 3.

Procedure 3. *Procedure to estimate lower bound in iteration q for subset $\mathcal{X}^s = \mathcal{X}_{x(1),\ldots,x(k)}$.*

```
Generate a set of aircraft weight classes, $\mathcal{W}'_{w(1),\ldots,w(k)}$.        (Section 5.5.3.1)
Let $N = l(q)\,[\mathcal{X}^s]$
for $i = 1 \to N$ do
  for $\mathbf{w} \in \mathcal{W}'_{w(1),\ldots,w(k)}$ do
    Generate an aircraft sequence $\mathbf{x}$ from $\mathbf{w}$        (Section 5.5.3.2)
    Evaluate $f(\mathbf{x}, \omega^i)$        (Section 5.5.2 )
  end for
  Store the best objective as $\min_{\mathbf{x} \in X^s} f(\mathbf{x}, \omega^i)$
end for
Update the lower bound $\xi^{l(q)}(\mathcal{X}^s)$ (according to Equation (67))
```

In Procedure 3 above, $W'_{w(1),\ldots,w(k)} \subseteq W_{w(1),\ldots,w(k)}$, where $W_{w(1),\ldots,w(k)}$ denote the set of all possible aircraft weight class sequences when positions 1 through k have fixed weight classes $w(1), \ldots, w(k)$.

The motivation behind the use of aircraft weight class sequences in the algorithm stems from the fact that separation requirements are defined for aircraft weight classes and not individual aircraft. Thus, we can limit our search for good aircraft sequences to good weight class sequences. Defining K as the set of aircraft weight classes present in $\mathcal{A}$ and $\mathcal{W}$ as the set of all possible weight class sequences corresponding to the aircraft of $\mathcal{A}$, we see that $|\mathcal{X}| = n!$ whereas $|\mathcal{W}| = n!/\Pi_{k \in K}(n_k!)$ where the number of aircraft of weight class $k \in K$ is given by n_k. Clearly, unless $n_k = 1$, $\forall k \in K$, $|\mathcal{W}| << |\mathcal{X}|$.

5.5.3.1 Generating Weight Class Sequences

Because we are using an enumerative approach to solve formulation (70), it is important that we select a good set of weight class sequences $\mathcal{W}'_{w(1),\ldots,w(k)}$ to evaluate. Of course, if $|\mathcal{W}_{w(1),\ldots,w(k)}|$ is small, we can use $\mathcal{W}'_{w(1),\ldots,w(k)} = \mathcal{W}_{w(1),\ldots,w(k)}$, otherwise we need to generate the set $\mathcal{W}'_{w(1),\ldots,w(k)}$. Once again (see Section 5.3) we make use of the constrained position shift idea, although now we use the CPS heuristic in a different manner. Instead of using the CPS network to solve the dynamic program, we enumerate all possible weight class sequences given parameter τ to form the set $\mathcal{W}'_{w(1),\ldots,w(k)}$. If implemented carefully, the creation and enumeration of CPS networks is fast relative to the time spent evaluating samples.

5.5.3.2 Obtaining Aircraft Sequence from Weight Class Sequence

Although we potentially could generate a set of aircraft sequences directly, many of these sequences would be dominated by other sequences. For each new sample we would either have to evaluate a large number of aircraft sequences or spend time eliminating dominated sequences. Instead, we create a pool of weight class sequences, $\mathcal{W}'_{w(1),\ldots,w(k)}$, and use these to generate non-dominated aircraft sequences after the runway times have been realized.

Given a sample ω with the associate runway times $\mathbf{r}^\omega$ and a fixed weight class sequence $w' = [k(1), \ldots, k(\kappa), \ldots, k(n)]$, $w' \in \mathcal{W}'_{w(1),\ldots,w(\kappa)}$ we use the following steps to extract a non-dominated aircraft sequence. We begin by ordering the aircraft in non-decreasing order of $\mathbf{r}^\omega$. Starting from the first position, we put the first available aircraft of weight class $k_{(i)}$ in position i. The procedure is formally defined as Procedure 4

Procedure 4. *Let* $\mathbf{w} = \left[k_{(1)}, \ldots, k_{(n)}\right]$ *be a weight class sequence and let* $\mathcal{A}$ *be a set of aircraft such that* $|\mathcal{A}| = n$. *Each aircraft* a *has an earliest time* r^ω_a. *The procedure defined below builds an aircraft sequence* $\mathbf{x}$ *from the weight class sequence* $\mathbf{w}$.

```
$\mathbf{x} = ()$
$\mathcal{A}(k) = \{a \in \mathcal{A} : h_a = k\}$
for $i = 1 \to n$ do
    $\bar{x}_{(i)} = \{a \in \mathcal{A}(k_{(i)}) : r^\omega_a \le r^\omega_{a'},\ a' \in \mathcal{A}(k_{(i)})\}$
    if $|\bar{x}_{(i)}| > 1$ then
        Break ties arbitrarily to obtain a single aircraft $\bar{x}_{(i)}$
    end if
    $\mathbf{x} = \mathbf{x} + \bar{x}_{(i)}$ *(Appended to the end)*
    $\mathcal{A}(k_{(i)}) = \mathcal{A}(k_{(i)}) \setminus \bar{x}_{(i)}$
  **end for**
```

In the procedure we use h_a to denote the weight class of aircraft a.

In Proposition 8, we show that Procedure 4 gives the optimal aircraft sequence with respect to delays for individual aircraft, which is our secondary objective. With this result we have an efficient way to generate aircraft sequences from weight class sequences that are optimal with respect to delay for individual aircraft. To show the result we make use of the following Lemma:

Lemma 1. *Let f be a convex function and let $\lambda_1 < \lambda_2 \leq \lambda_3 < \lambda_4$. Furthermore, let $\lambda_1 + \lambda_4 = \lambda_2 + \lambda_3$. Then $f(\lambda_1) + f(\lambda_4) \geq f(\lambda_2) + f(\lambda_3)$.*

Proof. To prove the result we show that $f(\lambda_4) - f(\lambda_3) \geq f(\lambda_2) - f(\lambda_1)$, or equivalently, $\frac{f(\lambda_4)-f(\lambda_3)}{\lambda_4-\lambda_3} \geq \frac{f(\lambda_2)-f(\lambda_1)}{\lambda_2-\lambda_1}$.

Expressing λ_2 as a convex combination of λ_1 and λ_4 we have $\lambda_2 = \frac{\lambda_4-\lambda_2}{\lambda_4-\lambda_1}\lambda_1 + \frac{\lambda_2-\lambda_1}{\lambda_4-\lambda_1}\lambda_4$. By convexity of f, $f(\lambda_2) \leq \frac{\lambda_4-\lambda_2}{\lambda_4-\lambda_1} f(\lambda_1) + \frac{\lambda_2-\lambda_1}{\lambda_4-\lambda_1} f(\lambda_4)$. After rearranging we get

$$\frac{f(\lambda_4) - f(\lambda_1)}{\lambda_4 - \lambda_1} \geq \frac{f(\lambda_2) - f(\lambda_1)}{\lambda_2 - \lambda_1} \tag{80}$$

Similarly, by expressing λ_3 as a convex combination of λ_1 and λ_4, using convexity of f and rearranging we get

$$\frac{f(\lambda_4) - f(\lambda_3)}{\lambda_4 - \lambda_3} \geq \frac{f(\lambda_4) - f(\lambda_1)}{\lambda_4 - \lambda_1} \tag{81}$$

The result is obtained by combining (80) and (81). □

Proposition 8. *Assume that all aircraft have a convex function describing the cost of delay. Furthermore, assume that all aircraft within the same weight class have the same function describing the delay. Then, given a weight class sequence $\mathbf{w} \in \mathcal{W}$ and realization $\mathbf{r}^{\omega}$, Procedure 4 gives an aircraft sequence with minimum cost solution with respect to the delay cost function for each aircraft.*

Proof. Assume that this is not the case. Then w.l.o.g. there exist an aircraft sequence $\mathbf{x}$ such that aircraft b is scheduled before aircraft a but $r_a \leq r_b$, which has a cost that is strictly smaller than the sequence obtained in Procedure 4. (For clarity we drop the superscript ω.) Let t_a and t_b be the scheduled times in sequence $\mathbf{x}$ for aircraft a and b, respectively, where $t_b < t_a$. Define the delay from the earliest time as

$$\Delta_a = t_a - r_a$$

$$\Delta_b = t_b - r_b$$

Consider a modified schedule $\mathbf{x}'$ obtained from Procedure 4, where the order of operations for aircraft a and b are interchanged, but everything else remains the same. Let t'_i be the scheduled time for aircraft i, $i = a, b$ in schedule $\mathbf{x}'$, all other aircraft keep their scheduled time from sequence $\mathbf{x}$. We have $t'_a = t_b$ and $t'_b = t_a$. The delay is defined as

$$\Delta'_a = t'_a - r_a = t_b - r_a$$

$$\Delta'_b = t'_b - r_b = t_a - r_b$$

The sum of delays in both sequences are equal, i.e. $\Delta_a + \Delta_b = \Delta'_a + \Delta'_b$. As $t_a > t_b$, we have $\Delta_a > \Delta'_a$. Similarly, $\Delta'_b > \Delta_b$. Also, as $r_b \geq r_a$, we have $\Delta_a \geq \Delta'_b$ and $\Delta'_a \geq \Delta_b$. Applying Lemma 1 with $\Delta_a = \lambda_4$ and $\Delta_b = \lambda_1$ we get $f(\Delta_a) + f(\Delta_b) \geq f(\Delta'_a) + f(\Delta'_b)$. This contradicts the fact that the cost of $\mathbf{x}$ is strictly less than the cost of $\mathbf{x}'$. $\square$

5.5.4 Upper Bound Estimation

In order to get a complete aircraft sequence $\mathbf{x}^*$ to use in Equation (66), we extend the given sequence $\mathbf{x} = [x(1), \ldots, x(\kappa)]$ by ordering the remaining aircraft by their expected time at the runway, $\mathbb{E}(\mathbf{R}_i)$, $\forall i \in \mathcal{I} \setminus \{x(1), \ldots, x(k)\}$, such that $E(\mathbf{R}_{x(k+1)}) \leq \cdots \leq E(\mathbf{R}_{x(n)})$. With this complete sequence, we estimate the upper bound of subset X^s by repeatedly solving $f(\mathbf{x}^*, \omega)$ for different realizations ω. We evaluate $f(\mathbf{x}^*, \omega)$ using the heuristic approach described in Section 5.5.2.

5.6 Computational Study for Runway Scheduling

The primary objective of the computational study for the runway scheduling implementation is twofold. First, we investigate the quality of the solutions generated by the stochastic

Table 45: Aircraft weight class distributions used for schedule generation.

Aircraft Weight Class	Probability
Heavy	0.4
Large	0.3
Small	0.3

branch and bound algorithm applied to runway scheduling and compare it to solutions obtained from deterministic models. By sampling the aircraft sequences obtained from the various models we can compare the quality of the solutions. When sampling the complete aircraft sequences we use the same procedure as when we estimate the upper bound. Second, we record the runtime requirements for our stochastic branching algorithm. We also incorporate promising strategies and termination criteria from the analysis in Section 5.4.

In addition to the analysis introduced above, we investigate the sensitivity in the stochastic distributions describing the delay from the earliest possible runway time. By changing the parameters for the distribution, we can evaluate the performance and solution quality for different levels of uncertainty.

5.6.1 Input Data and Parameters

For the computational experiments we reuse the earlier developed schedule generator to generate flight schedules. We generate an equal number of arrivals and departures and use the aircraft weight class distribution specified in Table 45.

In each experiment we generate schedules with 8 to 16 flights and arrival and departure rates varying between 30 and 50 flights per hour and runway. In total, 405 random schedules are generated for each experiment.

The distribution of runway times for aircraft a is, similar to the previous study, defined as $P_a = r_a^0 + P'_a$, where r_a^0 is the earliest possible runway time for aircraft a. For P'_a we use a triangular distribution with parameters shown in Table 46. In Section 5.6.2.3, we evaluate the sensitivity of these parameters. In this study, the delay distributions P'_a, $\forall a \in \mathcal{A}$ are independent from each other. This is a valid assumption if the airport configuration allow independent taxiway operations for departures, or arriving aircraft are using different arrival streams. At airports where operations are not independent, distributions capturing

Table 46: Triangular delay distribution, in minutes, from earliest possible release time r^0.

Direction	Lower Limit	Mode	Upper Limit
Arrival	0	1	6
Departure	0	1	12

Table 47: Parameters used in the algorithm applied to runway scheduling.

Parameter	Value
$T^{\max}$, Time limit	20 minutes
q^T, iterations without improvement	50
q^E, iterations between sample evaluation	5
N^0, sample size for newly created subsets	50
N^1, sample size for existing subsets	5
$N^1_{\min}$, minimum sample size for existing subsets	1
N^-, sample size decrease	2
tt, travel time from touchdown to runway crossing	45 seconds
S^{dc}, separation between departure followed by a crossing	60 seconds
S^{cd}, separation between crossing followed by a departure	30 seconds
β, fraction of taxi time spent on runway	0.5

the dependency need to be used. Note that the algorithm only requires that samples can be drawn from the distributions, there is no requirement that realizations among different aircraft are independent.

There are a number of parameters that define the behavior of the stochastic branch and bound algorithm. These parameters are presented in Table 47. In addition, we use the separation requirements specified in Tables 12, 13, and 14 for operations on close parallel runways.

5.6.2 Configuration 1: General Layout

The implementation details in Section 5.5 and the parameters presented in this section are given for a runway configuration with two close parallel runways using a single crossing for arrivals to cross the departure runway. We begin our analysis by comparing the sequence obtained from the stochastic branch and bound algorithm to aircraft sequences obtained from a FCFS policy and a deterministic scheduling algorithm.

5.6.2.1 Solution Quality

To assess the quality of the solution obtained from the stochastic branch and bound procedure, we extract the best sequence generated by the algorithm and estimate the makespan

Table 48: Comparison between aircraft sequences obtained from the stochastic branch and bound algorithm (*Opt.*), the deterministic scheduling model (*Det.*) and the FCFS scheduling policy (*FCFS*).

Number of	Avg. Makespan	Avg. Makespan Change		% H_0 rejected	
Flights	*Opt.*	*Opt.* to *Det.*	*Opt.* to *FCFS*	*Det.*	*FCFS*
8	14.72	7.80%	13.81%	0.00%	0.00%
9	15.97	6.13%	13.04%	4.44%	0.00%
10	16.99	6.43%	14.55%	6.67%	0.00%
11	18.21	7.17%	12.80%	0.00%	0.00%
12	19.55	6.27%	15.94%	8.89%	0.00%
13	20.44	6.65%	16.29%	2.22%	0.00%
14	21.65	5.83%	16.08%	8.89%	0.00%
15	23.12	5.16%	16.18%	4.44%	0.00%
16	24.14	5.20%	16.63%	11.11%	0.00%

H_0 : *Opt.* makespan < *Det./FCFS* makespan

of the sequence through sampling. Note that makespan estimation is a part of the algorithm, but since we have no way of controlling the number of samples used for the optimal sequence we simply re-estimate the makespan for the optimal sequence using 1,000 samples. For the estimation, we use Equation (66) with x^* set to the optimal sequence. Similarly, we find an optimal deterministic sequence and a FCFS sequence, where we in both cases use r_a^0 as the earliest time an aircraft can be scheduled, and estimate the makespan of these two sequences. In Table 48, we can see that the sequence obtained from a deterministic model increase the average makespans by 5% to 8% compared to the optimal sequence from the algorithm. Similarly, the FCFS sequence increases the makespans by, on average, 13% to 16% compared to the optimal sequence obtained in the stochastic branch and bound algorithm. The result is visualized in Figure 20.

In Table 48, we record the percentage of instances in which the null hypothesis H_0 : *Estimated Makespan of Opt. Sequence* < *Estimated Makespan of Ref. Sequence* is rejected. In the table, we use the deterministic and FCFS sequence as reference sequences. There is only a small number of instances in which the null hypothesis is rejected.

5.6.2.2 *Runtime Requirements*

The efficiency of the algorithm is not only defined by the quality of the solution, but also by the computational time required. For practical applications it is important to have a relatively short runtime, so that the recommended aircraft sequence can be implemented

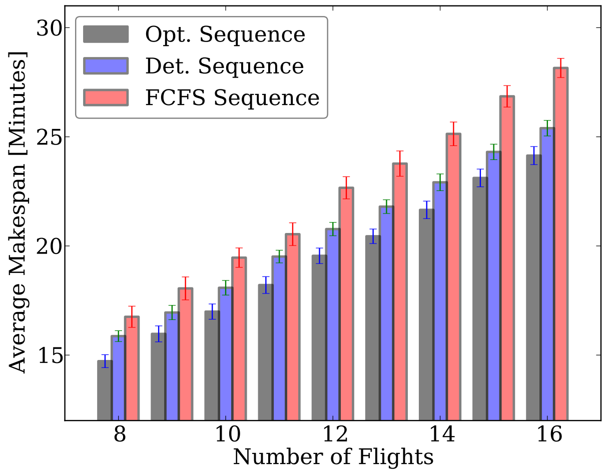


Figure 20: Average makespan for three different scheduling methods: stochastic branch and bound algorithm (*Opt. Sequence*), optimal deterministic sequence (*Det. Sequence*), and a FCFS policy (*FCFS Sequence*). The error bars indicate the 95% confidence interval for the average makespan.

before the scheduled activity takes place. In Figure 21, we show the average and median runtimes for various number of flights. We also indicate the 95% confidence interval for the average. In Figure 21, we also show the percentage of instances where we are not able to find a feasible solution within 20 minutes of available runtime.

Note that the runtimes for the runway application in this section is a magnitude smaller than the runtimes reported in Section 5.4. This is due to the different heuristics that are used. In Section 5.4, a dynamic program is solved for every sample, whereas we in this application generate a pool of sequences for each node and then enumerate over this pool to estimate (heuristic) lower bounds. Despite the fact heuristic lower bounds are used, Figure 20 and Table 48 indicate that we are obtaining high quality solutions.

5.6.2.3 Sensitivity in Stochastic Input

In the analysis up to this point we have used the probabilistic distributions defined in Table 46. To evaluate the impact of the probability distribution describing the delay, we run the base algorithm for three different triangular distributions and for the completely deterministic case where $P' = 0$ *w.p.* 1. In the triangular distributions, we keep the lower limit and mode fixed at 0 and 1, respectively. We alter the shape of the distribution by

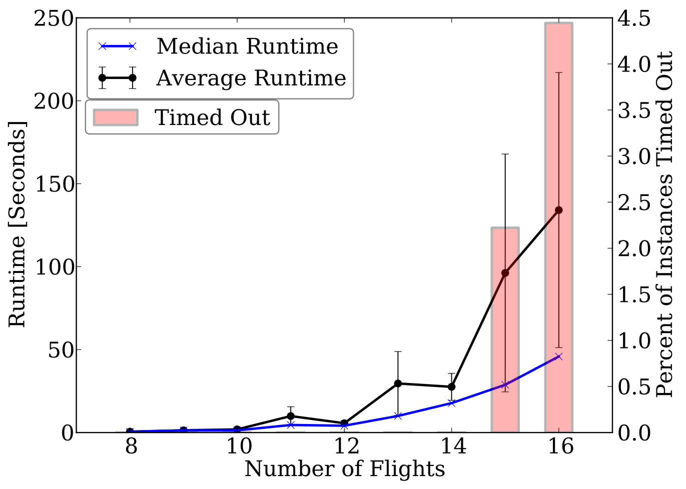
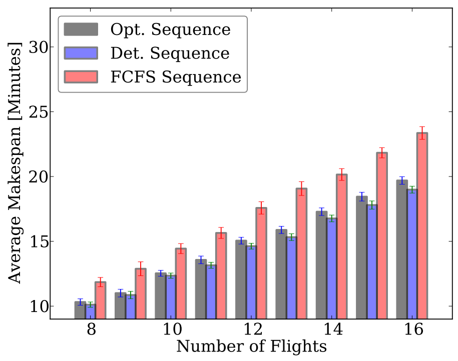


Figure 21: Average (with 95% confidence interval) and median runtime for instances solved by the stochastic branch and bound algorithm. The bars show the percentage of instances in which we were not able to find a solution within 20 minutes of runtime.

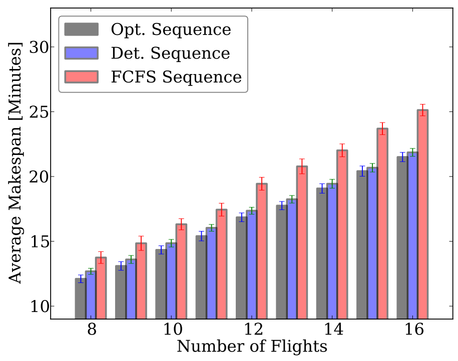
Table 49: Change in average makespan when using a deterministic model compared to using the stochastic branch and bound algorithm for four different input distributions. *Distr. 1* has no uncertainty whereas *Distr. 4* has the highest level of uncertainty.

Number of Flights	*Distr. 1*	*Distr. 2*	*Distr. 3*	*Distr. 4*
8	-1.97%	4.79%	7.80%	8.35%
9	-1.36%	3.84%	6.13%	6.33%
10	-1.45%	3.60%	6.43%	7.52%
11	-3.07%	4.14%	7.17%	8.03%
12	-2.84%	3.01%	6.27%	7.79%
13	-3.51%	2.76%	6.65%	8.12%
14	-3.01%	1.91%	5.83%	7.63%
15	-3.51%	1.28%	5.16%	7.27%
16	-3.59%	1.67%	5.20%	7.58%

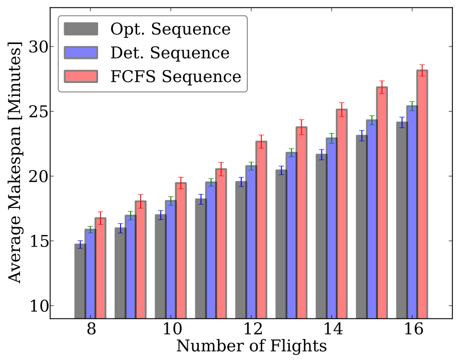
changing the upper limit. For arrivals, we consider upper limits of {3,6,9} and for departures we use {6,12,18} as upper limits. We denote the four different input distributions by *Dist. 1* through *Dist. 4*, in increasing order of uncertainty level. The outcome can be seen in Figure 22 with some additional details in Table 49. Table 49 shows, as expected, that the benefits of a stochastic scheduling algorithm increase as the level of uncertainty increases. However, even at relatively low levels of uncertainty, e.g. *Distr. 2*, the stochastic branch and bound algorithm generates significantly better solutions than a deterministic model.



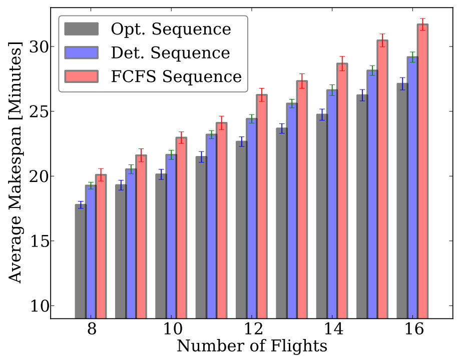
(a) Arrival Distribution: No uncertainty.
Departure Distribution: No uncertainty.



(b) Arrival Distribution: Triangular (0,1,3)
Departure Distribution: Triangular (0,1,6).



(c) Arrival Distribution: Triangular (0,1,6)
Departure Distribution: Triangular (0,1,12).



(d) Arrival Distribution: Triangular (0,1,9)
Departure Distribution: Triangular (0,1,18).

Figure 22: Average makespan for different levels of uncertainty around the nominal runway time. The triangular distributions are given as (lower limit, mode, upper limit)

5.6.3 Configuration 2: Departure Scheduling with Crossings

The second configuration is motivated by the work presented in Gupta, Malik, and Jung (2010) and Gupta, Malik, and Jung (2011), where the authors consider departure scheduling with taxiing arrivals crossing the departure runway, i.e. the sequencing and scheduling of arrivals is given. In this configuration, we assume that there is more than one runway crossing. We also assume that the decision maker is free to select any runway crossing for taxiing aircraft, but each taxiway leading up to the runway crossing can only hold a single aircraft.

Similar to Configuration 1, we treat the runway operations as a first stage decision and let the crossing operations be a recourse decision handled in the second stage. As we only consider departure operations in this configuration, the first stage solution space is significantly smaller, resulting in a smaller branching tree. To determine when crossings will occur, we use a simple decision rule for the second stage. Runway crossings take place when there is a gap in the departure stream that is large enough to allow crossings without interfering with departures, *or* if all taxiways leading up to the runway crossings are occupied. In the latter case, departures are held to clear all aircraft waiting to cross.

Due to the smaller solution space and the simple decision rule for when arrivals can cross, the algorithm allows for a larger set of aircraft to be included. To that end, we consider instances with 12 to 20 aircraft. Although the improvement in solution quality is relatively small as compared to a deterministic model or a FCFS policy, the results in Figure 23 and Table 50 indicate that there is benefit in using the stochastic branch and bound algorithm. The reason for the small improvement is that half of the first stage problem, i.e. sequencing of arrivals, has been eliminated. As the number of aircraft in an instance increases, the benefit of using the stochastic model also increases. This benefit can be seen both in terms of the average makespan increase for both the deterministic sequence and the FCFS sequence, and the percentage of instances in which H_0 is rejected.

In Figure 24, it is clear that this configuration is significantly easier to solve. This stems from two facts. First, in the first stage we only consider departing aircraft, which results in a much smaller branch and bound tree. Second, we use a simple decision rule in the second

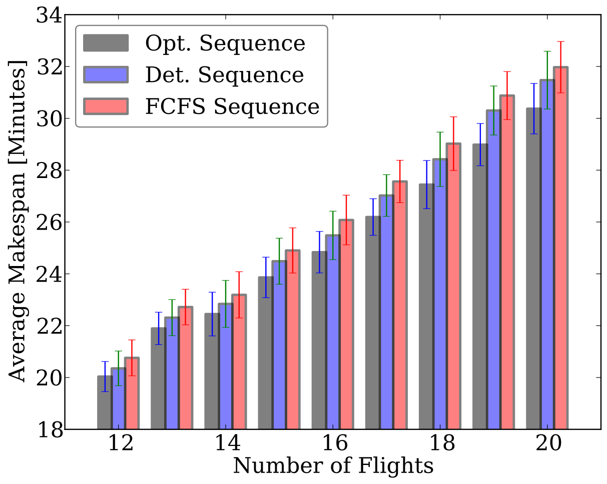


Figure 23: Average makespan for three different scheduling methods in the special case where only departures and runway crossings are considered: stochastic branch and bound algorithm (*Opt*), optimal deterministic sequence (*Det*), and a FCFS policy (*FCFS*). The error bars indicate the 95% confidence interval for the average makespan.

Table 50: Comparison between aircraft sequences obtained from the stochastic branch and bound algorithm (*Opt.*), the deterministic scheduling model (*Det.*) and the FCFS scheduling policy (*FCFS*). The result is for the special case when only departure operations and crossings are considered.

Number of	Avg. Makespan	Avg. Makespan Change		% H_0 rejected	
Flights	*Opt.*	*Det.* to *Opt.*	*FCFS* to *Opt.*	*Det.*	*FCFS*
12	20.04	1.58%	3.60%	37.78%	22.22%
13	21.89	1.90%	3.77%	31.11%	20.00%
14	22.45	1.74%	3.30%	35.56%	26.67%
15	23.86	2.62%	4.36%	40.00%	17.78%
16	24.83	2.61%	5.00%	37.78%	13.33%
17	26.19	3.17%	5.23%	28.89%	8.89%
18	27.44	3.56%	5.76%	24.44%	8.89%
19	28.98	4.54%	6.53%	17.78%	4.44%
20	30.37	3.62%	5.27%	24.44%	13.33%

H_0 : *Opt.* makespan < *Det.*/*FCFS* makespan

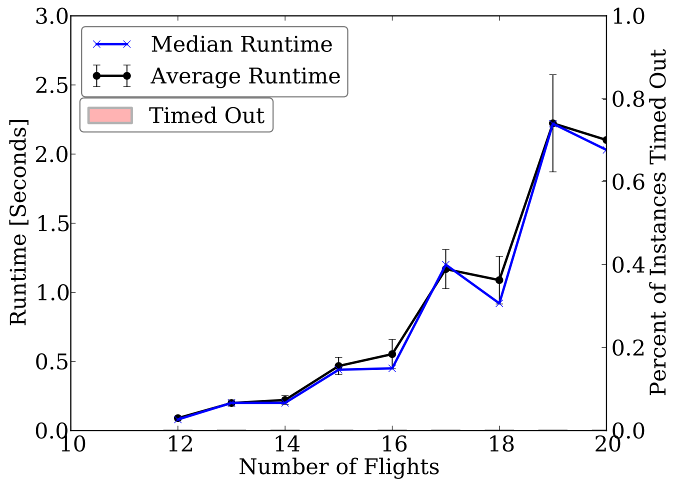


Figure 24: Average (with 95% confidence interval) and median runtime for instances solved by the stochastic branch and bound algorithm. The bars show the percentage of instances in which we were not able to find a solution within 20 minutes of runtime. The result is for the special case when only departure operations and crossings are considered.

stage which simplifies the evaluation of the second stage. This configuration gives us room for computational enhancements, both in terms of the number of aircraft we can include and the complexity of the second stage decision.

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH

This thesis addresses the stochastic airport runway scheduling problem in which a set of aircraft are to be scheduled on one or multiple dependent runways. In the literature review, we conclude that while the deterministic runway scheduling problem has received significant attention in the literature, the stochastic version has received no formal treatment. Solveling, Solak, Clarke, and Johnson (2011a) present the first two-stage stochastic programming formulation of the problem and propose a solution methodology based on Benders' decomposition. The model is simplified and does not capture all relevant aspect of the runway scheduling problem. Despite this fact, simulations performed in their study indicate that the proposed stochastic program adds value to the scheduling problem, when compared to deterministic models.

In Chapter 4, we propose a two-stage formulation that remedies the shortcomings of the model developed in Solveling, Solak, Clarke, and Johnson (2011a). The formulation, which captures all relevant information, results in a large scale stochastic program to which a solution methodology based on scenario decomposition is developed.

A second, sampling based, stochastic program tailored to sequencing problems is developed in Chapter 5. We enhance the stochastic branch and bound algorithm to generate runway schedules whose expected objective function values outperform deterministic scheduling models. Further discussions regarding the two proposed models and their performance are presented in subsequent subsections.

In the first part of the thesis we investigate the environmental value of optimization in runway scheduling, based on a global scheduling model that captures several complexities inherent in runway operations planning. The first complexity involves the trade-off between (sometimes conflicting) objective functions, and the second complexity involves the integrated cost structure associated with the re-sequencing of aircraft. To this end, we explicitly consider emission costs including CO_2 and several other pollutants, as well as noise costs. The cost functions developed in this chapter are later used in the two-stage model developed in Chapter 4.

We find that optimization based scheduling with explicit consideration of the environmental costs produce significant savings for both airlines and society. We also find that even if the environmental components are not directly included in the optimization, but instead a fuel-consumption based objective is used, the environmental savings over a FCFS policy are still significant. This implies that the additional operational costs incurred by airlines due to optimal schedules with reduced environmental impact are very minimal, especially at higher traffic volumes.

6.1 Conclusion for Two-Stage Sequencing Model

In Chapter 4, we consider the stochastic runway scheduling problem (SRSP) under uncertainty and propose two alternative stochastic programming formulations for this and other similarly structured scheduling problems. We analyze the two formulations from a computational efficiency perspective and develop tight valid inequalities to improve computational performance. As an efficient solution methodology, a Lagrangian decomposition scheme is used within a sample average approximation implementation with several special improvement steps. Using this framework, a detailed computational analysis is performed to demonstrate the challenges associated with SRSP and to identify the best formulations under different types of problem setups. In Table 6.1, we list the most efficient formulation for each problem category based on the experimental studies performed. The computational tests suggest that very high quality solutions can be obtained for SRSP when heuristic implementations based on truncated runs are used. This is especially relevant for practical implementability, as the efficiency and tractability of stochastic approaches have been the main reason for such approaches not being considered in the literature.

As potential procedures for further improvement in the algorithmic framework, some other upper bounding heuristics can be considered and analyzed within the Lagrangian decomposition scheme. A more specific quantitative measure can also be developed to

Table 51: The preferred models with respect to run times for the restricted SRSP (SRSP-R) and SRSP for different input schedules.

Problem	Characteristics	SRSP-R	SRSP
Deterministic Problem	Small Number of Aircraft	Flow	Flow/Slot
	Large Number of Aircraft	Flow	Flow/Slot
	Mixed Operations (all rates)	Flow	Flow
	Arrivals Only	Flow	Slot
Stochastic Problem With Limited Number of Scenarios	Small Number of Aircraft	Flow	Slot
	Large Number of Aircraft	Flow	Slot
	Mixed Operations (Lower Rate)	Flow	Slot
	Mixed Operations (Higher Rate)	Flow	Flow
	Arrivals Only	Flow	Slot

represent the relationship between run times and the value achieved from the runs, which may help better represent the tradeoffs involved. Also related to the practical aspect, detailed simulations comparing the performances of practical implementations of the FCFS, deterministic and stochastic scheduling procedures can be considered.

6.2 Conclusion for Single Stage Model

In Chapter 5.4, we use the stochastic branch and bound method to solve the runway scheduling problem and related machine scheduling problems with uncertain input parameters. More specifically, we assume that the time by which a job is available for scheduling is given in the form of a probability distribution. Given an input schedule and specified distributions describing the uncertainty, the algorithm generates a sequence of jobs (aircraft operations) that minimize the expected makespan or the expected system delay.

In order to obtain solutions in a short amount of time, we propose several enhancements to the stochastic branch and bound algorithm. By dynamically changing the number of samples used to estimate the upper and lower bounds during the course of the algorithm, we can place less emphasis on parts of the branch and bound tree that are unlikely to contain good solutions. Furthermore, we ensure that the algorithm always terminates with the best solution obtained so far, even if we have not found a complete sequence. With these enhancements, we are able to obtain high quality solutions using less than 10 minutes of computation time.

For a machine scheduling problem with sequence-dependent setup times and probabilistic release times, the proposed algorithm is able to reduce the makespan by up to 4% when compared to a deterministic model. Total system delay for sequences obtained by the algorithm can be reduced by up to 26% compared to sequences obtained from deterministic models. The proposed enhancements to the algorithm are able to reduce runtime by up to 30% for the makespan objective and up to 70% when system delay is considered.

In the later part of the chapter, the algorithm is applied to two instances of the stochastic airport runway scheduling problem. The computational results in this study indicate that the makespan decreases by 5% to 8% when using the stochastic branch and bound algorithm as compared to an aircraft sequence obtained from a deterministic model. This represents a significant saving if the result can be translated to a longer planning horizon.

In this work, we assume that the deviation from earliest runway time is independent between aircraft, whereas in reality there are several cases where delay is dependent. As an example, consider a sequence of departing aircraft taxiing along the same taxiway. If the first aircraft is delayed, the remaining aircraft are likely to be delayed as well. The impact of dependent aircraft operations is a direction of future research.

One major challenge in the stochastic branch and bound algorithm is to estimate good lower bounds in a short amount of time. In this application we use a partial enumeration scheme, taking advantage of heuristics incorporating constrained position shifts. It is not within the scope of this thesis to explore the trade-off between runtime and solution quality for different lower (and upper) bound models, rather, we suggest that as another direction of future research.

6.3 Directions for Future Research

The work presented in this thesis focuses on improved solution methodologies for the airport runway scheduling problem and similarly structured problems. This is an important area due to the dynamic environment faced by air traffic controllers, where decision support systems need to produce good quality solutions fast. Directions for future research in this area include continued development of both heuristic and optimal scheduling methods.

Future research efforts also need focus on various aspects of the control problem, i.e. how can air traffic controllers achieve the recommendations provided by a decision support system. There are several issues that need to be investigated in order to use stochastic runway scheduling methods in practice. For example, can the recommended sequence be achieved given the current status of the system? If not, the models need to take these constraints into account. How should traffic be managed to achieve a specific weight class sequence before the weight class sequence has been determined? What recourse actions are available if a weight class sequence is not achievable? Questions like these need to be further investigated before a two-stage model can be used in practice. We also suggest a deeper analysis of the stochastic characteristics inherent to the problem, e.g. what happens when we remove the assumption of independent aircraft operations.

REFERENCES

Agustín, A., Alonso-Ayuso, A., Escudero, L., and Pizarro, C. (2012). "On Air Traffic Flow Management with Rerouting. Part II: Stochastic Case". *European Journal of Operational Research* vol. 219, no. 1, pp. 167–177.

Ahmed, S. (2010). "Two-Stage Stochastic Integer Programming: A Brief Introduction". In: *Wiley Encyclopedia of Operations Research and Management Science*. Ed. by J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith. John Wiley & Sons, Inc.

Ahmed, S., Tawarmalani, M., and Sahinidis, N. (2004). "A Finite Branch-and-Bound Algorithm for Two-Stage Stochastic Integer Programs". *Mathematical Programming* vol. 100, no. 2, pp. 355 –377.

Allahverdi, A., Ng, C., Cheng, T., and Kovalyov, M. (June 2008). "A Survey of Scheduling Problems with Setup Times or Costs". *European Journal of Operational Research* vol. 187, no. 3, pp. 985–1032.

Alonso, A., Escudero, L. F., and Ortuno, M. T. (2000). "A Stochastic 01 Program Based Approach for the Air Traffic Flow Management Problem". *European Journal of Operational Research* vol. 120, no. 1, pp. 47–62.

Anagnostakis, I. (2004). "A Multi-Objective, Decomposition-Based Algorithm Design Methodology and its Application to Runway Operations Planning". PhD thesis. Massachusetts Institute of Technology.

Anagnostakis, I. and Clarke, J.-P. (Jan. 2003). "Runway Operations Planning: A Two-Stage Solution Methodology". In: *Proceedings of the 36th Hawaii International Conference on System Sciences*. Honolulu, HI.

Atkin, J., Burke, E., Greenwood, J., and Reeson, D. (2008). "On-line Decision Support for Take-off Runway Scheduling with Uncertain Taxi Times at London Heathrow Airport". *Journal of Scheduling* vol. 11, no. 5, pp. 323–346.

Balakrishan, H. and Chandran, B. (Nov. 2010). "Algorithms for Scheduling Runway Operations Under Constrained Position Shifting". *Operations Research* vol. 58, no. 6.

Ball, M. et al. (2010). *Total Delay Impact Study – A Comprehensive Assessment of the Costs and Impacts of Flight Delay in the United States.* Tech. rep. NEXTOR.

Ball, M. O., Hoffman, R., Odoni, A. R., and Rifkin, R. (Mar. 2003). "A Stochastic Integer Program with Dual Network Structure and Its Application to the Ground-Holding Problem". *Operations Research* vol. 51, no. 1, pp. 167–171.

Beasley, J., Krishnamoorthy, M., Sharaiha, Y., and Abramson, D. (2004). "Displacement Problem and Dynamically Scheduling Aircraft Landings". *Journal of the Operational Research Society* vol. 55, pp. 54–64.

Beasley, J., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D. (May 2000). "Scheduling Aircraft Landings–The Static Case". *Transportation Science* vol. 34, no. 2, pp. 180–197.

Benders, J. F. (1962). “Partitioning Procedures for Solving Mixed-Variables Programming Problems”. *Numerische Mathematik* vol. 4, pp. 238 –252.

Bennell, J., Mesgarpour, M., and Potts, C. (2011). “Airport Runway Scheduling”. *4OR: A Quarterly Journal of Operations Research* vol. 9, pp. 115–138.

Bianco, L., Dell’Olmo, P., and Giordani, S. (1997). “Scheduling Models and Algorithms for TMA Traffic Management”. In: *Modeling and Simulation in Air Traffic Management.* Ed. by L. Bianco, P. Dell’Olmo, and A. Odoni. Berlin, Germany: Springer-Verlag, pp. 139–167.

Bianco, L., Rinaldi, G., and Sassano, A. (1987). “A Combinatorial Optimization Approach to Aircraft Sequencing Problem”. In: *Flow Control of Congested Networks.* Ed. by A. Odini, L. Bianco, and G. Szego. Vol. 38. NATO ASI Series, Series F: Computer and Systems Science. Berlin, Germany: Springer-Verlag, pp. 323–339.

Bigras, L.-P., Gamache, M., and Savard, G. (2008). “The Time-Dependent Traveling Salesman Problem and Single Machine Scheduling Problems with Sequence Dependent Setup Times”. *Discrete Optimization* vol. 5, pp. 685–699.

Birge, J. R. and Louveaux, F. V. (1988). “A Multicut Algorithm for Two-Stage Stochastic Linear Programs”. *European Journal of Operational Research* vol. 34, no. 3, pp. 384 –392.

Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., and Sudan, M. (May 1994). “The minimum latency problem”. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing.* Montreal, QC, Canada, pp. 163–171.

Bly, E. (2005). “Effects of Reduced IFR Arrival-Arrival Wake Vortex Separation Minima and Improved Runway Operations Sequencing on Flight Delay”. M.S. thesis. Massachusetts Institute of Technology.

Brentnall, A. and Cheng, R. (2009). “Some Effects of Aircraft Arrival Sequence Algorithms”. *Journal of the Operational Research Society* vol. 60, pp. 962–972.

Brinton, C. (Oct. 1992). “An Implicit Enumeration Algorithm for Arrival Aircraft Scheduling”. In: *Proceedings of the 11th IEEE/AIAA Digital Avionics Systems Conference.* Seattle,WA.

Brinton, C., Cook, L., and Atkins, S. (2007). “Collaborative Airport Surface Metering for Efficiency and Environmental Benefits”. In: *Proceedings of Integrated Communications, Navigation and Surveillance Conference.* Herndon, VA.

Cai, X. and Zhou, X. (2005). “Single-Machine Scheduling with Exponential Processing Times and General Stochastic Cost Functions”. *Journal of Global Optimization* vol. 31, pp. 317 –332.

Caroe, C. and Schultz, R. (1999). “Dual Decomposition in Stochastic Integer Programming”. *Operations Research Letters* vol. 24, pp. 37–45.

Carøe, C. and Tind, J. (1998). “L-shaped Decomposition of Two-Stage Stochastic Programs with Integer Recourse”. *Mathematical Programming* vol. 83, no. 1, pp. 451 – 464.

Celikel, A., Hustache, J.-C., Lepinay, I., Martin, K., and Melrose, A. (2005). "Environmental Tradeoffs Assessment Around Airports". In: *Proceedings of the 6th USA/Europe ATM Seminar.* Baltimore, MD.

Chandran, B. and Balakrishan, H. (July 2007). "A Dynamic Programming Algorithm for Robust Runway Scheduling". In: *Proceedings of the American Control Conference.* New York, NY.

Chang, Y.-H. (2010). "Stochastic Programming Approaches to the Air Traffic Flow Management Under the Uncertainty of Weather". PhD thesis. Georgia Institute of Technology.

Codato, G. and Fischetti, M. (2006a). "Combinatorial Benders' Cuts for Mixed-Integer Linear Programming". *Operations Research* vol. 54, no. 4, pp. 756–766.

Codato, G. and Fischetti, M. (2006b). "Combinatorial Benders Cuts for Mixed-Integer Linear Programming". *Operations Research* vol. 54, no. 4, pp. 756 –766.

Cook, A., Tanner, G., and Anderson, S. (2004). *Evaluating the True Cost to Airlines of One Minute of Airborne or Ground Delay.* Tech. rep. Eurocontrol Performance Review Unit.

Dear, R. (1976). *The Dynamic Scheduling of Aircraft in the Near Terminal Area.* Tech. rep. Massachusetts Institute of Technology.

Dear, R. and Sherif, Y. (1991). "An Algorithm for Computer Assisted Sequencing and scheduling of Terminal Area Operations". *Transportation Research: Part A* vol. 25, no. 2-3, pp. 129–139.

Doyle, T. and McGee, F. (1998). *Air Traffic and Operational Data on Selected U.S. Airports with Parallel Runways.* Tech. rep. NASA/CR-1998-207675. Langley Research Center, Hampton, VA 23681: National Aeronautics and Space Administration.

Dubois, D. and Paynter, G. (Aug. 2006). "Fuel Flow Method 2 for Estimating Aircraft Emissions". In: *Society of Automotive Engineers.* 400 Commonwealth Dr, Warrendale PA 15096, USA.

EIA (Sept. 2009). *Energy Information Administration: Voluntary Reporting of Greenhouse Gases Program.* `http://www.eia.doe.gov/oiaf/1605/coefficients.html`.

Eren, T. and Güner, E. (Aug. 2006). "A Bicriteria Scheduling with Sequence-Dependent Setup Times". *Applied Mathematics and Computation* vol. 179, no. 1, pp. 378–385.

Ernst, A., Krishnamoorthy, M., and Storer, R. (1999). "Heuristic and Exact Algorithms for Scheduling Aircraft Landings". *Networks* vol. 34, no. 2, pp. 229–241.

Eun, Y., Hwang, I., and Bang, H. (June 2010). "Optimal Arrival Flight Sequencing and Scheduling Using Discrete Airborne Delays". *IEEE Transactions on Intelligent Transportation Systems* vol. 11, no. 2, pp. 359–373.

Eurocontrol (2011). *SESAR Research: ATM Operations and System Validation.* http://-www.eurocontrol.int/articles/atm-operations-and-systems-validation. Accessed in July 2011.

Federal Aviation Administration (2009). *Report to Congress: National Plan of Integrated Airport Systems (2009-2013).* Tech. rep. Federal Aviation Administration.

Federal Aviation Administration (2010a). *Federal Aviation Administration Order JO 7110.65T Air Traffic Control.* Effective February 11, 2010.

Federal Aviation Administration (2010b). *Order JO 7110.65T Air Traffic Control.* Effective February 11, 2010.

Federal Aviation Administration (2011). *NextGen Portfolio.* http://www.faa.gov/-nextgen/portfolio/. accessed in March 2011.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York, NY: W. H. Freeman and Company.

Geoffrion, A. (1972). “Generalized Benders Decomposition”. *Journal of Optimization Theory and Applications* vol. 10, no. 4, pp. 237–260.

Glover, C. (2010). “Computationally Tractable Stochastic Integer Programming Models for Air Traffic Flow Management”. PhD thesis. University of Maryland.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. (1979). “Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey”. *Annals of Discrete Mathematics* vol. 5, pp. 287 –326.

Guan, Y., Ahmed, S., and Nemhauser, G. L. (Mar. 2009). “Cutting Planes for Multistage Stochastic Integer Programs”. *Operations Research* vol. 57, no. 2, pp. 287–298.

Gupta, G., Malik, W., and Jung, Y. (Sept. 2009). “A Mixed Integer Linear Program for Airport Departure Scheduling”. In: *Proceedings of the 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO).* Hilton Head, South Carolina, USA.

Gupta, G., Malik, W., and Jung, Y. (Aug. 2010). “Incorporating Active Runway Crossings in Airport Departure Scheduling”. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference.* Toronto, ON.

Gupta, G., Malik, W., and Jung, Y. (Sept. 2011). “Effect of Uncertainty on Deterministic Runway Scheduling”. In: *Proceedings of the 11th AIAA Aviation Technology, Integration, and Operations Conference (ATIO).* Virginia Beach, VA.

Gupta, S. R. and Smith, J. S. (2006). “Algorithms for Single Machine Total Tardiness Scheduling with Sequence Dependent Setups”. *European Journal of Operational Research* vol. 175, pp. 722 –732.

Gutjahr, W., Hellmayr, A., and Pflug, G. (1999). “Optimal Stochastic Single-Machine-Tardiness Scheduling by Stochastic Branch-and-Bound”. *European Journal of Operational Research* vol. 117, pp. 396–413.

Gutjahr, W., Strauss, C., and Wagner, E. (2000). “A Stochastic Branch-and-Bound Approach to Activity Crashing in Project Management”. *INFORMS Journal on Computing* vol. 12, no. 7, pp. 125–135.

Heilporn, G., Cordeau, J.-F., and Laporte, G. (2010a). “The Delivery Man Problem with Time Windows”. *Discrete Optimization* vol. 7, pp. 269–282.

Heilporn, G., Cordeau, J.-F., and Laporte, G. (2010b). "The Delivery Man Problem with Time Windows". *Discrete Optimization* vol. 7, pp. 269–282.

Hiriart-Urruty, J.-B. and Lemarechal, C. (Oct. 1993). *Convex Analysis and Minimization Algorithms I.* Springer. isbn: 0387568506.

Hsu, C. and Lin, P. (2005). "Performance Assessment for Airport Noise Charge Policies and Airline Network Adjustment Response". *Transportation Research Part D* vol. 10, no. 4, pp. 281–304.

Hu, X. and Chen, W. (June 2005). "Receding Horizon Control for Aircraft Arrival Sequencing and Scheduling". *IEEE Transactions on Intelligent Transportation Systems* vol. 6, no. 2, pp. 189–197.

Hu, X. and Paolo, E. D. (June 2008). "Binary-Representation-Based Genetic Algorithm for Aircraft Arrival Sequencing and Scheduling". *IEEE Transactions on Intelligent Transportation Systems* vol. 9, no. 2, pp. 301–310.

Hu, X. and Paolo, E. D. (2011). "A Ripple-Spreading Genetic Algorithm for the Aircraft Sequencing Problem". *Evolutionary Computation* vol. 19, no. 1, pp. 77–106.

Idris, H. et al. (Aug. 1998). "Identification of Flow Constraints and Control Points in Departure Operations at Airport Systems". In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. Boston, MA.

Kesgin, U. (2006). "Aircraft Emissions at Turkish Airports". *Energy* vol. 31, pp. 372–384.

Kiwiel, K. C. (1990). "Proximity Control in Bundle Methods for Convex Nondifferentiable Minimization". *Mathematical Programming* vol. 46, no. 1, pp. 105 –122.

Kleywegt, A., Shapiro, A., and De-Mello, T. (2002). "The Sample Average Approximation Method for Stochastic Discrete Optimization". *SIAM Journal on Optimization* vol. 12, no. 2, pp. 479–502. issn: 1052-6234. doi: `http://dx.doi.org/10.1137/S1052623499363220`.

Kotnyek, B. and Richetta, O. (May 2006). "Equitable Models for the Stochastic Ground-Holding Problem Under Collaborative Decision Making". *Transportation Science* vol. 40, no. 2, pp. 133–146.

Laporte, G. and Louveaux, F. V. (Apr. 1993). "The Integer L-shaped Method for Stochastic Integer Programs with Complete Recourse". *Operations Research Letters* vol. 13, no. 3, pp. 133 –142.

Lee, S. M. and Asllani, A. A. (2004). "Job scheduling with dual criteria and sequence-dependent setups: mathematical versus genetic programming". *Omega* vol. 32, pp. 145 –153.

Levinson, D., Kanafani, A., and Gillen, D. (1999). "Air, High Speed Rail or Highway: A Cost Comparison in the California Corridor". *Transportation Quarterly* vol. 53, no. 1, pp. 123–132.

Linderoth, J., Shapiro, A., and Wright, S. (2006). "The empirical behavior of sampling methods for stochastic programming". *Annals OR* vol. 142, no. 1, pp. 215–241.

Louveaux, F. and Schultz, R. (2003). "Handbooks in Operations Research and Management Science". In: ed. by A. Ruszczynski and A. Shapiro. Vol. 10. Elsevier Science. Chap. 4, pp. 213–266.

Mesgarpour, M., Potts, C., and Bennell, J. (June 2010). "Models for Aircraft Landing Optimization". In: *Proceedings of the 4th International Conference on Research in Air Transportation.* Budapest, Hungary.

Möller, A., Römisch, W., and Weber, K. (2007). "Airline Network Revenue Management by Multistage Stochastic Programming". *Computational Management Science* vol. 5, no. 4, pp. 355–377.

Monroe, G., Jung, Y., and Tobias, L. (Aug. 2008). "Analysis of Environmental Impact of Eliminating Arrival Hold Short Operations for Runway Crossings at Dallas/Ft. Worth Airport". In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference.* Honolulu, HI.

Mukherjee, A. and Hansen, M. (Nov. 2007). "A Dynamic Stochastic Model for the Single Airport Ground Holding Problem". *Transportation Science* vol. 41, no. 4, pp. 444–456.

Mulvey, J. M. and Ruszczyski, A. (May 1995). "A New Scenario Decomposition Method for Large-Scale Stochastic Optimization". *Operations Research* vol. 43, no. 3, pp. 477 –490.

Nelson, J. (Jan. 2004). "Meta-Analysis of Airport Noise and Hedonic Property Values". *Journal of Transport Economics and Policy* vol. 38, no. 1, pp. 1–27.

Nero, G. and Black, J. (1998). "Hub-and-spoke Networks and The Inclusion of Environmental Costs on Airport Pricing". *Transportation Research Part D* vol. 3, no. 5, pp. 275–296.

NIST/SEMATECH (2012). *e-Handbook of Statistical Methods.* http://www.itl.nist.gov-/div898/handbook/. Access date: 4/18/2012.

Norkin, V. I., Ermoliev, Y., and Ruszczyński, A. (1998). "On Optimal Allocation of Indivisibles under Uncertainty". *Operations Research* vol. 46, no. 3, pp. 381–395.

Norkin, V. I., Pflug, G., and Ruszczyński, A. (1998). "A Branch and Bound Method for Stochastic Global Optimization". *Mathematical Programming* vol. 83, pp. 425–450.

Picard, J.-C. and Queyranne, M. (Jan. 1978). "The Time-Dependent Traveling Salesman Problem and its Application to the Tardiness Problem in One-Machine Scheduling". *Operations Research* vol. 6, no. 1, pp. 86–110.

Psaraftis, H. (1980). "A Dynamic Programming Approach for Sequencing Groups of Identical Jobs". *Operations Research* vol. 28, pp. 1347–1359.

Rathinam, S., Wood, Z., Sridhar, B., and Jung, Y. (Aug. 2009). "A Generalized Dynamic Programming Approach for a Departure Scheduling Problem". In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference.* Chicago, IL.

Ren, L. and Clarke, J.-P. (2008). "Flight-Test Evaluation of the Tool for Analysis of Separation and Throughput". *Journal of Aircraft* vol. 45, pp. 323–332.

Rockafellar, R. T. and Wets, R. J.-B. (Feb. 1991). “Scenarios and Policy Aggregation in Optimization under Uncertainty”. *Mathematics of Operations Research* vol. 16, no. 1, pp. 119 –147.

Rosa, C. and Ruszczyski, A. (1996). “On Augmented Lagrangian Decomposition Methods for Multistage Stochastic Programs”. *Annals of Operations Research* vol. 64, no. 1, pp. 289 –309.

Ruszczyski, A. (1986). “A Regularized Decomposition Method for Minimizing a Sum of Polyhedral Functions”. *Mathematical Programming* vol. 35, no. 3, pp. 309 –333.

Sen, S. and Higle, J. L. (Mar. 1999). “An Introductory Tutorial on Stochastic Linear Programming Models”. *Interfaces* vol. 29, no. 2, pp. 33 –61.

Sen, S. and Higle, J. L. (2005). “The C^3 theorem and a D^2 algorithm for large scale stochastic integer programming: Set convexification”. *Mathematical Programming* vol. 104, no. 1, pp. 1 –20.

Shapiro, A. (2003). “Inference of statistical bounds for multistage stochastic programming problems”. *Mathematical Methods of Operations Research* vol. 58, pp. 57–68.

Sherali, H., Ghoniem, A., Baik, H., and Trani, A. (2012). “A Combined Arrival-Departure Aircraft Sequencing Problem”. Submitted Manuscript.

Sherali, H. and Fraticelli, B. (2002). “A Modification of Benders’ Decomposition Algorithm for Discrete Subproblems: An Approach for Stochastic Programs with Integer Recourse”. *Journal of Global Optimization* vol. 22, no. 1, pp. 319 –342.

Simaiakis, I. and Balakrishan, H. (Aug. 2009). “Queuing Models of Airport Departure Processes for Emissions Reduction”. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference.* Chicago, IL.

Skutella, M. and Uetz, M. (2005). “Stochastic Machine Scheduling with Precedence Constraints”. *SIAM Journal on Computing* vol. 34, no. 4, pp. 788 –802.

Slyke, R. M. V. and Wets, R. (1969). “L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming”. *SIAM Journal on Applied Mathematics* vol. 17, no. 4, pp. 638 –663.

Solveling, G., Solak, S., Clarke, J., and Johnson, E. (2011a). “Runway Operations Optimization in the Presence of Uncertainties”. *Journal of Guidance, Control, and Dynamics* vol. 34, no. 5, pp. 1373–1382.

Solveling, G., Solak, S., Clarke, J., and Johnson, E. (2011b). “Scheduling of Runway Operations for Reduced Environmental Impact”. *Transportation Research Part D* vol. 16, no. 2, pp. 110–120.

Soroush, H. M. and Fredendall, L. (1994). “The stochastic single machine scheduling problem with earliness and tardiness costs”. *European Journal of Operational Research* vol. 77, pp. 287 –302.

Tan, K.-C., Narasimhan, R., Rubin, P. A., and Ragatz, G. L. (2000). “A Comparison of Four Methods for Minimizing Total Tardiness on a Single Processor with Sequence Dependent Setup Times”. *Omega* vol. 28, pp. 313 –326.

Trivizas, D. (1998). "Optimal Scheduling with Maximum Position Shift Constraints". *Journal of Navigation* vol. 51, no. 2, pp. 250–266.

Venkatakrishnan, C., Barnett, A., and Odoni, A. (Aug. 1993). "Landings at Logan Airport: Describing and Increasing Airport Capacity". *Transportation Science* vol. 27, no. 3, pp. 211–227.

Wu, X. and Zhou, X. (Oct. 2008). "Stochastic Scheduling to Minimize Expected Maximum Lateness". *European Journal of Operational Research* vol. 190, no. 1, pp. 103–115.

Yen, J. W. and Birge, J. R. (Feb. 2006). "A Stochastic Programming Approach to the Airline Crew Scheduling Problem". *Transportation Science* vol. 40, no. 1, pp. 3–14.

Yu, S., X.Cao, and Zhang, J. (2011). "A real-time schedule method for Aircraft Landing Scheduling problem based on Cellular Automation". *Applied Soft Computing* vol. 11, pp. 3485–3493.

Zhang, L. and Zheng, W. (1996). "On Some Single-Machine Scheduling with Sequence Dependent Set-up Times". In: *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics.* Vol. 2. Beijing , China, pp. 1162–1165.